09/04/2023

Reporte de proyecto final

Nombre: Naomi Vanessa Lara Hernández

Carné: 18001404

Sección: L

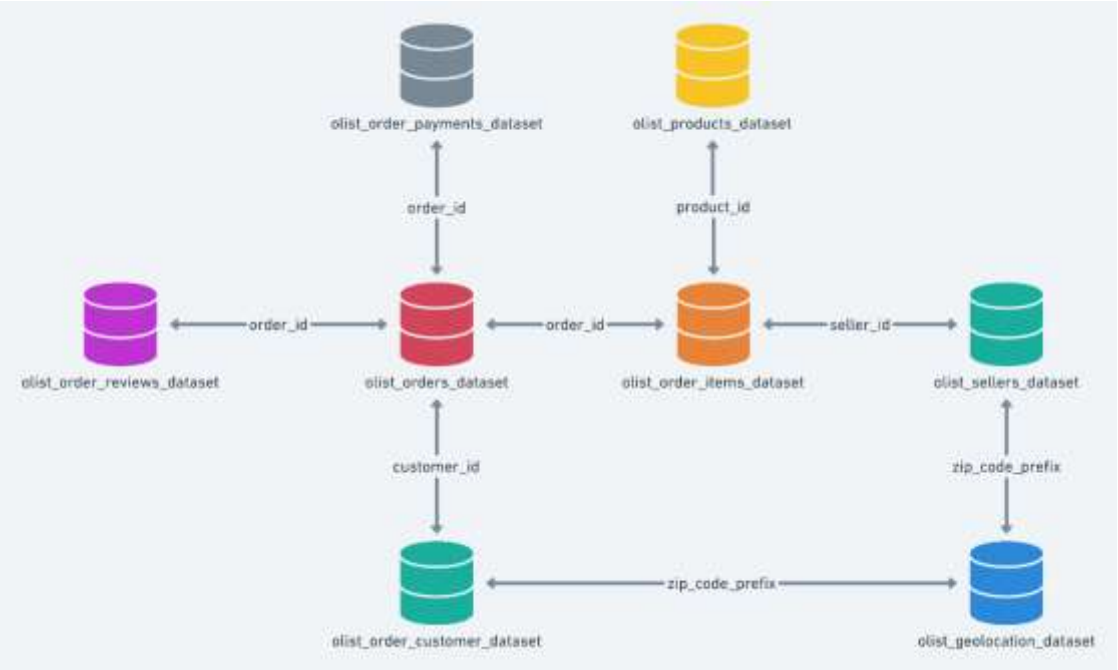
Obtención de fuente de datos:

Los datos utilizados en el proyecto fueron obtenidos del sitio web de kaggle: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

El dataset consiste en información sobre las ventas realizadas en Olist desde 2016 hasta 2018, la cual es un ecommerce que conecta pequeñas empresas de todo Brasil con los clientes a través de un contrato. Esos comerciantes pueden vender sus productos a través de la Tienda Olist y enviarlos directamente a los clientes utilizando los socios logísticos de Olist. Su funcionamiento es muy similar al de Amazon, con la diferencia que está enfocado primordialmente al transporte dentro de Brasil.

Después de que un cliente compra el producto de Olist Store, se notifica a un vendedor para cumplir con ese pedido. Una vez que el cliente recibe el producto, o vence la fecha estimada de entrega, el cliente recibe una encuesta de satisfacción por correo electrónico donde puede dejar una nota sobre la experiencia de compra y anotar algunos comentarios.

El modelo de datos que contiene la información del sistema, desde la compra hasta el envío y su reseña se puede observar en el siguiente diagrama:

