Fundación ObservatorioFiscal

Modelo de red flags para compras públicas

Licitaciones Públicas

José Mora González,

Adam Waitzer

Apoyado por Open Contracting Partnership

Observatorio del Gasto Fiscal en Chile

Santiago, Chile

Abril, 2020

Contenido

1. Introducción

Uno de los aspectos más relevantes para la correcta ejecución de los recursos públicos son las compras que realiza el Estado, su riesgo es alto y el nivel de transparencia es aún insuficiente.

Debido a lo anterior es que el Observatorio del Gasto Fiscal comenzó la elaboración de un sistema que permita detectar procesos que puedan contener algún tipo de vicio, que lo vuelva ineficiente o vulnerable a la corrupción. Para esto se definió un marco de trabajo y análisis en que se analizó cada una de las distintas etapas del proceso de compras y se aplicará mediante distintos indicadores de nivel de riesgo de cada proceso de adquisiciones.

El presente documento registra este proceso inicial de definiciones y búsqueda de bibliografía.

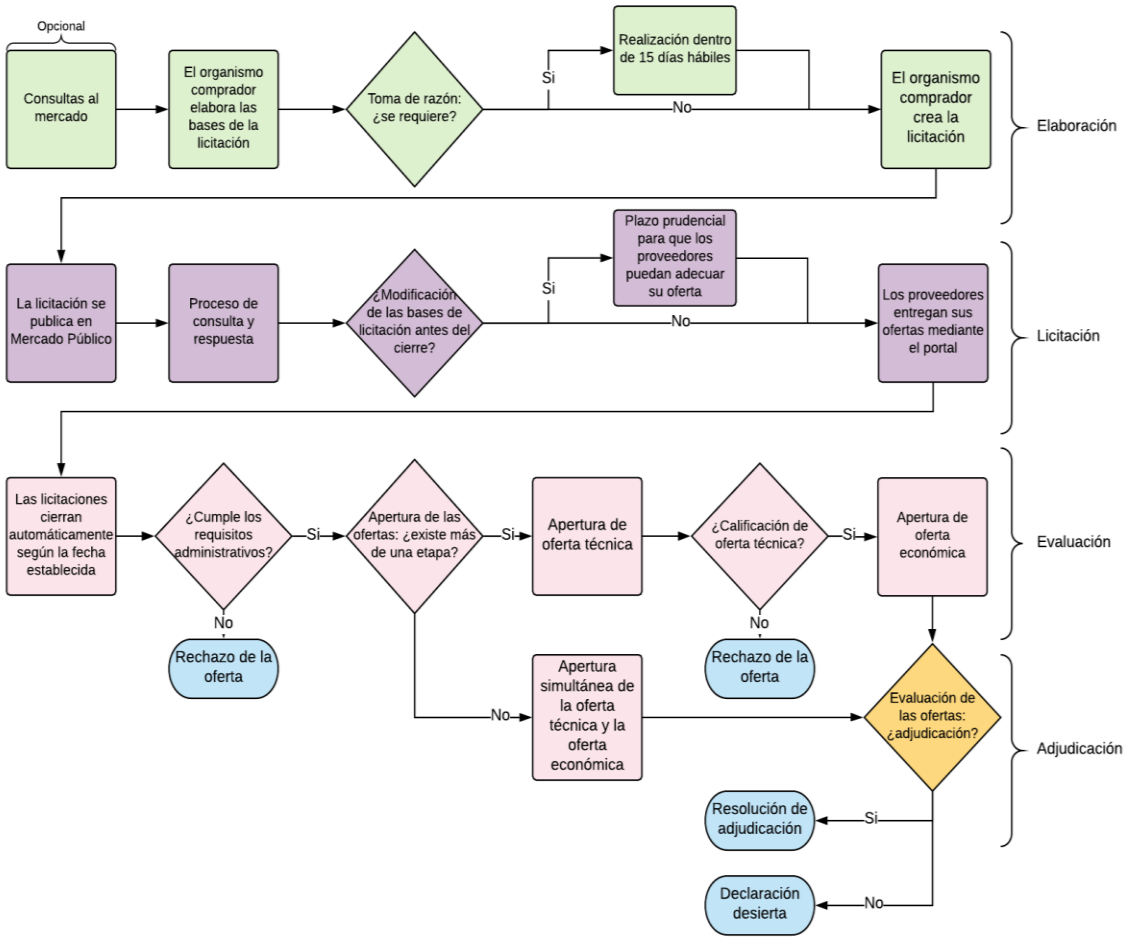
Los organismos compradores de Chile pueden realizar la adquisición de los bienes y servicios a través de cuatro modalidades: licitaciones públicas, licitaciones privadas, trato directo y convenios marco. El presente documento trata específicamente de las licitaciones públicas. Lo anterior se debe a que la implementación de este modelo será de manera gradual, comenzando por la modalidad mencionada, estableciendo cuales son los indicadores tratados en literatura nacional e internacional, para luego modelar su comportamiento y establecer que procesos se comportan de manera similar.

2. Definiciones

La licitación pública es un procedimiento administrativo mediante el cual un organismo comprador invita a los proveedores interesados a proporcionar un bien o servicio y selecciona la oferta más ventajosa. Por ley, los organismos están obligados a realizar licitaciones públicas por contrataciones que superen las 1.000 UTM. No obstante, según lo establecido en Artículo 10 de la Ley de Compras (Nº 19.886) los organismos pueden, en casos fundados, realizar sus procesos de contratación a través de otras modalidades.

A continuación, en la figura 1, se presenta el flujograma de cómo se lleva a cabo un proceso de licitación pública genérico, esto con la finalidad de detectar etapas y actividades claves que se puedan asociar a un indicador de riesgo.

Figura 1: Flujograma del procedimiento de la licitación pública



Elaboración propia en base a Documento de capacitación de ChileCompra para licitaciones públicas [1] [2]

Como se puede apreciar en la figura 1, se dividió el proceso en cuatro etapas: Elaboración, Licitación, Evaluación y Adjudicación.

Cada una de estas etapas tiene una serie de actividades cuyo detalle será definido a continuación, es importante señalar que las actividades corresponden a aquellas que se simbolizan con un cuadrado dentro del flujograma.

2.1. Descripción

2.1.1. Consultas al mercado

Según lo establecido en Artículo 13 de la Ley de Compras, los organismos compradores pueden efectuar procesos de consultas o reuniones con proveedores antes de que se elaboren las bases de licitación. Estos procesos ocurren mediante llamados públicos y abiertos, convocados a través del portal Mercado Público. Las consultas al mercado permiten a los organismos compradores obtener información acerca de los precios, características de los bienes o servicios requeridos, tiempos de preparación de las ofertas, o cualquier otra que se requiera para la elaboración de las bases.

2.1.2. Creación de la licitación

El organismo comprador elabora las bases de la licitación y la crea a través del portal Mercado Público. Por ley, las bases de licitación deben establecer las condiciones que permiten alcanzar la combinación más ventajosa entre todos los beneficios del bien o servicio por adquirir y todos sus costos asociados, presentes y futuros. El usuario comprador tiene que precisar el precio, la cantidad, el monto total, y la moneda de la adquisición. Durante el procedimiento de la creación de la licitación, el portal Mercado Público avisará al usuario comprador si ya existe una licitación parecida o si existe un convenio marco vigente que corresponde al bien o servicio requerido.

Las licitaciones públicas deben contener, a lo menos, las siguientes materias:

1. Los requisitos y condiciones que deben cumplir los oferentes para que sus ofertas sean aceptadas.

2. Las especificaciones de los bienes o servicios que se quieren contratar, las cuales deben ser genéricas, sin hacer referencia a marcas específicas.

3. Las etapas y plazos de la licitación, los plazos y modalidades de aclaración de las bases, la entrega y la apertura de las ofertas, la evaluación de las ofertas, la adjudicación y la firma del Contrato de Suministro y Servicio respectivo y el plazo de duración de dicho contrato.

4. La condición, el plazo y el modo en que se compromete el o los pagos del Contrato de Suministro y Servicio, una vez recepcionados conforme los bienes o servicios de que se trate.

5. El plazo de entrega del bien o servicio adjudicado.

6. La naturaleza y monto de la o las garantías que la entidad licitante exija a los oferentes y la forma y oportunidad en que serán restituidas.

7. Los criterios objetivos que serán considerados para decidir la adjudicación, atendido la naturaleza de los bienes y servicios que se licitan, la idoneidad y calificación de los oferentes y cualquier otro antecedente que sea relevante para efectos de la adjudicación.

8. El nombre completo del funcionario de la entidad licitante encargado del proceso de compras y el medio de contacto.

9. Los medios para acreditar si el proveedor adjudicado registra saldos insolutos de remuneraciones o cotizaciones de seguridad social con sus actuales trabajadores o con trabajadores contratados en los últimos dos años y la oportunidad en que ellos serán requeridos.

10. La forma de designación de las comisiones evaluadoras, que se constituirán de conformidad con lo establecido en el Artículo 37 de la Ley de Compras.

Recientemente, la Dirección ChileCompra ha introducido un nuevo aplicativo de Licitación Simplificada. El nuevo aplicativo permite llevar a efecto las compras por licitación menores a UTM 100 de una manera más simple y automatizada, permitiendo una mayor eficiencia en el proceso que esto involucra. Entre otros beneficios, el nuevo aplicativo permite acceder a cláusulas estandarizadas previamente aprobados por la Controlaría, no exige la incorporación de anexos adicionales al proceso y permite realizar el proceso de forma totalmente digital.

2.1.3. Publicación de la licitación

La licitación requiere autorización antes de que se publique en Mercado Público, lo que depende del trámite administrativo de cada institución compradora. Una vez publicada, se determina el plazo mínimo asociado a la licitación según el monto de la adquisición del bien o la contratación del servicio:

- L1: Licitaciones inferiores a 100 UTM / Mínimo 5 días
- LE: Licitaciones > = 100 y < 1.000 UTM / Mínimo 10 días
- LP: Licitaciones > = 1.000 y < 2.000 UTM / Mínimo 20 días
- LQ: Licitaciones > = 2.000 y < 5.000 UTM / Mínimo 20 días
- LR: Licitaciones > = 5.000 UTM / Mínimo 30 días

2.1.4. Recepción y apertura de las ofertas

Una vez publicada, los oferentes ingresan al portal los antecedentes requeridos en las bases de licitación. Los oferentes pueden enviar sus ofertas irrevocablemente hasta la fecha de cierre, establecido por adelantado según las bases de licitación. En caso de que las bases sean modificadas antes de la fecha de cierre, debe considerarse un plazo prudencial para que los proveedores interesados puedan adecuar su oferta a tales modificaciones.

Las bases de licitación establecen si la apertura de licitaciones ocurre en una etapa o en dos etapas. La apertura de licitaciones en una etapa consiste en que se procede a abrir tanto la oferta técnica como la oferta económica. De otra manera, la apertura de licitaciones en dos etapas consiste en que la apertura de la oferta técnica precede la apertura de la oferta económica. Así, se evalúa en detalle la calidad técnica de las propuestas, sin sesgos originados por la oferta económica.

2.1.5. Evaluación de las ofertas

El organismo comprador debe evaluar las ofertas de los proveedores mediante una comisión evaluadora, cuyos miembros se nombran según las bases de licitación. Los miembros de la comisión evaluadora no pueden tener conflictos de intereses con los oferentes, y los contactos entre los oferentes y el organismo comprador son regulados por Artículo 39 de la Ley de Compras. En las licitaciones en las que la evaluación de las ofertas revista gran complejidad y en todas aquellas superiores a 1.000 UTM, las ofertas deben ser evaluadas por una comisión de al menos tres funcionarios públicos, internos o externos al organismo respectivo, de manera garantizar la imparcialidad y competencia entre los oferentes.

