PRUEBA TÉCNICA

Cornejo Martínez María de los Ángeles

Sección 1: Procesamiento y transferencia de datos

1.1. Carga de información

La información se cargó en una base de datos relacional MySQL, por medio de la herramienta Workbench con el menú "*Table Data Import Wizard*", el cual permite importar directamente los datos desde un archivo CSV o JSON a una base de datos ya creada.

Se seleccionó este tipo de base de datos por lo siguiente:

1. Al observar el conjunto de datos proporcionado se pudo notar que las columnas están relacionadas entre sí, por lo cual MySQL se adapta a ese enfoque, siendo una opción práctica y fácil de usar.
2. MySQL resulta una buena opción ya que, al utilizar Python para la extracción y transformación de datos, se pueden utilizar ciertas librerías que permiten llevar acabo estas operaciones de una manera fácil y clara. Este es el caso de las librerías "*pandas*" y "*sqlalchemy*", las cuales permiten establecer una conexión con la base de datos, crear un *dataframe* con los datos extraídos y, a partir de esto, exportar el conjunto de datos a otro formato (CSV, JSON, XML, etc) o bien comenzar a realizar transformaciones en los datos. Todo esto con muy pocas líneas de código y de forma muy intuitiva.
3. Para este escenario de prueba y con un conjunto de datos no muy grande, MySQL resulta como una opción viable y rápida de usar para las operaciones solicitadas.

1.2. Extracción

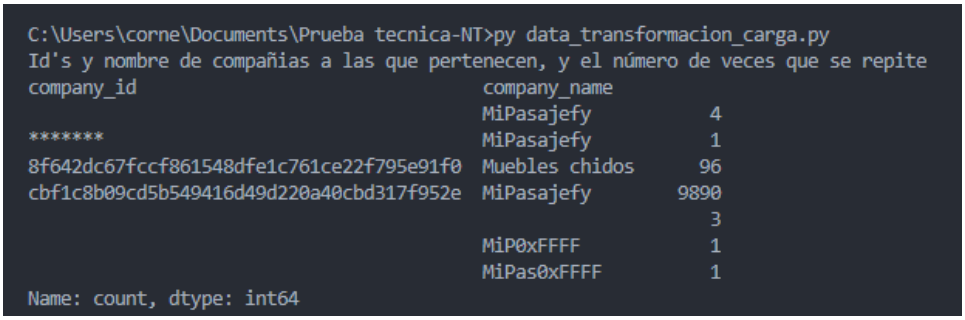
Para la extracción de datos se utilizó el lenguaje Python en conjunto con las librerías "*pandas*" y "*sqlalchemy*". El formato seleccionado fue JSON ya que:

1. Permite representar el conjunto de datos de forma más clara en cuestión de legibilidad.
2. Es fácil de manipular desde Python y compatible con las librerías que se manejan.
3. Este formato es flexible y es posible guardar distintos tipos de datos, no solo de texto.

1.3. Transformación

Para todo el proceso de transformación de los datos se utilizó la librería "*pandas*". Se creó un *dataframe* a partir de los datos extraídos del archivo JSON creado en el punto anterior. Las transformaciones hechas fueron:

1. Se modificaron los nombres de las columnas para que coincidieran con las del esquema propuesto y que se asociaran de este modo al momento de hacer la inserción en la base de datos con la tabla creada tomando en cuenta el mismo esquema.
2. Cambiar el tipo de dato de cada campo tomando en cuenta los tipos de datos del esquema propuesto, ya que este esquema se vería reflejado en la base de datos, y si los tipos de datos no coincidían se podrían tener conflictos al momento de cargar los datos.
   Las columnas modificadas fueron:
   - id -> string para coincidir con varchar en bd
   - company_name -> string para coincidir con varchar en bd
   - company_id -> string para coincidir con varchar en bd
   - amount -> float para coincidir con decimal en bd
   - status -> string para coincidir con varchar en bd
   - created_at -> datetime para coincidir con timestamp en bd
   - updated_at -> datetime para coincidir con timestamp en bd
3. Se validó que los valores de la columna amount no tuvieran más de 14 cifras, ya que en ese caso superarían la longitud definida en la base de datos, esto tomando en cuenta que en el esquema se propone para amount un tipo de dato decimal(16,2).
4. Se observó que cada company_id perteneciera solo a una compañía, es decir que los pares (company_id, company_name) fueran únicos. En este punto se identificó que existía el mismo company_id para más de una compañía, y que, a su vez, había compañías del mismo nombre con id's diferentes. Esto se muestra de forma más clara en la siguiente imagen:



A partir de esto, se pudo notar que existen id's inválidos como es el caso de "*******" o el que se encuentra en blanco (""), estos casos no cumplen con algunas cuestiones básicas de un id, como que debería ser 'no nulo' y cumplir con una estructura (ya sea cadena de caracteres o números enteros) que lo haga único. Por otro lado, un mismo company_id está asociado a más de un company_name, entre los cuales hay uno vacío y otros que parecen nombres erróneos del original.