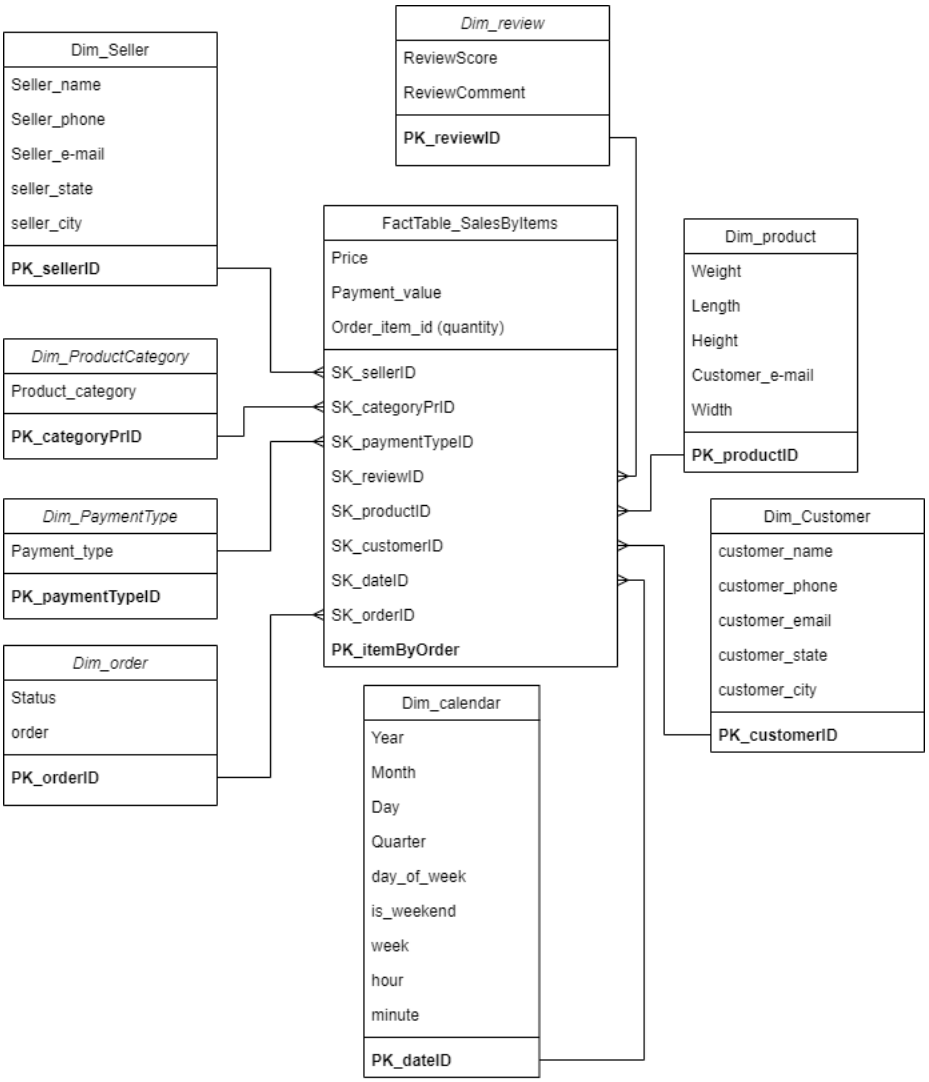


Preguntas de negocio por responder

Dentro de las preguntas que el modelo dimensional permitirá responder se encuentran:

1. ¿Qué categoría de productos son los que generan la mayor cantidad de ventas?
2. ¿En dónde se encuentra la mayor parte de los clientes (ciudad y estado)?
3. ¿Cuál es la estacionalidad de compra en línea que tienen los usuarios, existen horas dónde hay una mayor solicitud de ordenes?
4. ¿Cuáles son los productos mejor y peor calificados? En caso a los peor calificados, ¿a qué se debe?
5. ¿Cuál es el método de pago preferido por los clientes?

Modelo dimensional



Exploración y limpieza de datos

Se realizo la respectiva exploración y limpieza del conjunto de datos a través de Python. En la que se determinó:

1. Existe una relación de 1 a 1 entre id de la orden y el id del cliente
2. Una orden puede ser pagada por partes, es decir una parte del total de la orden está ingresada como tarjeta de crédito mientras que otra parte es transferencia a cuenta. También puede ser un mismo método de pago, pero ingresado por partes.
3. Para un mismo código postal, el nombre de ciudad y estado varían debido a que tiene acentos. Por lo que es necesario eliminar acentos para que exista uniformidad con los datos de cliente y vendedor en relación con su ubicación.
4. Los id de cliente, vendedor, orden y producto son de tipo texto de 32 caracteres. Por lo que se propone crear llave sorrugada para que el rendimiento del tiempo de consulta en la base de datos no se vea afectado, de igual forma si en algún momento se necesita actualizar o modificar id del usuario o producto se podrá realizar (en creación de escenario).
5. Filtración de columnas de interés previo a su ingreso a la base de datos (simulando sistema (escenario)), con la finalidad de reducir tiempo de ejecución.
6. Adicionalmente se agrega Fake data sobre datos personales (nombre, correo y teléfono) a cliente y vendedor, con la finalidad de simular el escenario del sistema lo más cercano posible a la realidad.

Creación de escenario

Después de explorar los datos y realizar su limpieza respectiva, se procede a crear la base de datos de PostgreSQL a través de los servicios que ofrecen AWS Amazon utilizando Python.