La comisión evaluadora asigna puntajes de acuerdo con los criterios que se establecen en las bases de licitación. Se puede considerar como criterios técnicos o económicos el precio, la experiencia, la metodología, la calidad técnica, la asistencia técnica o soporte, los servicios de postventa, los plazos de entrega, los recargos por fletes, consideraciones medioambientales, de eficiencia energética, los consorcios entre oferentes, el comportamiento contractual anterior, el cumplimiento de los requisitos formales de la oferta, así como cualquier otro criterio que sea pertinente a los bienes o servicios licitados.

En el caso de la prestación de servicios habituales, que deben proveerse a través de licitaciones o contrataciones periódicas, las bases deberán contemplar como criterio técnico las condiciones de empleo y remuneración. Para evaluar este criterio, se podrán considerar como factores de evaluación el estado de pago de las remuneraciones y cotizaciones de los trabajadores, la contratación de discapacitados, el nivel de remuneraciones sobre el sueldo mínimo, la composición y reajuste de las remuneraciones, la extensión y flexibilidad de la jornada de trabajo, la duración de los contratos, la existencia de incentivos, así como otras condiciones que resulten de importancia en consideración a la naturaleza de los servicios contratados.

2.1.6. Adjudicación de la oferta y notificación

Después de haber evaluado las ofertas, el organismo comprador debe publicar los resultados del proceso de licitación. El organismo comprador acepta una oferta mediante acto administrativo, debidamente notificado al adjudicatario y al resto de los oferentes. Asimismo, debe publicar la resolución fundada que declare la inadmisibilidad o la declaración de desierto del proceso. Cuando la adjudicación no se realice dentro del plazo señalado en las bases de licitación, el organismo debe informar en el portal Mercado Público las razones que justifican el incumplimiento e indicar un nuevo plazo para la adjudicación.

3. Modelo de red flags

El proyecto contempla una serie de etapas para poder llegar a elaborar un modelo de red flags que dé respuesta acerca de licitaciones que pudieran ser objeto de revisión, debido a su valor en un determinado indicador de riesgo.

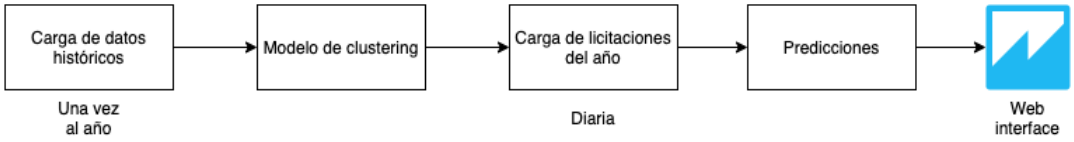
Lo primero que se abordó fue la definición de indicadores que permitan establecer si una licitación es riesgosa, para ello se revisó una serie de documentos internacionales y nacionales[1], de los cuales se desprenden las definiciones descritas en el capítulo "Indicadores de riesgo por Etapa".

Posteriormente en base a estas definiciones, se extrajeron los datos desde la base de datos histórica de ChileCompra (datos desde 2014 a 2019), con la finalidad de generar un aprendizaje a partir de ellos.

El mencionado aprendizaje se tradujo en la elaboración de un modelo de clustering no supervisado, el cual aplica por separado para cada una de las cuatro etapas del proceso de licitación, con esto se generaron grupos que están descritos en el capítulo "Clustering".

A continuación, se extrajeron los datos de 2020 a los cuales se les aplicó un pronóstico de pertenencia a un grupo de los ya definidos, utilizando el modelo creado, para finalmente mostrar los resultados a través de una interfaz web. Este proceso se puede apreciar en la figura 2:

Figura 2: Etapas del modelo de red flags



[1] La bibliografía de la cual se extrajo cada uno de los indicadores se señala en siguiente capítulo, específicamente en la descripción individual.

4. Indicadores de riesgo por Etapa y Extracción

En esta sección se presentan las definiciones de cada uno de los indicadores de riesgo separados por etapa, además de la descripción se detalla la fuente y en un anexo al final del documento, el código en Python utilizados para extraer los datos históricos (2014 - 2019) desde la base de datos Aquiles de ChileCompra (MSSQLSERVER) y cargarlos a una base de datos propia (MySQL).

4.1. Elaboración

Los siguientes son los indicadores que fueron levantados desde la bibliografía y que aplican al proceso de elaboración de las licitaciones.

4.1.1. Requisito de boleta de garantía irrazonable

Fuente: Dozorro [3] [4]/TI [5]/Chile Transparente [6]

Descripción: El monto de la boleta de garantía exigida para la licitación excede un porcentaje razonable del monto del contrato. Este tipo de requisitos pueden actuar como barreras de entrada a empresas más pequeñas con acceso limitado al capital o al mercado financiero, por otro lado, un bajísimo porcentaje podría indicar lo contrario, pues cuando no hay requisitos que resguarden los montos a ser invertidos en una adquisición, también podría ser riesgoso.

Para este indicador se definió como porcentaje umbrales lo siguiente:

- Fiel cumplimiento: entre 0,00001% y 30%.
- Seriedad de oferta: entre 0,00001% y 5%.
- Otros: entre 0,00001% y 5%.

Extracción de datos históricos: Anexo 1 – 1.1 Requisito de boleta de garantía irrazonable

4.1.2. Posibilidad de renovación de contrato

Fuente: TI [5]

Descripción: Este indicador de bandera roja se basa en información sobre la renovabilidad de contratos y establece como riesgosa situaciones en que el número de posibles renovaciones puede considerarse alto (3 o más) (no aplica a Chile), o dar lugar a un contrato por un tiempo demasiado largo (más de 4 años).

En el caso del indicador adaptado por el Observatorio, se fijó como riesgoso aquellas licitaciones que son renovables.

Extracción de datos históricos: Anexo 1 – 1.2 Posibilidad de renovación de contrato

4.1.3. Requisitos de experiencia

Fuente: TI [5]/Chile Transparente [6]

Descripción: El criterio experiencia no es garantía de la calidad de un bien o servicio y puede servir como una barrera de entrada para las empresas que buscan expandir sus oportunidades participando del abastecimiento del Estado.

En el caso de la adaptación de este indicador, el Observatorio constató que la cantidad de años de experiencia no es un campo que se individualice por licitación, sin embargo, está como parte de las descripciones de los criterios de adjudicación, a partir de esto, se estableció una búsqueda de palabras claves y se fijó como criterio que, si se identifican solicitudes de experiencia mayores a 5 años, se consideran riesgosos.

Extracción de datos históricos: Anexo 1 – 1.3 Requisitos de experiencia

4.1.4. Monto cercano a umbral de control

Fuente: Dozorro [3] [4]

Descripción: Este indicador se activa o es positivo cuando el monto de la licitación es cercano a su límite respectivo de control por parte de una entidad de fiscalización superior (EFS).

En el caso de la adaptación de este indicador a la realidad de Chile, se analizó la normativa respectiva, estableciendo que el control de este tipo de procesos lo realiza la Contraloría General de la República, mediante la Toma de Razón. Previo a 2019, todas las licitaciones menores a 5.000 UTM estaban exentas, de este trámite, por lo tanto, esa es una de las condiciones aplicadas. Desde 2019 en adelante, hay tres límites según la ubicación geográfica de la Contraloría Regional que realice el control cuando se trata de la Metropolitana entonces el límite es 15.000 UTM, Biobío es 10.000 UTM y para el resto es 8.000 UTM. Sin embargo, como no es posible saber que contraloría realiza el control de estos procesos a través de la plataforma de mercado público, ya que no existe esta información dentro del sistema.

Extracción de datos históricos: Anexo 1 – 1.4 Monto cercano al umbral de control

4.2. Licitación

4.2.1. Ofertas aceptadas después de la fecha límite

Fuente: PwC [8]

Descripción: Este indicador se activa cuando se presentan ofertas a una licitación después de la fecha de cierre para entrega de ofertas.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, por lo que se aplicó la descripción de manera literal.

Extracción de datos históricos: Anexo 1 – 2.1 Ofertas aceptadas después de la fecha límite

4.2.2. Todas las ofertas son altas

Fuente: IACRC [9]/PwC [8]

Descripción: Este indicador se activa cuando se presentan todas las ofertas que se presentan a una licitación están por sobre el costo estimado.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, por lo que se aplicó la descripción de manera literal.

Extracción de datos históricos: Anexo 1 – 2.2 Todas las ofertas son altas

4.2.3. Todas las ofertas son bajas

Fuente: IACRC [9]/PwC [8]

Descripción: Este indicador se activa cuando todas las ofertas que se presentan a una licitación están muy por debajo del costo estimado.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, pero se definió un límite arbitrario para considerarlas “muy por debajo”, estableciéndose en “todas aquellas que estén bajo el 20% del monto estimado”.

Extracción de datos históricos: Anexo 1 – 2.3 Todas las ofertas son bajas

4.2.4. Respuesta oportuna a reclamos

Fuente: Chiletransparente [6]

Descripción: El organismo comprador responde de una manera concisa y oportuna a los reclamos de los proveedores, considerando como oportuna 14 días.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, por lo que se aplica de manera literal la descripción.

Extracción de datos históricos: Anexo 1 – 2.4 Respuesta oportuna a reclamos

4.2.5. Cierre de preguntas en fin de semana

Fuente: ChileCompra [10]

Descripción: Variable que indica si el cierre del periodo de realización de preguntas se da un fin de semana o feriado.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, por lo que se aplica de manera literal la descripción.

Extracción de datos históricos: Anexo 1 – 2.5 Cierre de preguntas en fin de semana

4.2.6. Incumple plazo mínimo de publicación

Fuente: PwC [8]/ChileCompra [10]

Descripción: La licitación no cumple con un estándar mínimo de días de publicación dado el tipo de licitación.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, por lo que se aplica de manera literal la descripción.

Extracción de datos históricos: Anexo 1 – 2.6 Incumple plazo mínimo de publicación

4.2.7. Cantidad de ofertas

Fuente: PwC [8]/EU [11]/OCP [12]/TI [5]/OECD [7]

Descripción: Número de oferentes es significativamente menor que el promedio para un artículo similar o entidad contratante.

En el caso de la adaptación de este indicador a la realidad de Chile, todos los campos existen, sin embargo, debido a la enorme cantidad de productos que puede tener una licitación, se decidió abordarlo desde la perspectiva de la entidad contratante. Una vez establecido eso, se aplicó de manera literal.

Extracción de datos históricos: Anexo 1 – 2.7 Cantidad de ofertas.

4.2.8. Porcentaje reclamos por irregularidad sin respuesta

Fuente: ChileCompra [10]

Descripción: Porcentaje de reclamos por presunta irregularidad del proceso que quedan sin respuesta por parte del comprador, respecto del total de reclamos por presunta irregularidad presentados en el proceso.

Para la aplicación de este indicador, todos los campos existen, sin embargo, se debe definir un valor para establecer que procesos se marcan como riesgosos, en este caso como valor inicial se estableció que aquellos procesos licitatorios con un % menor al 90, se consideran riesgosos.

Extracción de datos históricos: Anexo 1 – 2.8 Porcentaje reclamos por irregularidad sin respuesta.

4.2.9. Ratio días de publicación preguntas

Fuente: ChileCompra [10]

Descripción: Ratio que representa la relación entre la cantidad de días para la realización de preguntas, respecto al total de días de publicación de la licitación.

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal, sin embargo, se debe definir un valor para establecer que procesos se marcan como riesgosos, en este caso como valor inicial se estableció que aquellos procesos licitatorios con un % menor al 30, se consideran riesgosos.