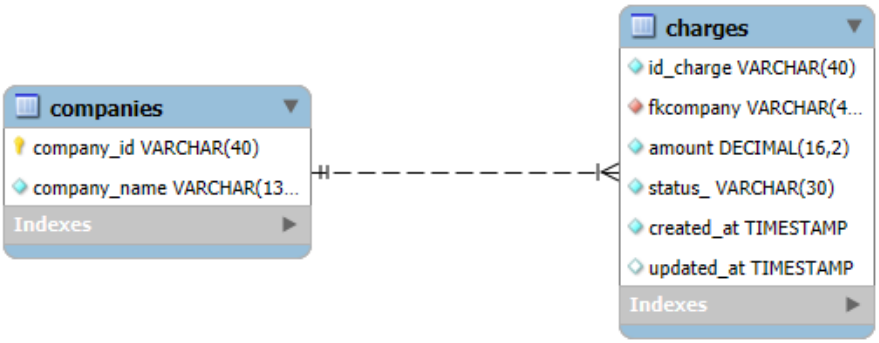
Para solucionar lo anterior se decidió llevar a cabo las siguientes acciones:

- Identificar y guardar en una lista el id que se asocia a más de una compañía (debería ser unico por compañia). Se genera una lista para cuando exista más de un id.
- Buscar el nombre de compañía que más veces se encuentra asignado al id problemático (el que está asociado a más de una compañía) para asumir el nombre de la compañía correcta.
- Guardar el par (company_id, company_name) encontrado en un diccionario que se tomará como los valores correctos.
- Mapear las columnas (company_name) y (company_id) usando el diccionario generado con los valores que se asumen son los correctos. De este modo, si existe un nombre correcto de compañía se le asigna su id correcto (corrigiendo los ids nulos o de '****') y viceversa, si existe un id correcto se le asigna el nombre correcto de la compañía que representa (corrigiendo casos como nombres nulos, 'MiP0xFFF' o 'MiPas0xFFFF').

Como conclusión de esta sección de transformación, se puede decir que el punto 4 fue el que representó un mayor reto ya que los casos mencionados pueden verse como datos erróneos o faltantes. Ante esto pueden existir otras acciones a tomar, por ejemplo, descartar completamente estos datos. Sin embargo, la solución propuesta fue pensando en un enfoque de no tener perdida de datos y "restaurar" los registro. Esto presentó para mí un reto al momento de ejecutarlo en código, ya que la lógica principal de tomar el id correcto asociado al nombre correcto, y corregirlo en los casos erróneos fue fácil de idear, pero al momento de decidir que funciones y qué métodos utilizar podía volverse más complejo si no tenia claro la secuencia de pasos a seguir.

Finalmente, cabe mencionar que desde mi perspectiva las acciones a tomar en casos como estos de datos faltantes o erróneos dependen mucho del contexto de un entorno real y de las reglas de negocio que se establezcan desde un inicio. Es por eso que el enfoque que llevé a cabo fue únicamente por considerar un escenario de pruebas y como algo más intuitivo.

1.4. Dispersión de la información



1.5. Vista MySQL

company_name	fecha	total_transaccionado
MiPasajefy	2019-01-01	4150.04
MiPasajefy	2019-01-02	17044.92
MiPasajefy	2019-01-03	6735.66
Muebles chidos	2019-01-03	3199.00
MiPasajefy	2019-01-04	6349.69
MiPasajefy	2019-01-05	5184.97
MiPasajefy	2019-01-06	4005.46
MiPasajefy	2019-01-07	26754.30
MiPasajefy	2019-01-08	5963.77
MiPasajefy	2019-01-09	5859.00
Muebles chidos	2019-01-09	399.00
MiPasajefy	2019-01-10	9040.75
MiPasajefy	2019-01-11	8646.82
MiPasajefy	2019-01-12	6456.20
MiPasajefy	2019-01-13	23541.02
MiPasajefy	2019-01-14	3884.07
Muebles chidos	2019-01-14	8197.00

vista_monto_diario_por_compa...