Creación de Servicio RDS

```python
try:
    response = aws_conn.create_db_instance(
            AllocatedStorage=10,
            DBName=config.get('RDS', 'DB_NAME'),
            DBInstanceIdentifier=rdsIdentifier,
            DBInstanceClass="db.t3.micro",
            Engine="postgres",
            MasterUsername=config.get('RDS', 'DB_USER'),
            MasterUserPassword=config.get('RDS', 'DB_PASSWORD'),
            Port=int(config.get('RDS', 'DB_PORT')),
            VpcSecurityGroupIds=[config.get('VPC', 'SECURITY_GROUP')],
            PubliclyAccessible=True
        )
    print(response)
except aws_conn.exceptions.DBInstanceAlreadyExistsFault as ex:
    print("La Instancia de Base de Datos ya Existe.")
```

Python

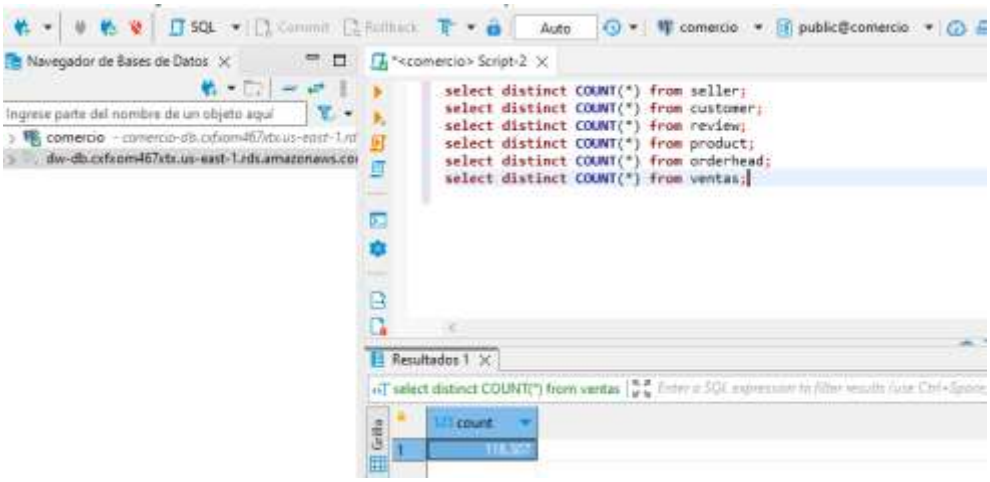
```
{'DBInstance': {'DBInstanceIdentifier': 'comercio-db', 'DBInstanceClass': 'db.t3.micro', 'Engine': 'postgres', 'DBInstanceStatus': 'creating
```

Conexión a Base de Datos desde Python

```python
import sql_queries

try:
    db_conn = psycopg2.connect(
        database=config.get('RDS', 'DB_NAME'),
        user=config.get('RDS', 'DB_USER'),
        password=config.get('RDS', 'DB_PASSWORD'),
        host=RDS_HOST,
        port=config.get('RDS', 'DB_PORT')
    )

    cursor = db_conn.cursor()
    cursor.execute(sql_queries.DDL_QUERY)
    db_conn.commit()
    print("Base de Datos Creada Exitosamente")
except Exception as ex:
    print("ERROR: Error al crear la base de datos.")
    print(ex)
```

```
Base de Datos Creada Exitosamente
```

Y se insertan los datos a cada una de las tablas.



Construcción de datawarehouse

Se lleva a cabo las consultas en la base de datos que contiene el escenario del sistema original y se construyen las dimensiones para el data warehouse.

```
postgres_driver = f"""postgresql://{config.get('RDS', 'DB_USER')}:{config.get('RDS', 'DB_PASSWORD')}@{RDS_HOST}:{config.get('RDS', 'DB_P
```

Python

Creación de dimensiones

```
sql_query = 'SELECT * FROM customer;'
dimcustomer = pd.read_sql(sql_query, postgres_driver)
dimcustomer.head()
```

Python

	customersk	customer_id	name	email	phone	customer_city	customer_state
0	2610	06b8999e2fba1a1fbc88172c00ba8bc7	Raquel Judd	Raquel_Judd2052@iscmr.app	Raquel Judd	franca	SP
1	9561	18955e83d337fd6b2def6b18a428ac77	Sharon Walsh	Sharon_Walsh658@yfxpw.shop	Sharon Walsh	sao bernardo do campo	SP
2	30460	4e7b3e00288586ebd08712fdd0374a03	Marvin Radcliffe	Marvin_Radcliffe1768@bu2lo.directory	Marvin Radcliffe	sao paulo	SP
3	69605	b2b6027bc5c5109e529d4dc6358b12c3	Javier Stone	Javier_Stone 9704@karnv.digital	Javier Stone	mogi das cruzes	SP
4	30707	4f2d8ab171c80ec8364f7c12e35b23ad	Carl Palmer	Carl_Palmer3085@qu9ml.info	Carl Palmer	campinas	SP

