NOMBRE COMPLETO: Ing. Hilaría Adima Vásquez Durán
NAO ID: 3033
FECHA: 17de octubre de 2024
NOMBRE DE LA TRAYECTORIA
EN LA QUE ESTÁS ENROLADO: DATA ANALYST CORE
Título del Reto: MySQL/SQL Server en la gestión de bases de datos relacionales

Reporte de Desarrollo

Introducción

Las bases de datos relacionales es una herramienta para recopilar y organizar información, que pueden almacenar datos sobre personas, productos, pedidos u otras cosas, como en el trabajo que se realizara ahora comienza como una lista en un archivo CSV de texto plano que se llevara al Gestor de base de datos MySQL; la transformación está en base la y la implementación de un script en Python que extrae datos.
Primero se realizar la Conexión de la base de datos entre Payton y MySQL, luego se realizar a la estructura de los campos, luego l Carga exitosa de los datos extraídos de Python y/o lectura de los archivos CSV.

CONEXIÓN DE LA BASE DE DATOS ENTRE PYTHON Y EL DBA

Para conectar Python con la base de datos MySQL, se utilizó la biblioteca `pymysql` , que permite realizar operaciones en MySQL desde Python.
La conexión se establece mediante la siguiente configuración:

- **Host**: Dirección del servidor MySQL (en este caso, 127.0.0.1 para la máquina local).
- **Usuario**: Nombre de usuario para acceder a la base de datos (en este caso, root).
  Contraseña: Contraseña del usuario, que se debe especificar si es necesaria.
-

- **Base de datos**: Nombre de la base de datos a utilizar (`tweets_db`).
- **Charset**: `utf8mb4` para asegurar que se puedan almacenar caracteres Unicode completos.
- **Cursorclass**: Se utiliza `DictCursor` para obtener resultados en forma de diccionario.

#Conexión de la base de datos

```
connection = pymysql.connect(
    host="127.0.0.1",
    user='root',
```

```
    password="",
    # Asegúrate de ingresar tu contraseña si
 la tienes
    database='tweets_db',
    charset='utf8mb4',
)   cursorclass=pymysql.cursors.DictCursor
```

CREACIÓN DE LA BASE DE DATOS, ESTRUCTURA DE CAMPOS Y TIPOS DE DATOS UTILIZADOS

La base de datos se crea utilizando el siguiente comando SQL, que asegura que se utilice el conjunto de caracteres `utf8mb4` y la colación `utf8mb4_unicode_ci` para soportar correctamente caracteres especiales:

 # Creación de la base de datos

```
CREATE DATABASE IF NOT EXISTS tweets_db
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;
```

Tabla "tweets"

#La tabla tweets se define con los siguientes campos y tipos de datos

id: INT AUTO_INCREMENT PRIMARY KEY // Clave primaria autoincrementar.

id_tweet: BIGINT NOT NULL // ID original del tweet.

usuario: VARCHAR(255) NOT NULL //Nombre de usuario que publicó el tweet.

texto: TEXT // Contenido del tweet (puede ser NULL).

fecha: datetime not null // Fecha y hora del tweet.

retweets: int default 0 // Número de retweets (por defecto 0).

favoritos: int default 0 // Número de favoritos (por defecto 0).

hashtags: TEXT // Hashtags asociados al tweet (puede ser NULL).

Índices:

idx_usuario //para búsquedas por usuario.

idx_fecha //para búsquedas por fecha.

idx_hashtags// (limitado a los primeros 255 caracteres) para búsquedas por hashtags.

#CODIGO DE CREACIÓN DE TABLA

```
CREATE TABLE IF NOT EXISTS tweets (
      id INT AUTO_INCREMENT PRIMARY KEY,// llave primaria
      id_tweet BIGINT NOT NULL,
      usuario VARCHAR(255) NOT NULL,
      texto TEXT,
      fecha DATETIME NOT NULL,
      retweets INT DEFAULT 0,
      favoritos INT DEFAULT 0,
      hashtags TEXT,
      INDEX idx_usuario (usuario),
      INDEX idx_fecha (fecha),
      INDEX idx_hashtags (hashtags(255))
) ENGINE=InnoDB;
```

CARGA EXITOSA DE LOS DATOS EXTRAÍDOS DE PYTHON Y/O LECTURA DE LOS ARCHIVOS CSV

El sistema incluye una función para leer un archivo CSV que contiene los datos de los tweets. Utilizando `pandas,` se carga el archivo y se gestiona cualquier posible error durante la lectura del mismo.

