Lufthansa
Industry Solutions

Building an ETL pipeline

Date: 25.09.2024

Lufthansa Industry Solutions

Table of Content

1 Introduction

This exercise focuses on data cleaning, creating calculated columns, using window functions, and saving the results to a data warehouse. The data will be split into fact and dimension tables based on the Olist dataset.

2 Data Sets

Download the dataset from Kaggle's Brazilian E-Commerce Public Dataset by Olist. The dataset includes multiple CSV files representing orders, customers, products, reviews, sellers, and payments.

CSV Files Overview

- Orders (olist_orders_dataset.csv): Contains order information such as order status, purchase date, and delivery date.
    - Fact Columns: Order ID, Order Status, Purchase Date, Estimated Delivery Date.
    - Dim Columns: Customer ID, Seller ID, Order Status, Order Purchase Timestamp, Order Approved At, Order Delivered Carrier Date, Order Delivered Customer Date, Order Estimated Delivery Date.
- Customers (olist_customers_dataset.csv): Information about customers and their geographic data.
    - Fact Columns: None (this is a dimension table).
    - Dim Columns: Customer ID, Customer Zip Code Prefix, Customer City, Customer State.
- Products (olist_products_dataset.csv): Contains product-related information.
    - Fact Columns: None (this is a dimension table).
    - Dim Columns: Product ID, Product Category, Product Name Length, Product Description Length, Product Photos Quantity, Product Weight, Product Length, Product Height, Product Width.
- Sellers (olist_sellers_dataset.csv): Details about sellers and their geographic information.
    - Fact Columns: None (this is a dimension table).
    - Dim Columns: Seller ID, Seller Zip Code Prefix, Seller City, Seller State.
- Order Items (olist_order_items_dataset.csv): Details about each item in an order, including price, freight value, and the connection between orders and products.
    - Fact Columns: Order Item ID, Product Price, Freight Value.
    - Dim Columns: Order ID, Product ID, Seller ID, Shipping Limit Date.
- Order Payments (olist_order_payments_dataset.csv): Payment information associated with each order.
    - Fact Columns: Payment Value, Payment Installments.
    - Dim Columns: Order ID, Payment Type.
- Order Reviews (olist_order_reviews_dataset.csv): Customer reviews and ratings for orders.

- Fact Columns: Review Score, Review Creation Date, Review Answer Timestamp.
- Dim Columns: Order ID, Review Comment Title, Review Comment Message.

3 Technology Stack

For this exercise, you will need to install on your local computer:

- SQL Server 2022
- SQL Server Management Studio

Here is a link that may help you in the process: https://www.sqlservertutorial.net/install-sql-server/

- Python 3.9 or later
- Anaconda

Here is a link that may help you in the process: https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

Hint: For data analysis in Python, the most used library is pandas, which reads the source data sets and transforms them in DataFrame object, which represents the data in a tabular form. Here is a link that may help you to get familiar with Pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html. To create the python script, you can use Jupyter Notebooks that can be launched from Anaconda. In order to access the DB from the Jupyter Notebook local server, you need to install pyodbc module. In this way, you can select/insert/update/delete data from SQL Server tables.

!Important! The ETL pipeline can be built in SQL Server or in Python or any other scripting language with which you feel comfortable. If you can built the entire ETL pipeline in SQL Server and in addition do some small transformation of your choice in Python (or any other scripting language), you are welcome to do so.

4 Requirements

4.1 Data Cleaning and Transformation

- Load all CSV files into a Jupyter Notebook using Pandas.
- Clean the datasets:
    - Handle missing values by filling them with appropriate data or dropping rows/columns.
    - Remove any duplicate rows.

- Deliverable: Python code (Jupyter Notebook) showing the task completed.
  Hint: Use Pandas' dropna(), fillna(), and drop_duplicates() for data cleaning.

4.2 Creating Calculated Columns

- Create the following calculated columns in the order items dataset:
  - Total Price: Sum of product price and freight value.
  - Delivery Time: Difference between the delivery date and the order purchase date.
  - Payment Count: Sum of payment installments for each order.
  - Profit Margin: Subtract freight value from product price to calculate a rough profit estimate.
- Deliverable: Python code (Jupyter Notebook) showing the steps to clean the data.
  Hint: Use Pandas' pd.to_datetime() to handle date conversions and create new columns using basic arithmetic operations.

4.3 Using Window Functions Over Partitions (Pandas)

- Using Pandas' window functions, create a rolling sum or cumulative metric for:
  - Total Sales per Customer: A running total of product price for each customer partitioned by Customer ID.
  - Average Delivery Time per Product Category: A rolling average of delivery time partitioned by product category.
- Deliverable: Python code (Jupyter Notebook) showing the task completed.
  Hint: Use Pandas' groupby() with the cumsum() function for the running total and rolling() for the rolling average.

4.4 Saving Processed Data to SQL Server (Fact & Dimension Tables)

- After cleaning and transforming the data, split it into a fact table and 3-4 dimension tables:
  - Fact Table: Order Items with calculated columns (Total Price, Delivery Time, etc.).
  - Dimension Tables:
    - Customers: Customer details including their geographic information.
    - Products: Product-related information.
    - Sellers: Seller details including geographic data.
    - Date: Order Purchase Date and Delivery Date as a dimension for time.
- Save these tables into Microsoft SQL Server using SQLAlchemy or PyODBC.
- Deliverable: Python code (Jupyter Notebook) showing the task completed and the SQL Server instance with the tables
  Hint: Use SQLAlchemy's to_sql() method to save Pandas DataFrames to SQL Server.

4.5 SQL Server Validation and Reporting

- Write a SQL query in SSMS to validate the data uploaded to SQL Server:
  - Query total sales per product category from the fact table.
  - Query the average delivery time per seller from the fact table.
  - Query the number of orders from each state from the customer dimension.
- Deliverable: SQL script used for the queries.
- Hint: Use GROUP BY in SQL for both product categories and state-wise queries.

General Hints:

- SQLAlchemy: Use to_sql() to write Pandas DataFrames directly to SQL Server.
- Window Functions in Pandas: Use groupby() with cumulative and rolling functions for running totals and averages.
- Pandas: Clean and process the data in a tabular format.