Extracción de datos históricos: Anexo 1 – 2.9 Ratio días de publicación preguntas.

4.2.10. Ratio reclamos irregularidad de ofertas

Fuente: ChileCompra [10]

Descripción: Ratio que representa la relación entre los reclamos ingresados por presunta irregularidad respecto al total de ofertas recepcionadas.

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal, sin embargo, se debe definir un valor para establecer que procesos se marcan como

riesgosos, en este caso como valor inicial se estableció que aquellos procesos licitatorios con una ratio mayor al 0.1, se consideran riesgosos.

Extracción de datos históricos: Anexo 1 – 2.10 Ratio reclamos irregularidad de ofertas.

4.2.11. Ratio reclamos pago

Fuente: ChileCompra [10]

Descripción: Ratio que representa la relación entre los reclamos ingresados por problemas de pago, respecto al total de ofertas recepcionadas.

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal, sin embargo, se debe definir un valor para establecer que procesos se marcan como riesgosos, en este caso como valor inicial se estableció que aquellos procesos licitatorios con un ratio mayor al 0.05, se consideran riesgosos.

Extracción de datos históricos: Anexo 1 – 2.11 Ratio reclamos pago.

4.2.12. Oferente único

Fuente: OCP [12]

Descripción: Licitaciones en las cuales se presenta un único oferente.

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal.

Extracción de datos históricos: Anexo 1 – 2.12 Oferente único

4.3. Evaluación

4.3.1. Largo periodo de decisión

Fuente: TI [5] /OECD [7]

Descripción: Licitaciones en las cuales el periodo de evaluación de las ofertas es demasiado largo (superior a 90 días)

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal.

Extracción de datos históricos: Anexo 1 – 3.1 Largo periodo de decisión

4.3.2. Plazo de decisión corto

Fuente: OCP [12]

Descripción: El plazo que se destina a la decisión de evaluación de ofertas es muy corto.

Para la aplicación de este indicador, todos los campos existen, por lo que se aplica de manera literal, y se establece como criterio “corto” si el plazo es menor o igual a diez días.

Extracción de datos históricos: Anexo 1 – 3.3 Plazo de decisión corto

4.3.3. Datos de proveedor faltantes

Fuente: OCP [12]

Descripción: los datos de contacto del proveedor están ausentes.

Para la aplicación de este indicador, se estableció como datos de contacto de primer nivel, la dirección y el teléfono, si alguno de estos no se registra o no tiene sentido (menos de 4 caracteres) entonces se considera riesgoso el proceso.

Extracción de datos históricos: Anexo 1 – 3.4 Datos de proveedor faltantes

4.3.4. Tiempo entre cierre y adjudicación estimado acotado

Fuente: ChileCompra [10]

Descripción: La cantidad de días entre el cierre y adjudicación es acotado en función del tipo de licitación, donde se compara contra el promedio por tipo.

Para la aplicación de este indicador, se encuentran todos los campos, por lo que se hizo literal, dejando el umbral del ratio en 0.5.

Extracción de datos históricos: Anexo 1 – 3.5 Tiempo entre cierre y adjudicación estimado acotado

4.3.5. Ofertas similares

Fuente: Dozorro [3] [4]/OCP [12]/OECD [7]

Descripción: Cualquier conexión entre licitantes que pueda socavar la competencia efectiva (por ejemplo, las propuestas de diferentes participantes de la misma licitación contienen el mismo nombre del punto de contacto, correo electrónico o número de teléfono).

Para la aplicación de este indicador, se estableció como similar en una primera instancia el monto ofertado, por lo tanto, si existen ofertas con idéntico monto se considera el proceso licitatorio como riesgoso.

Extracción de datos históricos: Anexo 1 – 3.6 Ofertas similares

4.4. Adjudicación

4.4.1. Extensión del contrato

Fuente: TI [5]

Descripción: Este indicador de bandera roja identifica un riesgo si el contratante anuncia la intención de celebrar un contrato por un período definido de más de cuatro años.

Dado que todos los campos que se necesitan para la aplicación se encuentran en el sistema, la definición se aplica de manera literal.

Extracción de datos históricos: Anexo 1 – 4.1 Extensión del contrato

4.4.2. Oferente novato gana

Fuente: TI [5]

Descripción: Este indicador se activa cuando el oferente que se adjudica el contrato no había adjudicado licitaciones previamente.

Dado que todos los campos que se necesitan para la aplicación se encuentran en el sistema, la definición se aplica de manera literal.

Extracción de datos históricos: Anexo 1 – 4.2 Oferente novato gana

4.4.3. Presencia de reclamos

Fuente: IACRC [9]/PwC [8]/OCP [12]

Descripción: Este indicador se activa si la licitación presenta reclamos.

Dado que todos los campos que se necesitan para la aplicación se encuentran en el sistema, la definición se aplica de manera literal.

Extracción de datos históricos: Anexo 1 – 4.3 Presencia de reclamos

4.4.4. Retraso entre la adjudicación y la firma del contracto

Fuente: OCP [12]/Dozorro [3] [4]

Descripción: El tiempo entre la adjudicación del contrato y la fecha real de firma del contrato excede un umbral razonable (debe ser inferior a 3 meses según el Banco Mundial).

Dado que la fecha de firma del contrato no se encuentra disponible en la base de datos, se tomó como variable proxy, la fecha de envío de la orden de compra.

Extracción de datos históricos: Anexo 1 – 4.4 Retraso entre la adjudicación y la firma del contracto

4.4.5. Diferencia entre monto final y estimado

Fuente: EU [11]/OCP [12]/TI [5]

Descripción: La relación entre el valor final total y el valor estimado no está dentro de un rango razonable (30% según OCP).

Ambos campos se encuentran en la base de datos, por lo que, la definición se aplica de manera literal. Umbrales fijados entre 0,7 y 1,3

Extracción de datos históricos: Anexo 1 – 4.5 Diferencia entre monto final y estimado

4.4.6. Falta de cláusulas de penalización

Fuente: OECD [7]

Descripción: Este indicador se activa frente a la falta de cláusulas de penalización por incumplimientos por parte del proveedor.

Este tipo de información no tiene un campo exclusivo para su almacenamiento, sin embargo, dentro del campo de texto libre se registran las cláusulas de las bases de licitación, por lo tanto, si una búsqueda a través de palabras clave asociadas a la penalización no arroja resultados, entonces el proceso se marca como riesgoso.

Extracción de datos históricos: Anexo 1 – 4.6 Falta de cláusulas de penalización

4.4.7. Falta de orden de compra

Fuente: TI [5]

Descripción: Este indicador se activa frente a la falta de generación de una orden de compra pese a que ha sido adjudicado el proceso.

Dado que todos los campos que generan este indicador se encuentran en la plataforma, la descripción se aplica de forma literal.

Extracción de datos históricos: Anexo 1 – 4.7 Falta de orden de compra

4.4.8. Monto adjudicado sobre el promedio

Fuente: IACRC [9]/OCP [12]/TI [5]

Descripción: Este indicador se activa si el monto adjudicado en el proceso licitatorio es superior al promedio de adjudicación histórico de la entidad licitante.

Dado que todos los campos que generan este indicador se encuentran en la plataforma, la descripción se aplica de forma literal.

Extracción de datos históricos: Anexo 1 – 4.8 Monto adjudicado sobre el promedio

4.4.9. Monto adjudicado muy cercano al monto estimado

Fuente: IACRC [9]

Descripción: Este indicador se activa cuando el monto adjudicado de la oferta ganadora es muy cercano al monto estimado.

Dado que todos los campos que generan este indicador se encuentran en la plataforma, la descripción se aplica de forma literal, y se considera muy cercano si el monto está dentro del intervalos del 3%, ya sea sobre o bajo el monto estimado.

Extracción de datos históricos: Anexo 1 – 4.9 Monto adjudicado muy cercano al monto estimado

4.4.10. Código proveedor ausente

Fuente: EU [11]

Descripción: Este indicador se activa cuando no es posible identificar el proveedor mediante su código de registro.

Dado que todos los campos que generan este indicador se encuentran en la plataforma, la descripción se aplica de forma literal.

Extracción de datos históricos: Anexo 1 – 4.10 Código proveedor ausente

4.4.11. Licitación desierta sin justificación

Fuente: TI [5]

Descripción: Este indicador se activa cuando el proceso se declara desierto y no existe registro alguno respecto de las razones para declararlo así.

Dado que todos los campos que generan este indicador se encuentran en la plataforma, la descripción se aplica de forma literal.

Extracción de datos históricos: Anexo 1 – 4.11 Licitación desierta sin justificación

4.4.12. Licitación cancelada por reclamos

Fuente: Dozorro [3] [4]

Descripción: Este indicador se activa cuando el proceso se cancela debido a que alguno de los reclamos presentados ameritaba su cancelación.

Para la aplicación de este indicador se presenta la dificultad de vincular el contenido de los reclamos con el hecho de que el proceso fue cancelada, por ello usamos como variable proxy que se presenten ambas condiciones, es decir, que se cancele el proceso y que haya reclamos.

Extracción de datos históricos: Anexo 1 – 4.12 Licitación cancelada por reclamos

5. Cálculos de agregaciones

Esta sección esta dedicada a la explicación de las agregaciones de los indicadores a nivel de cada licitación.

La fórmula para obtener el porcentaje de riesgo de cada categoría se calcula de la siguiente manera:

% riesgo categoría = (Cantidad de indicadores de la categoría con valor igual a 1)/(Cantidad de indicadores de la categoría total no nulos) * 100

Para el valor de riesgo agregado, la fómula es la misma pero sobre el total de indicadores.

6. Anexos

Anexo I: Códigos para extracción de datos históricos

1. Elaboración

1.1. Requisito de boleta de garantía irrazonable

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
AND YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# primero la seriedad de oferta
sqlSO = '''SELECT a.rbhCode,
CASE b.rgaAmountType
WHEN '1' THEN b.rgaAmount
WHEN '0' THEN (Case when a.rbhEstimatedAwardAmount=0 then null else
b.rgaAmount/a.rbhEstimatedAwardAmount*100 end)
ELSE 0
END as porctSerOfer
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBGuarantee b ON a.rbhCode = b.rgaRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
AND YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
AND b.rgaGuaranteeType = 1
AND b.rgaIsBasis = 1
ORDER by b.rgaAmount '''

# luego la de fiel cumplimiento
sqlFC = '''SELECT a.rbhCode,
CASE b.rgaAmountType
WHEN '1' THEN b.rgaAmount
WHEN '0' THEN (Case when a.rbhEstimatedAwardAmount=0 then null else
b.rgaAmount/a.rbhEstimatedAwardAmount*100 end)
```

```
ELSE 0
END as porctFC
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBGuarantee b ON a.rbhCode = b.rgaRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
AND YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
AND b.rgaGuaranteeType = 2
AND b.rgaIsBasis = 1
ORDER by b.rgaAmount'''

# otras garantías
sqlOtros = '''SELECT a.rbhCode,
CASE b.rgaAmountType
WHEN '1' THEN b.rgaAmount
WHEN '0' THEN (Case when a.rbhEstimatedAwardAmount=0 then null else
b.rgaAmount/a.rbhEstimatedAwardAmount*100 end)
ELSE 0
END as porctOtros
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBGuarantee b ON a.rbhCode = b.rgaRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
AND YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
AND b.rgaGuaranteeType not in (1,2)
AND b.rgaIsBasis = 1
ORDER by b.rgaAmount'''

licUniverso = pd.read_sql(sqlLic,cnxn)
garSO = pd.read_sql(sqlSO,cnxn)
garFC = pd.read_sql(sqlFC,cnxn)
garOtros = pd.read_sql(sqlOtros,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
garantias = licUniverso.merge(garSO, how='left')
garantias = garantias.merge(garFC, how='left')
garantias = garantias.merge(garOtros, how='left')