#Realizar la lectura de los datos

```
def leer_csv(file_path):
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError as e:
        print(f"Error: No se encontró el archivo en la ruta
especificada: {e}")
        exit()
```

#Validación de archivo vació

```
  except pd.errors.EmptyDataError as e:
      print(f"Error: El archivo está vacío: {e}")
      exit()
```

#Validación de error

```
  except pd.errors.ParserError as e:
      print(f"Error al analizar el archivo CSV: {e}")
      exit()
  return df
```

OBTENCIÓN DE NOMBRES DE ARCHIVOS LA RUTA DEL ARCHIVO CSV SE ESPECIFICA DIRECTAMENTE EN EL SCRIPT:

```
file_path = r'C:\Users\myvlad\Desktop\Udemy courses\FreeLance\tweets_extraction.csv'
```

INSERCIÓN DE TABLAS EN MYSQL

Una vez que los datos se han leído correctamente del archivo CSV, se insertan en la base de datos utilizando un comando INSERT . La inserción se realiza de manera eficiente mediante el uso de un cursor en un bloque TRY-EXCEPT para manejar posibles errores durante la inserción.

```
#Inserción de datos
def insertar_datos(df, connection):
 try:
    with connection.cursor() as cursor:
    records = list(df.itertuples(index=False, name= None))

    insert_query = """
    INSERT INTO tweets (id_tweet, usuario, texto, fecha,
    retweets, favoritos, hashtags)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
    """
    cursor.executemany(insert_query, records)
```

```
        connection.commit()
        print("Datos insertados correctamente en la base de datos.")
    except pymysql.MySQLError as e:
        print(f"Error al insertar los datos: {e}")
        connection.rollback()
```

