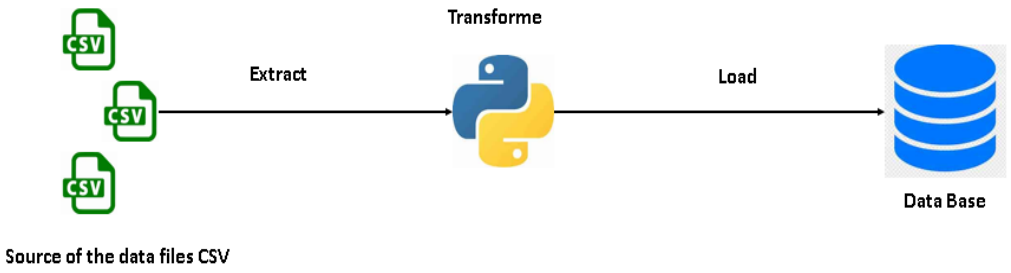
Oussama LAHLIMI

Documentation of Data Engineering Challenge: E-Commerce Data Analytics

Date: 2024/08/16

- **ETL processes for this challenge:**



  - ✓ **Objective of this challenge is**: Extract raw data from CSV files, transform it into a clean and usable format with python I do that, I verified I's transformation with my analysis the quality of this data like non-null value or duplication of the data I remove the fake data and the data (rows), and load it into a structured relational database **(postgresql**). The goal is to ensure that the data is properly formatted, cleansed, and stored in a way that makes it easy to query and analyze, and the also to create 3 scripts SQL.
  - ✓ The **library** with python I used the **pandas** for the extraction and the transformation and the load into DB I use **sqlalchemy**.
  - ✓ We can schedule automation this process ETL with python to be extract transform and load into database real time (if the data it's the same resources).

- **Load data :**

  load_to_db(df, table name, engine) ; 'it's take three 3 argument the data frame in this case it's the df from all resources files and the table name it's the name of the table he create in our data base and the last argument it's just the connection between the data base.

  This is the function responsible for the loading the data into data base.

- **SQL Query 1**

  Calculate the total **sales** revenue for each **product** category in the last **month**.

  To answer of this question I we need the tables he responsible and content this attributes and understand this question.

  - ✓ The **df_order_items DataFrame** contains order items, including price and **product_id**.
  - ✓ The **df_products DataFrame** contains product details, including **product_id** and **product_category_name**.
  - ✓ The **df_orders DataFrame** contains order details, including **order_purchase_timestamp**.

  This query calculates the total revenue for each product category in the last month by joining the order_items, products, and orders tables. The revenue is calculated as the sum of the price multiplied by the quantity (assumed as order_item_id).

```
SELECT p.product_category_name, SUM(oi.price * oi.order_item_id) AS total_revenue
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN orders o ON oi.order_id = o.order_id
WHERE  o.order_purchase_timestamp >= date('now', '-1 month')
GROUP BY p.product_category_name
ORDER BY  total_revenue DESC;
```

- **SQL Query 2**

Top 5 Customers by Total Spending:

- ✓ This query identifies the top 5 customers by their total spending, calculated by summing up the order item prices for each customer.
- ✓ The **df_customers** DataFrame contains customer details, including **customer_id**.
- ✓ The df_order_items DataFrame contains order items, including **order_id** and **price**.
- ✓ The df_orders DataFrame contains order details, including **order_id** and **customer_id**.

```
SELECT c.customer_unique_id,SUM(oi.price * oi.order_item_id) AS total_spent
FROM order_items oi
JOIN orders o ON oi.order_id = o.order_id
JOIN customers c ON o.customer_id = c.customer_id
GROUP BY c.customer_unique_id
ORDER BY total_spent DESC
LIMIT 5;
```

- **SQL Query 3**

New Customers Acquired Each Month:

- ✓ This query calculates the number of new customers acquired each month in the past year. The strftime function is used to extract the month and year from the order_purchase_timestamp.
- ✓ The **df_customers** DataFrame contains customer details, including **customer_id** and **order_purchase_timestamp**.

```
SELECT strftime('%Y-%m', o.order_purchase_timestamp) AS month, COUNT(DISTINCT c.customer_id) AS new_customers
FROM  orders o
JOIN customers c ON o.customer_id = c.customer_id WHERE o.order_purchase_timestamp >= date('now', '-12 months')
GROUP BY month ORDER BY month DESC;
```