# Calculamos el redflag y dejamos el dataframe en garantias2
garantias2 = garantias.copy()

# condiciones para definir un redflag, finalmente se definió un límite de
conditions = [
    (pd.isnull(garantias2['porctSerOfer']) & pd.isnull(garantias2['porctFC']) &
pd.isnull(garantias2['porctOtros'])),
    ((garantias2['porctSerOfer']>5) | (garantias2['porctFC']>30) |
(garantias2['porctOtros']>10)) & (pd.notna(garantias2['porctSerOfer']) &
pd.notna(garantias2['porctFC']) & pd.notna(garantias2['porctOtros'])),
    ((garantias2['porctSerOfer']<0.00001) | (garantias2['porctFC']<0.00001) |
(garantias2['porctOtros']<0.00001)) & (pd.notna(garantias2['porctSerOfer']) &
pd.notna(garantias2['porctFC']) & pd.notna(garantias2['porctOtros']))]
choices = [1,1,1]

# aplicamos el filtro mediante un select
garantias2['requisitoGarantía'] = np.select(conditions, choices, default=0)

# botamos las columnas innecesarias
garantias2 = garantias2.drop(['porctSerOfer', 'porctFC', 'porctOtros'], axis=1)
```

```
garantias2 = garantias2.drop_duplicates()
garantias2 = garantias2.groupby(['rbhCode'], as_index = False).sum()
garantias2['requisitoGarantía'] = garantias2['requisitoGarantía'].apply(lambda x: 1 if x > 0
else 0)
garantías2 = licUniverso.merge(garantias2, how = 'left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
garantias2.to_sql(con=engine, name='requisitoGarantía', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(garantias2.index)) + " licitaciones, en "+
tFormateado)
```

1.2. Posibilidad de renovación de contrato

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# query para obtener indicador de si el contrato es renovable
sqlRenova = '''SELECT a.rbhCode, CASE WHEN a.rbhOptionContratoRenovable is null THEN 0 ELSE
a.rbhOptionContratoRenovable END as renovacionContrato
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic,cnxn)
renovaCont = pd.read_sql(sqlRenova,cnxn)
```

```
# Cerramos la conexión
cnxn.close()

#unimos los dataframes
renCont = licUniverso.merge(renovaCont, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
renCont.to_sql(con=engine, name='renovacionContrato', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(renCont.index)) + " licitaciones, en "+ tFormateado)
```

1.3. Requisitos de experiencia

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# query para obtener indicador de si el contrato es renovable
sqlExp = '''SELECT  a.rbhCode, case WHEN a.rbhCode IN (SELECT  a.rbhCode
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBAwardCriteria b ON a.rbhCode = b.rbaRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
and (b.rbaIsChecked = 1)
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
and LOWER(b.rbaName) like '%experiencia%'
and (LOWER(b.rbaComment) LIKE '%20 años%'
or LOWER(b.rbaComment) LIKE '%6 años%'
or LOWER(b.rbaComment) LIKE '%7 años%'
```

```
or LOWER(b.rbaComment) LIKE '%8 años%'
or LOWER(b.rbaComment) LIKE '%9 años%'
or LOWER(b.rbaComment) LIKE '%10 años%')) THEN 1 ELSE 0 END as requisitoExperiencia
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic,cnxn)
experiencia = pd.read_sql(sqlExp,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
expExcesiva = licUniverso.merge(experiencia, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
expExcesiva.to_sql(con=engine, name='requisitoExperiencia', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(expExcesiva)) + " licitaciones, en "+ tFormateado)
```

1.4. Monto cercano al umbral de control

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce
import mysql.connector
import urllib.request
import json

# Inicio del "cronómetro"
tInicio = time.time()

# Creamos los dataframes vacios para almacenar los valores de UTM y Fechas
dolarDiarioDF = pd.DataFrame(columns=['Fecha', 'Dolar'])
CLFDiarioDF = pd.DataFrame(columns=['Fecha', 'CLF'])
euroDiarioDF = pd.DataFrame(columns=['Fecha', 'Euro'])
utmDF = pd.DataFrame(columns=['Fecha', 'UTM'])

# Generamos el ciclo para recorrer los años y leer desde la API los datos de UTM
for i in range(2014, 2020):
    with urllib.request.urlopen("https://mindicador.cl/api/dolar/"+str(i)) as url:
        dolar = json.loads(url.read().decode())

    with urllib.request.urlopen("https://mindicador.cl/api/uf/"+str(i)) as url:
        uf = json.loads(url.read().decode())
```

```
    with urllib.request.urlopen("https://mindicador.cl/api/euro/"+str(i)) as url:
        euro = json.loads(url.read().decode())

    with urllib.request.urlopen("https://mindicador.cl/api/utm/"+str(i)) as url:
        utm = json.loads(url.read().decode())

    # Generamos los ciclos en que se añaden los dataframes los valores obtenidos desde la API
    for x in range(len(dolar['serie'])):
        dolarDiarioDF = dolarDiarioDF.append({'Fecha' : dolar['serie'][x]['fecha'][0:7] ,
'Dolar' : dolar['serie'][x]['valor']} , ignore_index=True)

    for x in range(len(uf['serie'])):
        CLFDiarioDF = CLFDiarioDF.append({'Fecha' : uf['serie'][x]['fecha'][0:7] , 'CLF' :
uf['serie'][x]['valor']} , ignore_index=True)

    for x in range(len(euro['serie'])):
        euroDiarioDF = euroDiarioDF.append({'Fecha' : euro['serie'][x]['fecha'][0:7] , 'Euro' :
euro['serie'][x]['valor']} , ignore_index=True)

    for x in range(len(utm['serie'])):
        utmDF = utmDF.append({'Fecha' : utm['serie'][x]['fecha'][0:7] , 'UTM' :
utm['serie'][x]['valor']} , ignore_index=True)

# Agrupamos por promedio para efectos prácticos
dolarDF = dolarDiarioDF.groupby(['Fecha'])[['Dolar']].mean()
CLFDF = CLFDiarioDF.groupby(['Fecha'])[['CLF']].mean()
euroDF = euroDiarioDF.groupby(['Fecha'])[['Euro']].mean()

# Unimos los dataframes y lo guardamos en uno nuevo
monedas = reduce(lambda x,y: pd.merge(x,y, on='Fecha', how='left'), [dolarDF, CLFDF, euroDF,
utmDF])

# Cargamos estos valores en la tabla UTM de MySQL
# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
monedas.to_sql(con=engine, name='Monedas', if_exists='replace', index=False)


# Ahora trabajamos sobre los datos de las licitaciones para ver si el monto estimado está cerca
de los umbrales de control (5000 UTM)
# Apertura la conexión a Conexiones
cnxn = pyodbc.connect('string de conexion')
mycnxn = mysql.connector.connect(host="192.168.2.2", user="server",  passwd="server")

# Envío de las querys y lectura de los resultados
# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# query para obtener indicador de cercanía a umbral de control
sqltempLimites = '''SELECT a.rbhCode, a.rbhEstimatedAmount, a.rbhCurrency,
CAST(YEAR(c.rbdOpeningDate) as varchar) + '-' + CASE WHEN MONTH(c.rbdOpeningDate)<10 THEN '0' +
cast(MONTH(c.rbdOpeningDate) as varchar) ELSE cast(MONTH(c.rbdOpeningDate) as varchar) END as
Fecha , CASE WHEN YEAR(c.rbdOpeningDate) > 2018 THEN 0 ELSE 1 END as pre2019
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
```

```
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

sqlMonedas = '''
SELECT *
FROM redflags.Monedas;
'''

licUniverso = pd.read_sql(sqlLic,cnxn)
tempLimites = pd.read_sql(sqltempLimites,cnxn)
monedasTemp = pd.read_sql(sqlMonedas,mycnxn)


#unimos los dataframes
tempLimites = tempLimites.merge(monedasTemp, how='left')

# Cerramos la conexión
cnxn.close()
mycnxn.close()

#unimos los dataframes
tempLimites = tempLimites.merge(monedasTemp, how='left')

#calculamos los valores en UTM
tempLimites['valorEnUTM'] = 0
tempLimites.loc[tempLimites['rbhCurrency'] == 'UTM', 'valorEnUTM'] =
round(tempLimites['rbhEstimatedAmount']).fillna(0).astype(int)
tempLimites.loc[tempLimites['rbhCurrency'] == 'CLP', 'valorEnUTM'] =
round(tempLimites['rbhEstimatedAmount'] / tempLimites['UTM']).fillna(0).astype(int)
tempLimites.loc[tempLimites['rbhCurrency'] == 'CLF', 'valorEnUTM'] =
round(tempLimites['rbhEstimatedAmount'] * tempLimites['CLF'] /
tempLimites['UTM']).fillna(0).astype(int)
tempLimites.loc[tempLimites['rbhCurrency'] == 'USD', 'valorEnUTM'] =
round(tempLimites['rbhEstimatedAmount'] * tempLimites['Dolar'] /
tempLimites['UTM']).fillna(0).astype(int)
tempLimites.loc[tempLimites['rbhCurrency'] == 'EUR', 'valorEnUTM'] =
round(tempLimites['rbhEstimatedAmount'] * tempLimites['Euro'] /
tempLimites['UTM']).fillna(0).astype(int)

# Generamos las condiciones para establecer el valor del indicador
conditions = [
    (tempLimites['pre2019'] == 1) & (tempLimites['valorEnUTM'] > 4900) &
(tempLimites['valorEnUTM'] < 5100),
    (tempLimites['pre2019'] == 0) & (tempLimites['valorEnUTM'] > 14900) &
(tempLimites['valorEnUTM'] < 15100),
    (tempLimites['pre2019'] == 0) & (tempLimites['valorEnUTM'] > 9900) &
(tempLimites['valorEnUTM'] < 10100),
    (tempLimites['pre2019'] == 0) & (tempLimites['valorEnUTM'] > 7900) &
(tempLimites['valorEnUTM'] < 8100)]
choices = [1, 1, 1, 1]
tempLimites['montoCercanoUmbral'] = np.select(conditions, choices, default=0)

# botamos las columnas innecesarias y fusión con licitaciones totales
tempLimites = tempLimites.drop(['rbhEstimatedAmount', 'rbhCurrency', 'Fecha', 'pre2019',
'Dolar', 'CLF', 'Euro', 'UTM', 'valorEnUTM'], axis=1)
limites = licUniverso.merge(tempLimites, how='left')

# Cargamos estos valores en la tabla indValorrUTM
# Carga de datos en MySQL
engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
limites.to_sql(con=engine, name='montoCercanoUmbral', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))
```

```
# Resultados de la primera extracción
# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(limites)) + " licitaciones, en "+ tFormateado)
```

2. Licitación

2.1. Ofertas aceptadas después de la fecha límite

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con indicador de si las fechas son incumplidas
sqlFechasInc = '''SELECT a.rbhCode, SUM(case when c.rbdTechnicalBidReception <
b.bidTechnicalIssueDate then 1 else 0 end) as flag
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP by a.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
fechasInc = pd.read_sql(sqlFechasInc, cnxn)


# Cerramos la conexión
cnxn.close()

# generamos el indicador
fechasInc['ofertasFueraPlazo'] = 0
fechasInc.loc[fechasInc['flag'] > 0, 'ofertasFueraPlazo'] = 1