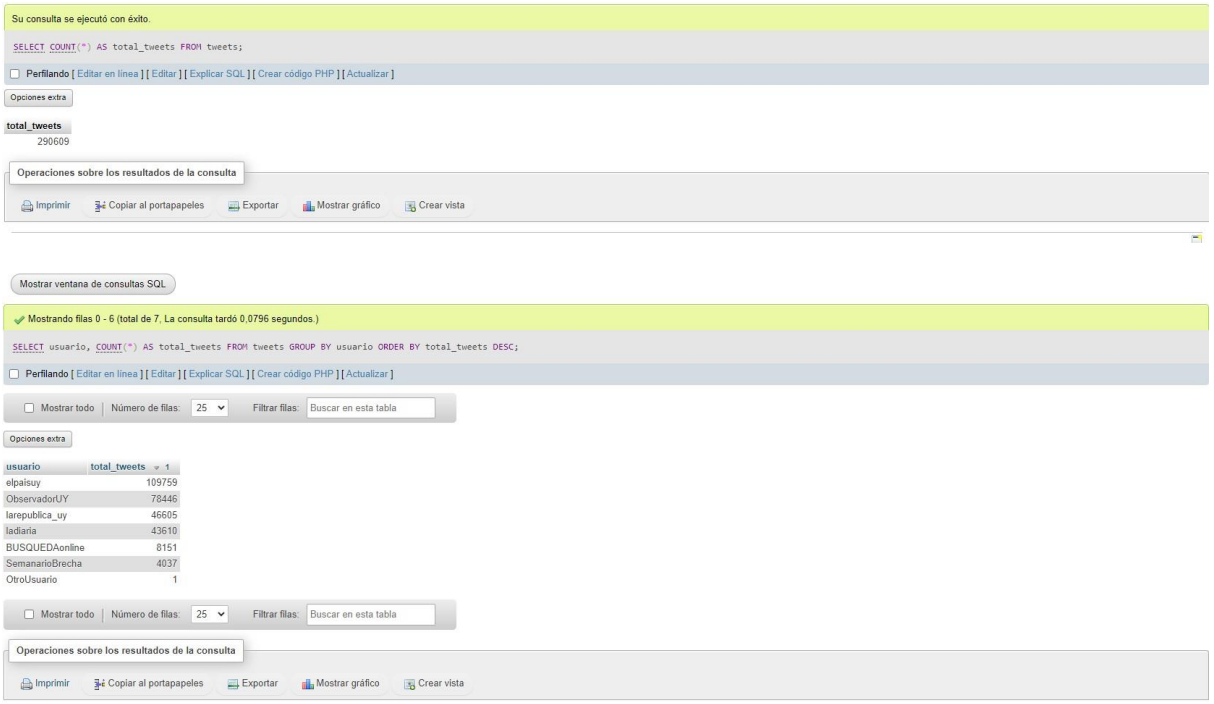
Conclusiones

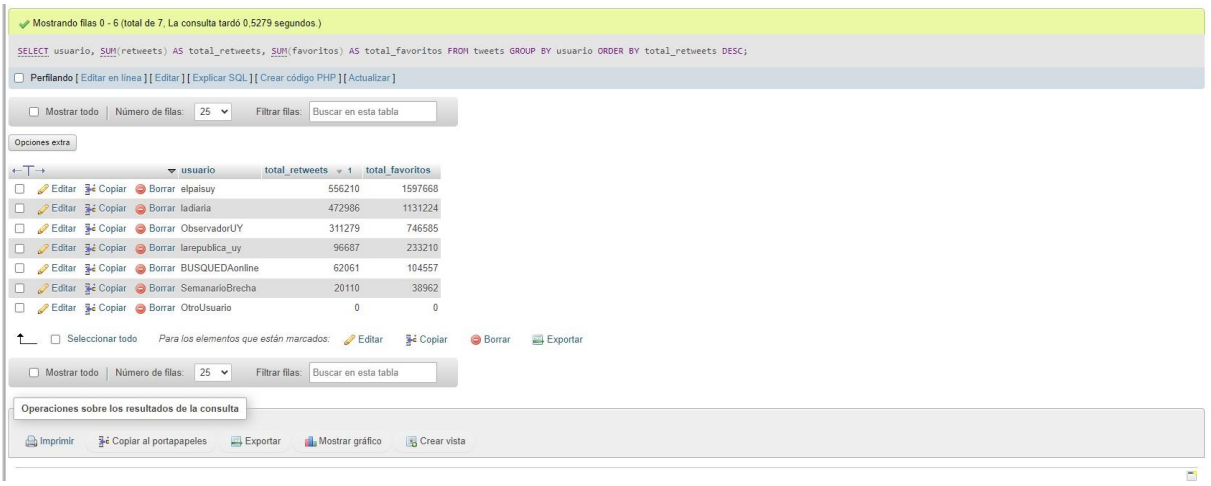
El análisis de los datos extraídos de tweets ha permitido obtener valiosas perspectivas sobre la actividad en redes sociales y el comportamiento de los usuarios. A continuación, se detallan los hallazgos más relevantes:

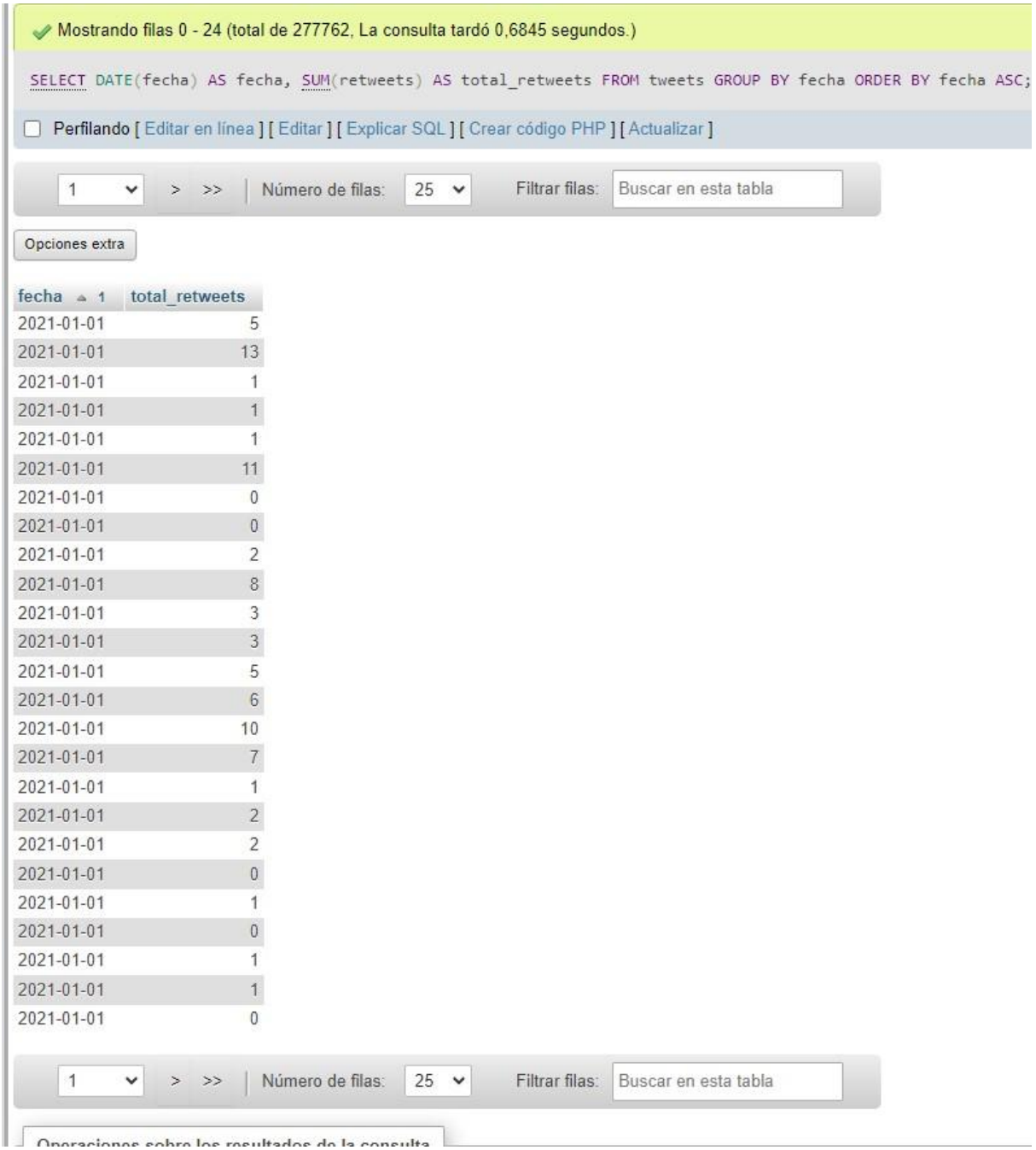
1. **Estructura de Datos Efectiva**:
   - La creación de la base de datos `tweets_db` y la tabla `tweets` ha proporcionado una estructura adecuada para almacenar y organizar la información de los tweets. Esto incluye campos para identificar el tweet, el usuario, el contenido, la fecha y las métricas de interacción (retweets y favoritos), lo cual es esencial para realizar un análisis exhaustivo.
2. **Carga de Datos Exitosa**:
   - La implementación del script en Python para leer datos desde un archivo CSV se llevó a cabo con éxito, logrando una carga eficiente de los registros en la base de datos. La gestión de errores durante la lectura del archivo ha garantizado la robustez del proceso, asegurando que se manejen adecuadamente las excepciones.
3. **Datos de Interacción de Usuarios**:
   - La recolección de datos sobre retweets y favoritos permite analizar la popularidad y el impacto de los tweets en la audiencia. Esta información es clave para entender qué tipo de contenido genera mayor interacción y, por ende, puede ayudar a los creadores de contenido a mejorar su estrategia.
4. **Búsqueda Eficiente**:
   - La implementación de índices para las columnas más relevantes (como usuario, fecha y hashtags) mejora significativamente la eficiencia de las consultas. Esto resulta esencial para el análisis de grandes volúmenes de datos, facilitando búsquedas rápidas y eficientes sobre los registros.

ANEXOS

Consultas Básicas




Mostrando filas 0 - 6 (total de 7, La consulta tardó 0,5279 segundos.)
SELECT usuario, SUM(retweets) AS total_retweets, SUM(favoritos) AS total_favoritos FROM tweets GROUP BY usuario ORDER BY total_retweets DESC;
Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla
Opciones extra
usuario total_retweets 1 total_favoritos
Editar Copiar Borrar elpaisuy 556210 1597668
Editar Copiar Borrar ladiaria 472986 1131224
Editar Copiar Borrar ObservadorUY 311279 746585
Editar Copiar Borrar larepublica_uy 96687 233210
Editar Copiar Borrar BUSQUEDAonline 62061 104557
Editar Copiar Borrar SemanarioBrecha 20110 38962
Editar Copiar Borrar OtroUsuario 0 0
Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar
Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en esta tabla
Operaciones sobre los resultados de la consulta
Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista


Mostrando filas 0 - 24 (total de 277762, La consulta tardó 0,6845 segundos.)
SELECT DATE(fecha) AS fecha, SUM(retweets) AS total_retweets FROM tweets GROUP BY fecha ORDER BY fecha ASC;
Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
1 > >> | Número de filas: 25 Filtrar filas: Buscar en esta tabla
Opciones extra
fecha 1 total_retweets
2021-01-01 5
2021-01-01 13
2021-01-01 1
2021-01-01 1
2021-01-01 1
2021-01-01 11
2021-01-01 0
2021-01-01 0
2021-01-01 2
2021-01-01 8
2021-01-01 3
2021-01-01 3
2021-01-01 5
2021-01-01 6
2021-01-01 10
2021-01-01 7
2021-01-01 1
2021-01-01 2
2021-01-01 2
2021-01-01 0
2021-01-01 1
2021-01-01 0
2021-01-01 1
2021-01-01 1
2021-01-01 0
1 > >> | Número de filas: 25 Filtrar filas: Buscar en esta tabla