Práctica 4 Análisis de Datos y Visualización

Introducción

En esta práctica, se nos lleva de vuelta a la Práctica 1, pero esta vez de cara a realizar las mismas acciones con el *dataset* que hicimos la vez anterior, pero esta vez desde SQL. De la misma manera se nos pide realizar todo el proceso de ingesta de datos por *.csv* al *container* de Docker.

Datos proporcionados

Los datos con los que trabajaremos se encuentran en tres documentos:

- **'contac_center_data.csv':** se trata del primer documento a analizar. Contiene datos de llamadas de usuarios, el Código postal, su teléfono, su id de sesión, información sobre su Vivienda (chalet/piso, unifamiliar/adosado/bajo/intermedio, con/sin rejas, con/sin perro) y por último el tipo de producto que tiene contratado (*home basic/home premium/home premium plus*).
- **'renta_por_hogar.csv':** se trata del Segundo documento a analizar. dispone de los datos de la renta neta media por persona, la renta neta media por hogar, Media de la renta por unidad de consumo, Mediana de la renta por unidad de consumo, Renta bruta media por persona y Renta bruta media por hogar de cada Municipio, Distrito y Sección de la comunidad de Madrid, indicando en cada uno de estos el código postal y el código de cada Distrito/Sección. Además contiene los datos desde 2015 hasta 2020, almacenando el total de cada dato indicado anteriormente.
- **'delitos_por_municipio.csv':** se trata del tercer documento a analizar. Dispone de una cabecera y pie de página con información sobre los datos del documento. Los datos nos indican diferentes delitos cometidos en enero-marzo de 2019 y enero-marzo de 2020.

Ingesta de datos

Para el proceso de ingesta de datos, utilizaremos ‘’sqlalchemy’’ y una base de datos de tipo MySQL.

Reutilizaremos el *container* definido en clase para MySQL y nos apoyaremos del código generado en clase para la ingesta de datos en formato .csv.

En primer lugar, definimos el fichero Dockerfile de la siguiente manera:

```
FROM python:3.11-buster
WORKDIR /app
COPY . .
RUN pip install pymysql sqlalchemy pandas cryptography
ENTRYPOINT python insert_mysql_sqlalchemy.py
```

Para asegurarnos de tener todas las librerías que nos serán de utilidad a lo largo de la práctica. A continuación, definimos el fichero llamado ‘’insert_mysql_sqlalchemy.py’’, fichero que será ejecutado en el *build* de la imagen de Docker que realizará la ingesta de datos a la base de datos.

Autor: Martín de las Heras

```python
contact_center = pandas.read_csv('contac_center_data.csv', sep=';')
delitos_por_municipio = pandas.read_csv('delitos_por_municipio.csv', sep=';')
renta_por_hogar = pandas.read_csv('renta_por_hogar.csv', sep=';')

database_connection = sqlalchemy.create_engine('mysql+pymysql://root:my-secret-pw@172.17.0.2/superset').connect()
contact_center.to_sql(con=database_connection, name='contact_center', if_exists='replace')
delitos_por_municipio.to_sql(con=database_connection, name='delitos_por_municipio', if_exists='replace')
renta_por_hogar.to_sql(con=database_connection, name='renta_por_hogar', if_exists='replace')
```

Este fragmento de código simplemente carga los ficheros y los envía a la base de datos MySQL que definimos anteriormente.

Posteriormente, realizamos el comando ‘’docker . -t build sqlalchemy-contact_center’’. De esta manera se nos crea la imagen en Docker, ejecutamos esta (asegurándonos de que el *container* de MySQL esté en ejecución), y esto produce la subida de los datos a Docker.

Transformaciones de los datos

Una vez tenemos los datos en MySQL, accedemos a ellos y realizamos las siguientes modificaciones:

contac_center_data.csv

En esta tabla de la base de datos, ejecutamos varias *queries* de limpieza de valores vacíos y una de agrupación de filas para reducir el número de estas:

```sql
select sessionid, dni, telef, cp, duration_call_mins, producto, group_concat(funnel_Q)
from contact_center group by sessionID, dni, telef, cp, duration_call_mins, producto;
```

De esta manera, obtenemos una columna donde aparecen las características de cada ‘’sessionID’’, la cual se puede separar en columnas en función de los elementos que contenga.

renta_por_hogar.csv

En esta tabla, fundamentalmente haremos *queries* de limpieza, de manera que se reduzca el número de filas a solamente las que serán necesarias de cara al análisis:

```sql
delete from renta_por_hogar where periodo < 2019;
delete from renta_por_hogar where `indicadores de renta media y mediana`!='renta neta media por hogar';
delete from renta_por_hogar where secciones is not null;
delete from renta_por_hogar where distritos is null;
alter table renta_por_hogar drop column secciones;
```

De esta manera, hacemos cambios como eliminar las filas con datos anteriores a 2019, o los indicadores que no son de renta neta media por hogar.

delitos_por_municipio.csv

En primer lugar, para esta tabla, modificamos las cabeceras antes de realizar la ingesta, esto se puede hacer en SQL, pero es un proceso tedioso, mientras que en Python se puede automatizar de manera más fácil.

A continuación, eliminamos la primera fila, la última columna y alteramos la columna de los municipios para que aparezca solamente el nombre de este:

```sql
delete from delitos_por_municipio where municipio like '%comunidad de%';
alter table delitos_por_municipio drop column `Unnamed: 31`;
update delitos_por_municipio set municipio = replace(municipio, '- Municipio de ', '');
```

Unión

Lo último a realizar sobre estas tablas es unirlas en una propia, para ello, tenemos la siguiente *query*, la cual selecciona los valores que nos interesan, es decir, los comunes a las tres tablas:

```
select c.*, r.*, d.* from contact_center as c
join renta_por_hogar as r on r.municipios like concat('%', c.cp, '%')
join delitos_por_municipio as d on r.municipios like concat('%', d.municipio, '%');
```

Estos datos los podemos almacenar en una tabla nueva mediante la creación de la misma y la introducción de esta *query* al poblarla con los datos.

Conclusiones

En esta práctica, hemos experimentado cómo hacer una ingesta de datos de manera independiente, y también se nos ha introducido un poco al mundo de SQL, pero con datos ya familiares.

Esto de cara a la inteligencia artificial nos sirve, ya que la gestión de la información es crucial para todos los aspectos de la creación y el entrenamiento de cualquier modelo. Conocer las técnicas de ingesta de datos nos proporciona una herramienta para poder gestionar de manera más eficiente el acceso a los mismos, tanto de cara al tipo de base de datos a utilizar, como de cara a qué herramientas utilizar para realizar la misma ingesta.

Autor: Martín de las Heras