# botamos las columnas innecesarias
fechasInc = fechasInc.drop(['flag'], axis=1)
```

```
#unimos los dataframes
fechIncumplidas = licUniverso.merge(fechasInc, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
fechIncumplidas.to_sql(con=engine, name='ofertasFueraPlazo', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(fechIncumplidas)) + " licitaciones, en "+
tFormateado)
```

2.2. Todas las ofertas son altas

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con contador de  las ofertas que son más altas que el monto
disponible
sqlOfertasAltas = '''SELECT t.rbhCode, SUM(t.cantOfer) as CantidadOfertas,  sum(case when t.porc
> 1.0 then 1 else 0 end) as OfertasAltas
FROM
(
SELECT a.rbhCode ,b.bidOrganization, count(b.bidID) as cantOfer,
case a.rbhEstimatedAmount
WHEN 0
then sum(c.bitUnitNetPrice * c.bitQuantity) else sum(c.bitUnitNetPrice * c.bitQuantity) /
a.rbhEstimatedAmount end as porc
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcBIDItem c ON b.bidID = c.bitBID
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d on a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
```

```
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
and c.bitUnitNetPrice > 1
AND a.rbhCurrency = c.bitCurrency
GROUP BY a.rbhCode, a.rbhEstimatedAmount ,b.bidOrganization
) as t
GROUP BY t.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
ofertasAltas = pd.read_sql(sqlOfertasAltas, cnxn)


# Cerramos la conexión
cnxn.close()

# generamos el indicador
ofertasAltas['todasOfertasAltas'] = 0
ofertasAltas.loc[ofertasAltas['CantidadOfertas'] == ofertasAltas['OfertasAltas'],
'todasOfertasAltas'] = 1

# botamos las columnas innecesarias
ofertasAltas = ofertasAltas.drop(['CantidadOfertas', 'OfertasAltas'], axis=1)

#unimos los dataframes
oferAltas = licUniverso.merge(ofertasAltas, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
oferAltas.to_sql(con=engine, name='todasOfertasAltas', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(oferAltas)) + " licitaciones, en "+ tFormateado)
```

2.3. Todas las ofertas son bajas

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
```

```
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con contador de  las ofertas que son menores que el 10% que
el monto disponible
sqlOfertasBajas = '''SELECT t.rbhCode, SUM(t.cantOfer) as CantidadOfertas,  sum(case when t.porc
< 0.2 then 1 else 0 end) as OfertasBajas
FROM
(
SELECT a.rbhCode ,b.bidOrganization, count(b.bidID) as cantOfer,
case a.rbhEstimatedAmount
WHEN 0
then sum(c.bitUnitNetPrice * c.bitQuantity) else sum(c.bitUnitNetPrice * c.bitQuantity) /
a.rbhEstimatedAmount end as porc
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcBIDItem c ON b.bidID = c.bitBID
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d on a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
and c.bitUnitNetPrice > 1
AND a.rbhCurrency = c.bitCurrency
GROUP BY a.rbhCode, a.rbhEstimatedAmount ,b.bidOrganization
) as t
GROUP BY t.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
ofertasBajas = pd.read_sql(sqlOfertasBajas, cnxn)


# Cerramos la conexión
cnxn.close()

# generamos el indicador
ofertasBajas['todasOfertasBajas'] = 0
ofertasBajas.loc[ofertasBajas['OfertasBajas'] == ofertasBajas['CantidadOfertas'],
'todasOfertasBajas'] = 1

# botamos las columnas innecesarias
ofertasBajas = ofertasBajas.drop(['CantidadOfertas', 'OfertasBajas'], axis=1)

#unimos los dataframes
oferBajas = licUniverso.merge(ofertasBajas, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
oferBajas.to_sql(con=engine, name='todasOfertasBajas', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(oferBajas)) + " licitaciones, en "+ tFormateado)
```

2.4. Respuesta oportuna a reclamos

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con indicador de si se considera plazo largo (14 días)
sqlplRespuesta = '''SELECT a.rbhCode, ((COUNT(DISTINCT (CASE WHEN DATEDIFF(HOUR,
b.FechaAsignacionReclamoOOPP, c.FechaEnvio) > 336 THEN 1 ELSE 0 END)))-1) AS plazoRespuestaLargo
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPReclamos.dbo.Reclamo b ON a.rbhExternalCode = b.NumLicOC
LEFT JOIN DCCPReclamos.dbo.GestionInternaOOPP c ON b.idReclamo = c.idReclamo
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d on a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
AND b.FechaAsignacionReclamoOOPP is not null
GROUP BY a.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
plRespuesta = pd.read_sql(sqlplRespuesta, cnxn)


# Cerramos la conexión
cnxn.close()

#unimos los dataframes
plalarRespuesta = licUniverso.merge(plRespuesta, how='left')
plalarRespuesta.loc[plalarRespuesta['plazoRespuestaLargo'].isnull(), 'plazoRespuestaLargo'] = 0
# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
plalarRespuesta.to_sql(con=engine, name='plazoRespuestaLargo', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))
```

```
# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(plalarRespuesta)) + " licitaciones, en "+
tFormateado)
```

2.5. Cierre preguntas en fin de semana

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones cuya fecha de cierre de preguntas corresponde a un fin de
semana o un día feriado
sqlFindeSemana = '''SELECT rbhCode
, rbhID IDLic
, d.rbdOpeningDate FechaLic
, rbhExternalCode CodLic
, b.rbdQuestionClose FechaCierrePreg
, DATENAME(dw,b.rbdQuestionClose) diadelasemana
, (CASE DATENAME(dw,b.rbdQuestionClose) WHEN 'Saturday' THEN 1 WHEN 'Sunday' THEN 1 ELSE 0 END)
FinDeSemana
, (CASE WHEN (CONVERT(varchar(2), DAY(b.rbdQuestionClose))+ '-' + CONVERT(varchar(2),
MONTH(b.rbdQuestionClose))+ '-' +RIGHT(CONVERT(varchar(4), YEAR(b.rbdQuestionClose)),2)) IN ('1-
1-14', '18-4-14', '19-4-14', '1-5-14', '21-5-14', '7-6-14', '29-6-14', '16-7-14', '15-8-14',
'20-8-14', '8-9-14', '18-9-14', '19-9-14', '20-9-14', '2-10-14', '12-10-14', '31-10-14', '1-11-
14', '8-12-14', '25-12-14', '1-1-15', '3-4-15', '4-4-15', '1-5-15', '21-5-15', '7-6-15', '29-6-
15', '16-7-15', '10-8-15', '15-8-15', '20-8-15', '18-9-15', '19-9-15', '12-10-15', '31-10-15',
'1-11-15', '8-12-15', '25-12-15', '1-1-16', '25-3-16', '26-3-16', '1-5-16', '21-5-16', '7-6-16',
'19-6-16', '27-6-16', '16-7-16', '10-8-16', '15-8-16', '20-8-16', '8-9-16', '18-9-16', '19-9-
16', '20-9-16', '10-10-16', '23-10-16', '31-10-16', '1-11-16', '8-12-16', '25-12-16', '1-1-17',
'2-1-17', '14-4-17', '15-4-17', '19-4-17', '1-5-17', '21-5-17', '7-6-17', '26-6-17', '2-7-17',
'16-7-17', '10-8-17', '15-8-17', '20-8-17', '18-9-17', '19-9-17', '20-9-17', '21-9-17', '2-10-
17', '9-10-17', '27-10-17', '1-11-17', '19-11-17', '8-12-17', '17-12-17', '25-12-17', '1-1-18',
'16-1-18', '17-1-18', '18-1-18', '30-3-18', '31-3-18', '1-5-18', '21-5-18', '7-6-18', '2-7-18',
'16-7-18', '15-8-18', '20-8-18', '17-9-18', '18-9-18', '19-9-18', '15-10-18', '1-11-18', '2-11-
18', '8-12-18', '25-12-18', '1-1-19', '19-4-19', '20-4-19', '1-5-19', '21-5-19', '7-6-19', '29-
6-19', '16-7-19', '15-8-19', '20-8-19', '18-9-19', '19-9-19', '20-9-19', '12-10-19', '31-10-19',
'1-11-19', '8-12-19', '25-12-19')
THEN 1 ELSE 0 END) Feriado
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhCode = b.rbdRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
```

```
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic, cnxn)
findeSemana = pd.read_sql(sqlFindeSemana, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes
cierreFinDe = licUniverso.merge(findeSemana, how='left')

# Aplicamos las condiciones para obtener el indicador
cierreFinDe.loc[cierreFinDe['FinDeSemana'].isnull(), 'FinDeSemana'] = 0
cierreFinDe.loc[cierreFinDe['Feriado'].isnull(), 'Feriado'] = 0
cierreFinDe['cierrePreguntasFinde'] = 0
cierreFinDe.loc[cierreFinDe['FinDeSemana'] == 1, 'cierrePreguntasFinde'] = 1
cierreFinDe.loc[cierreFinDe['Feriado'] == 1, 'cierrePreguntasFinde'] = 1

# Eliminamos los campos que no se utilizan
cierreFinDe = cierreFinDe.drop(['IDLic', 'FechaLic', 'CodLic', 'FechaCierrePreg',
'diadelasemana', 'FinDeSemana', 'Feriado'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
cierreFinDe.to_sql(con=engine, name='cierrePreguntasFinde', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(cierreFinDe)) + " licitaciones, en "+ tFormateado)
```

2.6. Incumple plazo mínimo de publicación

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
```

```
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones que cumplen con el mínimo de plazo de publicación
sqlPerMinPub = '''SELECT rbhCode, rbhID, b.rbdOpeningDate, rbhProcessSubType,
b.rbdTechnicalBidReception,
DATEDIFF(DAY, b.rbdOpeningDate, b.rbdTechnicalBidReception) AS PlazoPublicacion,
(CASE WHEN a.rbhProcessSubType = 1 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 5 THEN 1
         WHEN a.rbhProcessSubType = 2 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 10 THEN 1
         WHEN a.rbhProcessSubType = 3 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 20 THEN 1
         WHEN a.rbhProcessSubType = 24 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 20 THEN 1
         WHEN a.rbhProcessSubType = 25 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 30 THEN 1
         WHEN a.rbhProcessSubType = 23 AND DATEDIFF(DAY, b.rbdOpeningDate,
b.rbdTechnicalBidReception) < 10 THEN 1
         ELSE 0 END) AS fueraPlazoPublica
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhCode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) > 2013
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic, cnxn)
perMinPub = pd.read_sql(sqlPerMinPub, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes
perMinPublic = licUniverso.merge(perMinPub, how='left')

# Aplicamos las condiciones para obtener el indicador
perMinPublic.loc[perMinPublic['fueraPlazoPublica'].isnull(), 'fueraPlazoPublica'] = 0

# Eliminamos los campos que no se utilizan
perMinPublic = perMinPublic.drop(['rbhID', 'rbdOpeningDate', 'rbhProcessSubType',
'rbdTechnicalBidReception', 'PlazoPublicacion'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
perMinPublic.to_sql(con=engine, name='fueraPlazoPublica', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(perMinPublic)) + " licitaciones, en "+ tFormateado)
```

2.7. Incumple plazo mínimo de publicación

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de promedios de ofertas por institución
sqlPromOfertas = '''SELECT a.rbhOrganization, COUNT(DISTINCT b.bidID)/COUNT(DISTINCT a.rbhCode)
as promOfertas
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhOrganization'''

# Descarga de datos de cantidad de ofertas por licitación
sqlOferXLic = '''SELECT a.rbhCode, a.rbhOrganization, COUNT(DISTINCT b.bidID) AS BidCount
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode, a.rbhOrganization'''

licUniverso = pd.read_sql(sqlLic, cnxn)
promOfertas = pd.read_sql(sqlPromOfertas, cnxn)
oferXLic = pd.read_sql(sqlOferXLic, cnxn)