Realizando como único tipo de consulta “SELECT * FROM…” En Python se construye la dimensión de calendario y se realizan los respectivos joins.

```
dimFechaHora['dimfecha'] = dimFechaHora['year'].astype(str) + dimFechaHora['month'].astype(str)
dimFechaHora['dimfecha'] = dimFechaHora['dimfecha'].astype(str) + dimFechaHora['day'].astype(str)
dimFechaHora['dimfecha'] = dimFechaHora['dimfecha'].astype(str) + dimFechaHora['hour'].astype(str)
dimFechaHora['dimfecha'] = dimFechaHora['dimfecha'].astype(str) + dimFechaHora['minute'].astype(str)
dimFechaHora.drop_duplicates(inplace=True)
dimFechaHora.head()
```

	order_purchase_timestamp	year	month	quarter	day	week	dayofweek	hour	minute	is_weekend	dimfecha
0	2017-10-02 10:56:33	2017	10	4	2	40	0	10	56	0	20171021056
3	2018-07-24 20:41:37	2018	7	3	24	30	1	20	41	0	20187242041
4	2018-08-08 08:38:49	2018	8	3	8	32	2	8	38	0	201888838
5	2017-11-18 19:28:06	2017	11	4	18	46	5	19	28	0	201711181928
6	2018-02-13 21:18:39	2018	2	1	13	7	1	21	18	0	20182132118

Posteriormente, se agrega información de los archivos provenientes de S3

```
[41]
#extraemos todo lo que está en el bucket
remoteFileList = []
for objt in s3.Bucket(S3_BUCKET_NAME).objects.all():
    remoteFileList.append(objt.key)

remoteFileList
```

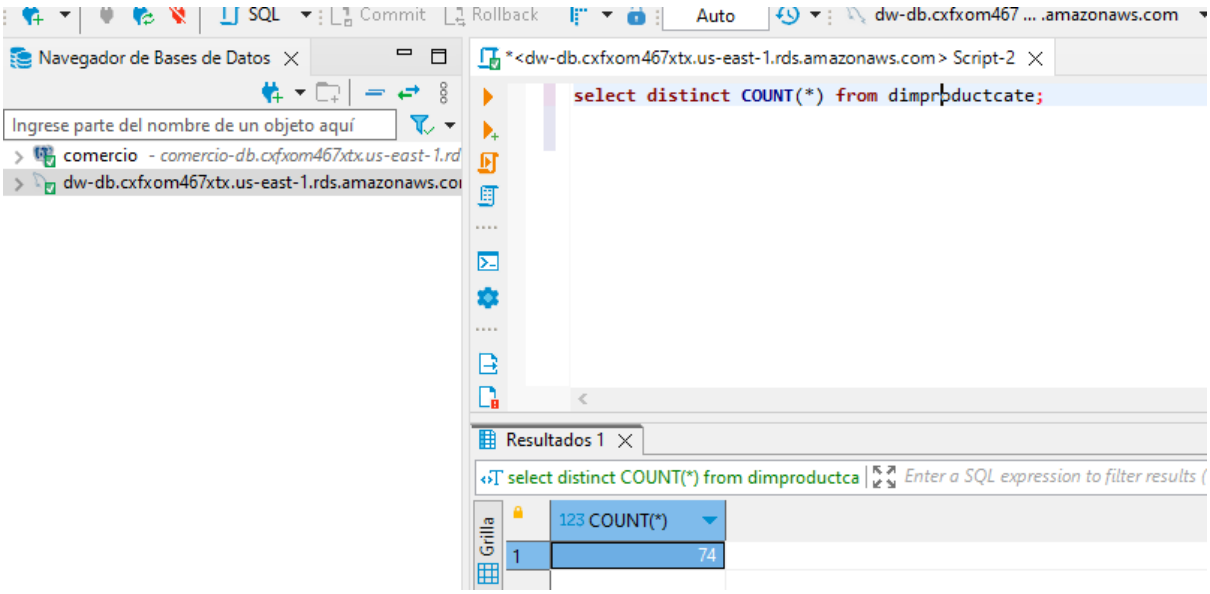
```
['paymentype.xlsx', 'productcate.xlsx']
```

Se crea el data warehouse en MySQL

```
import create_dw_query
import mysql.connector as mysqlC
try:
    myDw = mysqlC.connect(
    host=RDS_DW_HOST,
    user=config.get('RDS_MYSQL', 'DB_USER'),
    password=config.get('RDS_MYSQL', 'DB_PASSWORD'),
    database=config.get('RDS_MYSQL', 'DB_NAME')
    )

    mycursor = myDw.cursor()
    mycursor.execute(create_dw_query.CREATE_DW, multi=True)

    myDw.close()

    print("Data Warehouse Creado Exitosamente")
except mysqlC.Error as err:
    print("Something went wrong:{}".format(err))
```

```
Data Warehouse Creado Exitosamente
```

Y se lleva a cabo la inserción de datos a MySQL



Enlace a video de youtube:

https://youtu.be/N7QnLU8XHbo