# Cerramos la conexión
cnxn.close()

# unimos los dataframes
tempPromOfer = licUniverso.merge(oferXLic, how='left')
tempPromOfer = tempPromOfer.merge(promOfertas, how='left')
```

```
# Aplicamos las condiciones para obtener el indicador
tempPromOfer['cantidadOfertas'] = 0
tempPromOfer.loc[tempPromOfer['BidCount'] < tempPromOfer['promOfertas'], 'cantidadOfertas'] = 1

# Eliminamos los campos que no se utilizan
promOfertas = tempPromOfer.drop(['rbhOrganization', 'BidCount', 'promOfertas'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
promOfertas.to_sql(con=engine, name='cantidadOfertas', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(promOfertas)) + " licitaciones, en "+ tFormateado)
```

2.8. Porcentaje reclamos por irregularidad sin respuesta

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de porcentajes de respuesta a los reclamos
sqlPorcRespRecla = '''SELECT a.rbhCode, CASE WHEN CONVERT(FLOAT,
COUNT(b.RespuestaOOPP))/CONVERT(FLOAT, COUNT(b.idReclamo))*100 < 90 THEN 1 ELSE 0 END as
porcReclamosIregularidad
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPReclamos.dbo.Reclamo b ON a.rbhExternalCode = b.NumLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND b.idMotivoReclamo != 1
AND b.idMotivoReclamo != 18
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
```

```
GROUP BY a.rbhCode, a.rbhExternalCode'''


licUniverso = pd.read_sql(sqlLic, cnxn)
porcRespRecla = pd.read_sql(sqlPorcRespRecla, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes
porcRespRec = licUniverso.merge(porcRespRecla, how='left')

# Aplicamos las condiciones para obtener el indicador
porcRespRec.loc[porcRespRec['porcReclamosIregularidad'].isnull(), 'porcReclamosIregularidad'] =
0

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
porcRespRec.to_sql(con=engine, name='porcReclamosIregularidad', if_exists='replace',
index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(porcRespRec)) + " licitaciones, en "+ tFormateado)
```

2.9. Ratio días de publicación preguntas

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de porcentajes de dias para preguntas versus el total publicado
sqlPorcPlazoRespuesta = '''SELECT rbhCode,
```

```
(Convert(Float,((DATEDIFF(day, b.rbdQuestionOpening,
b.rbdQuestionClose)))))/nullif((CONVERT(FLOAT,((DATEDIFF(day, b.rbdOpeningDate,
b.rbdTechnicalBidReception)))))),0) *100 as porc,
case WHEN (Convert(Float,((DATEDIFF(day, b.rbdQuestionOpening,
b.rbdQuestionClose)))))/nullif((CONVERT(FLOAT,((DATEDIFF(day, b.rbdOpeningDate,
b.rbdTechnicalBidReception)))))),0) *100 < 30 then 1 else 0 end as ratioDiasPreguntas
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhCode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) > 2013
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1 '''


licUniverso = pd.read_sql(sqlLic, cnxn)
porcPlazoRespuesta = pd.read_sql(sqlPorcPlazoRespuesta, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes
porcPlaResp = licUniverso.merge(porcPlazoRespuesta, how='left')

# Aplicamos las condiciones para obtener el indicador
porcPlaResp.loc[porcPlaResp['ratioDiasPreguntas'].isnull(), 'ratioDiasPreguntas'] = 0

# Eliminamos los campos que no se utilizan
porcPlaResp = porcPlaResp.drop(['porc'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
porcPlaResp.to_sql(con=engine, name='ratioDiasPreguntas', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(porcPlaResp)) + " licitaciones, en "+ tFormateado)
```

2.10. Ratio reclamos irregularidad de ofertas

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
```

```
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de cantidad de ofertas por licitación
sqlQOfertas = '''SELECT a.rbhCode, (CONVERT(FLOAT,(COUNT(b.bidID)))) AS 'TotalOfertas'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhcode'''

# Descarga de datos de cantidad de reclamos por irregularidad
sqlQReclamos = '''SELECT a.rbhCode, (CONVERT(FLOAT,(COUNT(DISTINCT c.idReclamo)))) AS
'ReclamosIrregularidad'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND c.idMotivoReclamo != 1
AND c.idMotivoReclamo != 18
AND a.rbhProcessType = 1
GROUP BY a.rbhcode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
QOfertas = pd.read_sql(sqlQOfertas, cnxn)
QReclamos = pd.read_sql(sqlQReclamos, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes que tienen los datos del indicador
tempRatRecl = QOfertas.merge(QReclamos, how='left')
tempRatRecl2 = licUniverso.merge(tempRatRecl, how='left')

# Reemplazamos los valores nulos por ceros
tempRatRecl2.loc[tempRatRecl2['ReclamosIrregularidad'].isnull(), 'ReclamosIrregularidad'] = 0
tempRatRecl2.loc[tempRatRecl2['TotalOfertas']== 0, 'TotalOfertas'] = 1

# Calculamos el indicador
tempRatRecl2['ratioReclamosIrregularidad'] =
tempRatRecl2['ReclamosIrregularidad']/tempRatRecl2['TotalOfertas']
tempRatRecl2.loc[tempRatRecl2['ratioReclamosIrregularidad'] < 0.1, 'ratioReclamosIrregularidad']
= 0
tempRatRecl2.loc[tempRatRecl2['ratioReclamosIrregularidad'] >= 0.1,
'ratioReclamosIrregularidad'] = 1


# Eliminamos los campos que no se utilizan
ratRecl = tempRatRecl2.drop(['TotalOfertas', 'ReclamosIrregularidad'], axis=1)
```

```
# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
ratRecl.to_sql(con=engine, name='ratioReclamosIrregularidad', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(ratRecl)) + " licitaciones, en "+ tFormateado)
```

2.11. Ratio reclamos irregularidad de ofertas

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de cantidad de ofertas por licitación
sqlQOfertas = '''SELECT a.rbhCode, (CONVERT(FLOAT,(COUNT(b.bidID)))) AS 'TotalOfertas'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhcode'''

# Descarga de datos de cantidad de reclamos por pagos
sqlQReclamos = '''SELECT a.rbhCode, (CONVERT(FLOAT,(COUNT(DISTINCT c.idReclamo)))) AS
'ReclamosPago'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
```

```
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND (c.idMotivoReclamo = 1 or c.idMotivoReclamo = 18)
AND a.rbhProcessType = 1
GROUP BY a.rbhcode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
QOfertas = pd.read_sql(sqlQOfertas, cnxn)
QReclamos = pd.read_sql(sqlQReclamos, cnxn)


# Cerramos la conexión
cnxn.close()

# unimos los dataframes que tienen los datos del indicador
tempRatRecl = QOfertas.merge(QReclamos, how='left')
tempRatRecl2 = licUniverso.merge(tempRatRecl, how='left')

# Reemplazamos los valores nulos por ceros
tempRatRecl2.loc[tempRatRecl2['ReclamosPago'].isnull(), 'ReclamosPago'] = 0
tempRatRecl2.loc[tempRatRecl2['TotalOfertas']== 0, 'TotalOfertas'] = 1

# Calculamos el indicador
tempRatRecl2['ratioReclamosPago'] = tempRatRecl2['ReclamosPago']/tempRatRecl2['TotalOfertas']
tempRatRecl2.loc[tempRatRecl2['ratioReclamosPago'] < 0.05, 'ratioReclamosPago'] = 0
tempRatRecl2.loc[tempRatRecl2['ratioReclamosPago'] >= 0.05, 'ratioReclamosPago'] = 1

# Eliminamos los campos que no se utilizan
ratRecl = tempRatRecl2.drop(['TotalOfertas', 'ReclamosPago'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
ratRecl.to_sql(con=engine, name='ratioReclamosPago', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(ratRecl)) + " licitaciones, en "+ tFormateado)
```

2.12. Oferente único

```
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
```

```
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de cantidad de ofertas por licitación
sqlQOfertas = '''SELECT a.rbhCode, (CONVERT(FLOAT,(COUNT(b.bidID)))) AS 'TotalOfertas'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhcode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
QOfertas = pd.read_sql(sqlQOfertas, cnxn)

# Cerramos la conexión
cnxn.close()

# unimos los dataframes que tienen los datos del indicador
tempQOfertas = licUniverso.merge(QOfertas, how='left')

# Reemplazamos los valores nulos por ceros
tempQOfertas['oferenteUnico'] = 0
tempQOfertas.loc[tempQOfertas['TotalOfertas'] < 2, 'oferenteUnico'] = 1

# Eliminamos los campos que no se utilizan
QOfertas = tempQOfertas.drop(['TotalOfertas'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
QOfertas.to_sql(con=engine, name='oferenteUnico', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(QOfertas)) + " licitaciones, en "+ tFormateado)
```

3. Licitación

3.1. Largo periodo de decisión

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
```

```
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con indicador de si los plazos de la decision de licitación
es muy larga
sqlPlazoDecLargo = '''SELECT a.rbhCode,
(CASE
WHEN (DATEDIFF(DAY, b.rbdTechnicalBidOpening, b.rbdAwardDate)) > 90
THEN 1
ELSE 0 END) AS plazoDecisionLargo
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhcode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) >= 2014
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic, cnxn)
plazoDecLargo = pd.read_sql(sqlPlazoDecLargo, cnxn)


# Cerramos la conexión
cnxn.close()

# regulariazamos el indicador
plazoDecLargo.loc[plazoDecLargo['plazoDecisionLargo'].isnull(), 'plazoDecisionLargo'] = 0

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
plazoDecLargo.to_sql(con=engine, name='plazoDecisionLargo', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(plazoDecLargo)) + " licitaciones, en "+ tFormateado)
```

3.2. Plazo de decisión corto

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con indicador de si los plazos de la decision de licitación
es muy corto
sqlPlazoDecCorto = '''SELECT a.rbhCode,
(CASE
WHEN (DATEDIFF(HOUR, b.rbdTechnicalBidOpening, b.rbdAwardDate)) <= 10
THEN 1
ELSE 0 END) AS plazoDecisionCorto
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhcode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) >= 2014
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic, cnxn)
plazoDecCorto = pd.read_sql(sqlPlazoDecCorto, cnxn)


# Cerramos la conexión
cnxn.close()

# regulariazamos el indicador
plazoDecCorto.loc[plazoDecCorto['plazoDecisionCorto'].isnull(), 'plazoDecisionCorto'] = 0

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
plazoDecCorto.to_sql(con=engine, name='plazoDecisionCorto', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))
```

```
# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(plazoDecCorto)) + " licitaciones, en "+ tFormateado)
```

3.3. Datos de proveedor faltante

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de licitaciones con indicador de si la dirección o el teléfono del proveedor
de la licitación es extraña o inexistente
sqlDirAnormal = '''SELECT a.rbhCode,
CASE
WHEN (sum(CASE WHEN c.eadAddress IS NULL THEN 1 WHEN LEN(c.eadAddress) < 5 THEN 1 WHEN
c.eadPhone IS NULL THEN 1 WHEN LEN(c.eadPhone) < 7 THEN 1 ELSE 0 END)) > 0 THEN 1 ELSE 0 END AS
direccionAnormal
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPPlatform.dbo.gblEnterpriseAddress c ON b.porSellerEnterprise = c.eadEnterprise
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d on a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
dirAnormal = pd.read_sql(sqlDirAnormal, cnxn)


# Cerramos la conexión
cnxn.close()

# regulariazamos el indicador
dirAnormal.loc[dirAnormal['direccionAnormal'].isnull(), 'direccionAnormal'] = 0

# Carga de datos en MySQL
from sqlalchemy import create_engine
```

```
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
dirAnormal.to_sql(con=engine, name='direccionAnormal', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(dirAnormal)) + " licitaciones, en "+ tFormateado)
```

3.4. Tiempo entre cierre y adjudicación estimado acotado

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de tiempo promedio de evaluación de ofertas por tip de licitación
sqlTiempoPromEvaluacionxTipo = '''SELECT rbhProcessSubType, AVG(DATEDIFF(DAY,
b.rbdTechnicalBidReception, b.rbdAwardDate)) AS PlazoPromedio
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhCode = b.rbdRFBCode
WHERE rbhProcessSubType IN (1, 2, 3, 23, 24, 25,  30)
AND YEAR(b.rbdOpeningDate) > 2013
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
GROUP BY rbhProcessSubType'''

sqlTiempoEvaluacion = '''SELECT rbhCode, rbhProcessSubType, DATEDIFF(DAY,
b.rbdTechnicalBidReception, b.rbdAwardDate) AS PlazoDecision
FROM DCCPProcurement.dbo.prcRFBHeader a LEFT JOIN DCCPProcurement.dbo.prcRFBDate b
ON a.rbhCode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) > 2013
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''
```

```
licUniverso = pd.read_sql(sqlLic, cnxn)
tiempoPromEvaluacionxTipo = pd.read_sql(sqlTiempoPromEvaluacionxTipo, cnxn)
tiempoEvaluacion = pd.read_sql(sqlTiempoEvaluacion, cnxn)


# Cerramos la conexión
cnxn.close()

# Unimos las consulta 2 y 3 para asociar los plazos promedio con los plazos de cada licitación
indTiempoEvaluacion = tiempoEvaluacion.merge(tiempoPromEvaluacionxTipo, how = 'left', left_on =
'rbhProcessSubType', right_on = 'rbhProcessSubType')

# calculamos cuanto representa porcentualmente el valor respecto del promedio
indTiempoEvaluacion['porcPromedio'] = indTiempoEvaluacion['PlazoDecision'] /
indTiempoEvaluacion['PlazoPromedio']

# Calculamos el indicador
indTiempoEvaluacion['plazoDecisionSobrePromedio'] = 0
indTiempoEvaluacion.loc[indTiempoEvaluacion['porcPromedio'] <= 0.5 ,
'plazoDecisionSobrePromedio'] = 1

# eliminamos los campos que no se utilizarán.
indTEval = indTiempoEvaluacion.drop(['rbhProcessSubType', 'PlazoDecision', 'PlazoPromedio',
'porcPromedio'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
indTEval.to_sql(con=engine, name='plazoDecisionSobrePromedio', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(indTEval)) + " licitaciones, en "+ tFormateado)
```

3.5. Ofertas similares

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
```

```
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Descarga de datos de tiempo promedio de evaluación de ofertas por tip de licitación
sqlOfertasSim = '''SELECT a.rbhCode, COUNT(b.bidID) AS TotalOfertas, COUNT(DISTINCT
c.bitUnitNetPrice) AS PreciosDistintos
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcBIDItem c ON b.bidID = c.bitBID
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
AND c.bitCurrency IS NOT NULL
AND b.bidIsAwarded IS NOT NULL
GROUP BY a.rbhCode'''

licUniverso = pd.read_sql(sqlLic, cnxn)
ofertasSimilares = pd.read_sql(sqlOfertasSim, cnxn)


# Cerramos la conexión
cnxn.close()


# Calculamos el indicador
ofertasSimilares['ofertasSimilares'] = 0
ofertasSimilares.loc[ofertasSimilares['PreciosDistintos'] < ofertasSimilares['TotalOfertas'],
'ofertasSimilares'] = 1

# Eliminamos los campos que no se utilizarán.
ofertasSim = ofertasSimilares.drop(['TotalOfertas', 'PreciosDistintos'], axis=1)

#cruzamos con el total
ofertasSim = licUniverso.merge(ofertasSim, how = 'left')
ofertasSim.loc[ofertasSim['ofertasSimilares'].isnull() , 'ofertasSimilares'] = 0

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
ofertasSim.to_sql(con=engine, name='ofertasSimilares', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(ofertasSim)) + " licitaciones, en "+ tFormateado)
```

4. Adjudicación

4.1. Extensión del contrato

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# tiempo en días por licitación
sqlTiempo = '''SELECT a.rbhCode
, (CASE a.rbhContractDuration
WHEN 5 THEN CONVERT(bigint ,a.rbhContractTime) * 365
WHEN 4 THEN CONVERT(bigint ,a.rbhContractTime) * 30
WHEN 3 THEN CONVERT(bigint ,a.rbhContractTime) * 7
WHEN 2 THEN CONVERT(bigint ,a.rbhContractTime) * 1
WHEN 1 THEN CONVERT(bigint ,a.rbhContractTime) * 24
WHEN 0 THEN CONVERT(bigint ,a.rbhContractTime) * 1
ELSE 0 END) dias
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic,cnxn)
durContrato = pd.read_sql(sqlTiempo,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
durCont = licUniverso.merge(durContrato, how='left')

# Calculamos el redflag y dejamos el dataframe en garantias2
durCont2 = durCont.copy()

# condiciones para definir un redflag, finalmente se definió un límite de
conditions = [
    (pd.isnull(durCont2['dias'])),
    (durCont2['dias']>=1460)]

choices = [0,1]
```

```
# aplicamos el filtro mediante un select
durCont2['plazoContratoLargo'] = np.select(conditions, choices, default=0)

# eliminamos las columnas innecesarias
durCont2 = durCont2.drop(['dias'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
durCont2.to_sql(con=engine, name='plazoContratoLargo', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(licUniverso)) + " licitaciones, en "+ tFormateado)
```

4.2. Oferente novato gana

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# tiempo en días por licitación
sqlFechaMinProv = '''SELECT MIN(a.bidTechnicalIssueDate) primerOferta, a.bidOrganization
FROM DCCPProcurement.dbo.prcBIDQuote a
GROUP BY a.bidOrganization'''

# tiempo en días por licitación
sqlLicOferFechaProv = '''SELECT a.rbhCode, a.rbhExternalCode , b.bidOrganization,
b.bidTechnicalIssueDate
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcBIDQuote b ON a.rbhCode = b.bidRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE bidOrganization is not NULL
AND b.bidIsAwarded > 0
AND YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
```

```
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic,cnxn)
fechaMinProv = pd.read_sql(sqlFechaMinProv,cnxn)
licOferFechaProv = pd.read_sql(sqlLicOferFechaProv,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
oferenteNovato = licOferFechaProv.merge(fechaMinProv, how='left', left_on = 'bidOrganization',
right_on = 'bidOrganization')

# cambiamos los nulos de fechas por fechas muy lejanas para que en la posterior comparación
aparezcan como oferentes nuevos
oferenteNovato.loc[oferenteNovato['primerOferta'].isnull(), 'primerOferta'] = '2099-01-01
00:00:00.000'

# Calculamos el indicador
oferenteNovato['oferenteNovato'] = 0
oferenteNovato.loc[oferenteNovato['bidTechnicalIssueDate'] == oferenteNovato['primerOferta'],
'oferenteNovato'] = 1

#Generamos los dataframes de nodos
oferNovato = oferenteNovato.groupby(['rbhCode'], as_index = False).sum()

# Normalizamos el indicador
oferNovato.loc[oferNovato['oferenteNovato'] > 0, 'oferenteNovato'] = 1

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
oferNovato.to_sql(con=engine, name='oferenteNovato', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(licUniverso)) + " licitaciones, en "+ tFormateado)
```

4.3. Presencia de reclamos

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlLic = '''select a.rbhCode
```

```
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# consulta para obtener si una licitación presenta reclamos
sqlPresentaReclamos = '''SELECT a.rbhCode, (CASE WHEN COUNT(DISTINCT c.idReclamo) > 0 THEN 1
ELSE 0 END) AS presenciaReclamos
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPReclamos.dbo.Reclamo c ON a.rbhExternalCode = c.NumLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode'''

licUniverso = pd.read_sql(sqlLic,cnxn)
presentaReclamos = pd.read_sql(sqlPresentaReclamos,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
presentaReclamos = licUniverso.merge(presentaReclamos, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
presentaReclamos.to_sql(con=engine, name='presenciaReclamos', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(licUniverso)) + " licitaciones, en "+ tFormateado)
```

4.4. Retraso entre la adjudicación y la firma del contrato

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
```

```
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# consulta para obtener diferencia entre plazos estimado y real de firma de contrato, se hace un
poxy con la adjudicación, ya que no tenemos fecha de firma.
sqlDiferenciaFechaFirmas = '''SELECT a.rbhCode, (CASE WHEN (DATEDIFF(DAY, b.rbdAwardDate,
b.rbdEstimatedContractSign)) >= 90 THEN 1 ELSE 0 END ) AS retrasoEnFirma
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate b ON a.rbhCode = b.rbdRFBCode
WHERE YEAR(b.rbdOpeningDate) > 2013
and YEAR(b.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

licUniverso = pd.read_sql(sqlLic,cnxn)
diferenciaFechaFirmas = pd.read_sql(sqlDiferenciaFechaFirmas,cnxn)

# Cerramos la conexión
cnxn.close()

#unimos los dataframes
diferenciaFechaFirmas = licUniverso.merge(diferenciaFechaFirmas, how='left')

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
diferenciaFechaFirmas.to_sql(con=engine, name='retrasoEnFirma', if_exists='replace',
index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(diferenciaFechaFirmas)) + " licitaciones, en "+
tFormateado)
```

4.5. Diferencia entre monto final y estimado

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados
```

```
# Listado de licitaciones
sqlLic = '''select a.rbhCode
from DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate c on a.rbhCode = c.rbdRFBCode
WHERE YEAR(c.rbdOpeningDate) > 2013
and YEAR(c.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

# Query para obtener el monto estimado, el monto total comprometido y el ratio
sqlDiferenciaMonto = '''SELECT rbhCode, SUM(rbhEstimatedAmount) AS MontoEstimado,
SUM(b.porTotalAmount) AS MontoAdjudicado,
(SUM(b.porTotalAmount)/NULLIF((SUM(rbhEstimatedAmount)), 0)) AS Desviacion
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY rbhCode'''

licUniverso = pd.read_sql(sqlLic,cnxn)
diferenciaMonto = pd.read_sql(sqlDiferenciaMonto,cnxn)


# Cerramos la conexión
cnxn.close()

# Calculamos el indicador
diferenciaMonto['diferenciaEntreMontos'] = 0
diferenciaMonto.loc[diferenciaMonto['Desviacion'] < 0.7 , 'diferenciaEntreMontos'] = 1
diferenciaMonto.loc[diferenciaMonto['Desviacion'] > 1.3 , 'diferenciaEntreMontos'] = 1

# eliminamos las columnas innecesarias
diferenciaMonto = diferenciaMonto.drop(['MontoEstimado', 'MontoAdjudicado', 'Desviacion'],
axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
diferenciaMonto.to_sql(con=engine, name='diferenciaEntreMontos', if_exists='replace',
index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(diferenciaMonto)) + " licitaciones, en "+
tFormateado)
```

4.6. Falta de cláusulas de penalización

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
```

```
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Query para indicador de presencia de multas, este se trata de una estimación ya que, se usan
palabras claves para saber si hay multas solicitadas en el formulario
sqlMultas = '''SELECT a.rbhCode,
MIN(CASE WHEN lower(rbcTitle) LIKE '%penal%' THEN 0
                WHEN lower(rbcTitle) LIKE '%sancion%' THEN 0
                WHEN lower(rbcTitle) LIKE '%sanción%' THEN 0
                WHEN lower(rbcTitle) LIKE '%multa%' THEN 0
                WHEN lower(rbcTitle) LIKE '%castigo%' THEN 0
                WHEN lower(rbcTitle) LIKE '%punicion%' THEN 0
                WHEN lower(rbcTitle) LIKE '%punición%' THEN 0
                WHEN lower(rbcTitle) LIKE '%correctivo%' THEN 0
                WHEN lower(rbcTitle) LIKE '%escarmiento%' THEN 0
                WHEN lower(rbcTitle) LIKE '%recargo%' THEN 0
                WHEN lower(rbcTitle) LIKE '%gravamen%' THEN 0
                WHEN lower(rbcTitle) LIKE '%amonestaci%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%penal%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%sancion%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%sanción%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%multa%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%castigo%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%punici%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%correctivo%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%escarmiento%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%imposici%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%recargo%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%gravamen%' THEN 0
                WHEN lower(b.rbcDescription) LIKE '%amonestaci%' THEN 0
ELSE 1 END) AS clausulaPenalizacion
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBClause b ON a.rbhCode = b.rbcRFBCode
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode'''

multas = pd.read_sql(sqlMultas,cnxn)


# Cerramos la conexión
cnxn.close()


# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
multas.to_sql(con=engine, name='clausulaPenalizacion', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
```

```
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(multas)) + " licitaciones, en "+ tFormateado)
```

4.7. Falta de cláusulas de penalización

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Query para indicador de presencia de multas, este se trata de una estimación ya que, se usan
palabras claves para saber si hay multas solicitadas en el formulario
sqlSinContratos = '''SELECT a.rbhCode, (CASE WHEN a.rbhDocumentStatus IN (8,9,10) AND
COUNT(b.porID) = 0 THEN 1 ELSE 0 END) AS sinOC
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode, a.rbhDocumentStatus'''

sinContratos = pd.read_sql(sqlSinContratos,cnxn)


# Cerramos la conexión
cnxn.close()


# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
sinContratos.to_sql(con=engine, name='sinOC', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(sinContratos)) + " licitaciones, en "+ tFormateado)
```

4.8. Monto adjudicado sobre el promedio

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Listado de licitaciones
sqlPromMontoProv = '''SELECT b.porBuyerOrganization , count(DISTINCT a.rbhCode) cantLic,
sum(b.porTotalAmount) sumMonto, sum(b.porTotalAmount)/count(DISTINCT a.rbhCode) promMonto
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
and b.porBuyerOrganization is not NULL
and b.porBuyerOrganization != ' '
GROUP BY b.porBuyerOrganization'''

# Query para obtener el monto estimado, el monto total comprometido y el ratio
sqlMontoLic = '''SELECT a.rbhCode, a.rbhOrganization, sum(b.porTotalAmount) Monto
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhOrganization, a.rbhCode'''

promMontoProv = pd.read_sql(sqlPromMontoProv,cnxn)
montoLic = pd.read_sql(sqlMontoLic,cnxn)


# Cerramos la conexión
cnxn.close()

#unimos los dataframes
montoAlto = montoLic.merge(promMontoProv, how='left', left_on = 'rbhOrganization', right_on =
'porBuyerOrganization')

#Calculamos el indicador primero con la división de monto por licitación sobre el promedio y
luego si esta es mayor que 2, entonces se considera riesgosa.
montoAlto['div'] = montoAlto['Monto']/montoAlto['promMonto']
montoAlto['desviacionMontoPromedio'] = 0
montoAlto.loc[montoAlto['div'] > 1.5 , 'desviacionMontoPromedio'] = 1

# eliminamos las columnas innecesarias
montoAlto = montoAlto.drop(['rbhOrganization', 'Monto', 'porBuyerOrganization', 'cantLic',
'sumMonto', 'promMonto', 'div'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql
```

```
engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
montoAlto.to_sql(con=engine, name='desviacionMontoPromedio', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(montoAlto)) + " licitaciones, en "+ tFormateado)
```

4.9. Monto adjudicado sobre el promedio

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Obtención de datos de montos adjudicados y montos estimados para saber si lo adjudicado se
ajusta mucho a lo estimado
sqlMontosimilar = '''SELECT a.rbhCode, sum(b.porTotalAmount ) montoAdj, sum(case when
a.rbhEstimatedAmount is null then 0 else a.rbhEstimatedAmount end) monto , sum(case when
a.rbhEstimatedAmount is null then 0 else a.rbhEstimatedAmount end)/(case when
sum(b.porTotalAmount) < 1 then 1 else sum(b.porTotalAmount) end) div
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode'''

montosimilar = pd.read_sql(sqlMontosimilar,cnxn)


# Cerramos la conexión
cnxn.close()

#Calculamos el indicador primero con la división de monto por licitación sobre el promedio y
luego si esta es mayor que 2, entonces se considera riesgosa.
montosimilar['adjudicacionCercana'] = 0
montosimilar.loc[(montosimilar['div'] > 0.97) & (montosimilar['div'] < 1.03),
'adjudicacionCercana'] = 1

# eliminamos las columnas innecesarias
montosimilar = montosimilar.drop(['montoAdj', 'monto', 'div'], axis=1)

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql
```

```
engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
montosimilar.to_sql(con=engine, name='adjudicacionCercana', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(montosimilar)) + " licitaciones, en "+ tFormateado)
```

4.10. Código proveedor ausente

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados

# Query para indicador de ausencia de codigo proveedor
sqlSinProveedor = '''SELECT a.rbhCode, SUM(CASE WHEN b.porSellerOrganization IS NULL and b.porID
is not NULL and b.porBuyerStatus in (4, 5, 6, 12, 13, 14) THEN 1 ELSE 0 END) AS
codigoProvAusente
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcPOHeader b ON a.rbhCode = b.porSourceDocumentNumber
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25, 30)
AND a.rbhDocumentStatus in (8, 9, 10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode'''

sinProveedor = pd.read_sql(sqlSinProveedor,cnxn)


# Cerramos la conexión
cnxn.close()


# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
sinProveedor.to_sql(con=engine, name='codigoProvAusente', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(sinProveedor)) + " licitaciones, en "+ tFormateado)
```

4.11. Licitación desierta sin justificación

```
#Imports necesarios
import pyodbc #odbc para sqlserver
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados
# Query para indicador de ausencia de codigo proveedor
sqlDesSinJustificacion = '''SELECT rbhCode,
(CASE WHEN
(rbhDocumentStatus = 7 OR rbhdocumentstatus = 15)
AND rbhRevokeJustify IS NULL THEN 1
ELSE 0 END) AS desiertaSinJustificacion
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1'''

desSinJustificacion = pd.read_sql(sqlDesSinJustificacion,cnxn)


# Cerramos la conexión
cnxn.close()


# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
desSinJustificacion.to_sql(con=engine, name='desiertaSinJustificacion', if_exists='replace',
index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(desSinJustificacion)) + " licitaciones, en "+
tFormateado)
```

4.12. Licitación cancelada por reclamos

```
#Imports necesarios
import pyodbc #odbc para sqlserver
```

```
import time #librería necesaria para medir el tiempo de ejecución
import datetime
import pandas as pd
import numpy as np
from functools import reduce

# Inicio del "cronómetro"
tInicio = time.time()

# Apertura la conexión a SQL SERVER
cnxn = pyodbc.connect('string de conexion')

# Envío de las querys y lectura de los resultados
# Query para indicador de ausencia de codigo proveedor
sqlCancelReclamos = '''SELECT a.rbhCode , (CASE WHEN (COUNT(DISTINCT b.idReclamo )) > 3 AND
a.rbhDocumentStatus IN (15, 16) THEN 1 ELSE 0 END) AS 'anulacionPorReclamo'
FROM DCCPProcurement.dbo.prcRFBHeader a
LEFT JOIN DCCPReclamos.dbo.Reclamo b ON a.rbhExternalCode = b.numLicOC
LEFT JOIN DCCPProcurement.dbo.prcRFBDate d ON a.rbhCode = d.rbdRFBCode
WHERE YEAR(d.rbdOpeningDate) > 2013
and YEAR(d.rbdOpeningDate) < 2020
AND a.rbhOwnerName != 'Dirección de Compras y Contratación  Pública'
AND a.rbhProcessSubType in (1, 2, 3, 23, 24, 25,  30)
AND a.rbhDocumentStatus in (8,9,10, 7, 15, 16)
AND a.rbhProcessType = 1
GROUP BY a.rbhCode, a.rbhDocumentStatus'''

cancelReclamos = pd.read_sql(sqlCancelReclamos,cnxn)


# Cerramos la conexión
cnxn.close()

# Carga de datos en MySQL
from sqlalchemy import create_engine
import pymysql

engine = create_engine('mysql+pymysql://conexionMysql/redflags', echo = False)
cancelReclamos.to_sql(con=engine, name='anulacionPorReclamo', if_exists='replace', index=False)

# Término del "crónometro" y transformaciones a hora, minutos y segundos
tFinal = time.time()
tSegundosGenerico = tFinal - tInicio
tFormateado = str(datetime.timedelta(seconds=tSegundosGenerico))

# Resultados de la primera extracción
print("Terminado, se procesaron "+ str(len(cancelReclamos)) + " licitaciones, en "+ tFormateado)
```

Bibliografía

[1] División de capacitación de usuarios de ChileCompra, «Capacitación ChileCompra,» Enero 2017. [En línea]. Available: https://capacitacion.chilecompra.cl/mod/resource/view.php?id=493.

[2] División de capacitación de usuarios de ChileCompra, «Material de apoyo,» Julio 2010. [En línea]. Available: http://www.mercadopublico.cl/Portal/MP2/descargables/manual_compradores.pdf.

[3] DOZORRO, «Dozorro story: from an idea of a platform to the biggest monitoring community in Ukraine,» Transparency International Ukraine, Kiev, 2016.

[4] DOZORRO, «List of public procurement risk indicators,» 2016. [En línea]. Available: https://drive.google.com/open?id=15-Dp4XETxdI-zNJ-dnn3mHNgq_SA7xQ7OiZoKTnfG4g.

[5] A. Németh y T. Tátai, «Red Flags Project: New warning system for the identification of red flags in public procurement,» Transparency International Hungary, Budapest, 2015.

[6] M. Figueroa, D. González y T. Tabilo, «Experiencias, percepciones e interpretaciones: las compras públicas desde la visión de los proveedores,» Santiago, 2018.

[7] OECD, «Tool: Indicators of procurement risk,» 2009. [En línea]. Available: https://www.oecd.org/governance/procurement/toolbox/search/indicators-procurement-risk.pdf. [Último acceso: 2019].

[8] PriceWaterhouseCoopers, «Identifying and Reducing Corruption in Public Procurement in the EU,» 2013. [En línea]. Available: https://ec.europa.eu/anti-fraud/sites/antifraud/files/docs/body/identifying_reducing_corruption_in_public_procurement_en.pdf. [Último acceso: 2019].

[9] International Anti-Corruption Resource Center IACRC, «Guide to Combating Corruption & Fraud in Development Projects: Red Flags Listed by Project Cycle,» 2013. [En línea]. [Último acceso: 2019].

[10] ChileCompra, «Indicadores internos,» [En línea]. Available: https://drive.google.com/open?id=17wAaZI5CdEuzQSznlwNjAnxLKPk53aEtCnMaEFjjKes.

[11] M. FAZEKAS, «ASSESSING THE QUALITY OF GOVERNMENT AT THE REGIONAL LEVEL USING PUBLIC PROCUREMENT DATA,» European Union, Cambridge, 2017.

[12] Open Contracting Partnership; Development Gateway, «Red Flags for integrity: Giving the green lights to open data solutions,» Open Contracting Partnership, Washington, 2016.