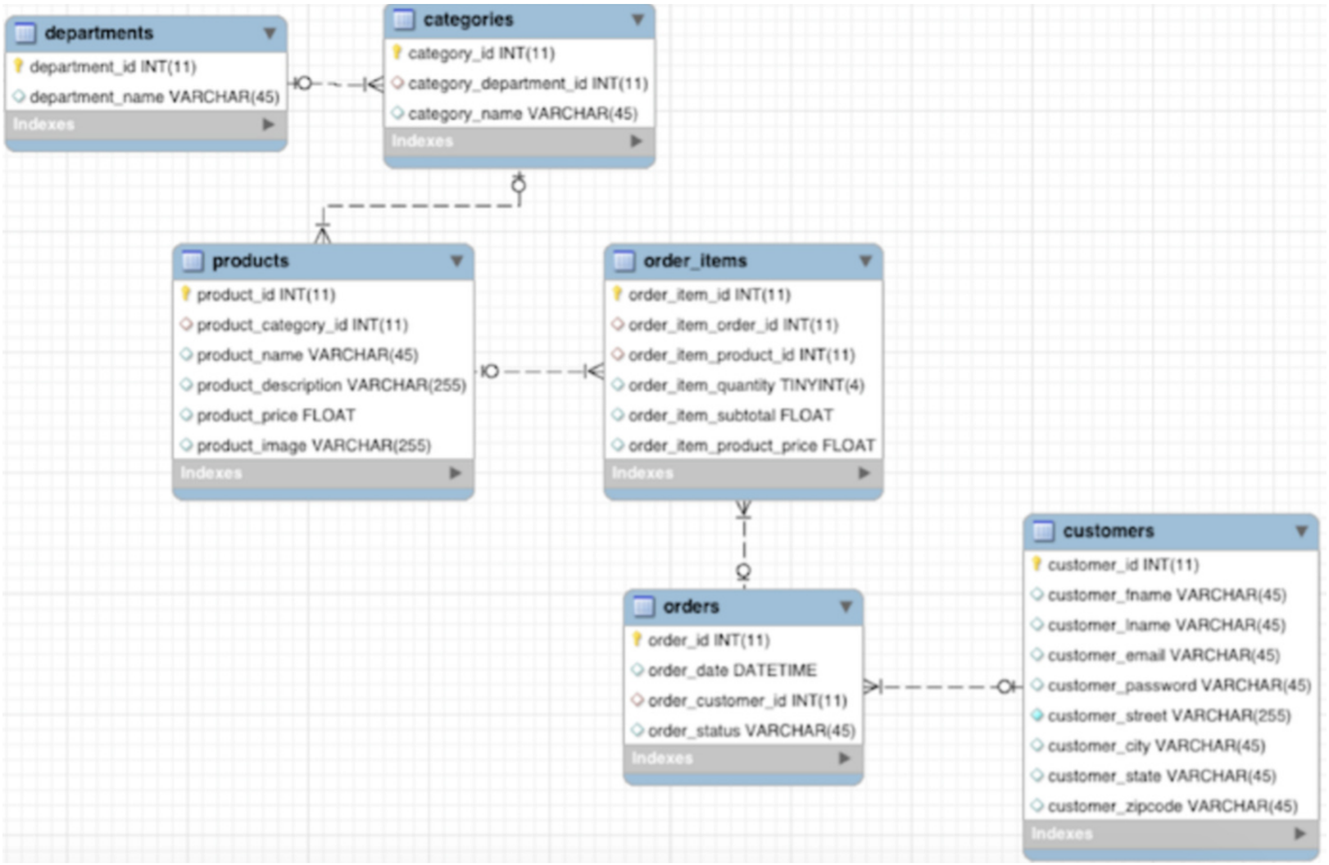
Project 1 - File Format Converter Handout

Overview

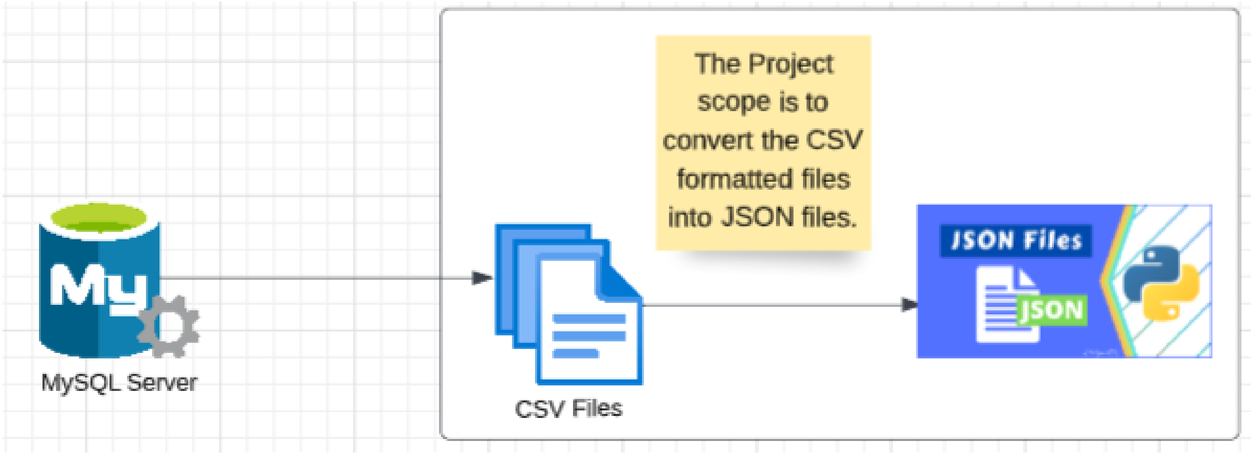
The objective of this project is to develop solutions based on the design provided. In this case, the source data was obtained in the form of CSV files from a MySQL DB.

To improve the efficiency of our data engineering pipelines, we need to convert these CSV files into JSON files, since JSON is better to use in downstream applications than CSV files. The scope of this project involves converting CSV files into JSON files.

Data Model Details:


departments
department_id INT(11)
department_name VARCHAR(45)
Indexes
categories
category_id INT(11)
category_department_id INT(11)
category_name VARCHAR(45)
Indexes
products
product_id INT(11)
product_category_id INT(11)
product_name VARCHAR(45)
product_description VARCHAR(255)
product_price FLOAT
product_image VARCHAR(255)
Indexes
order_items
order_item_id INT(11)
order_item_order_id INT(11)
order_item_product_id INT(11)
order_item_quantity TINYINT(4)
order_item_subtotal FLOAT
order_item_product_price FLOAT
Indexes
orders
order_id INT(11)
order_date DATETIME
order_customer_id INT(11)
order_status VARCHAR(45)
Indexes
customers
customer_id INT(11)
customer_fname VARCHAR(45)
customer_lname VARCHAR(45)
customer_email VARCHAR(45)
customer_password VARCHAR(45)
customer_street VARCHAR(255)
customer_city VARCHAR(45)
customer_state VARCHAR(45)
customer_zipcode VARCHAR(45)
Indexes

Design:


The Project scope is to convert the CSV formatted files into JSON files.
My
MySQL Server
CSV Files
JSON Files
JSON

Setup Instructions

1. Setup the Project Using VSCode
2. Make sure you have set up a virtual environment (creating venv, requirements.txt, etc.,) and installed dependencies for the project.
3. It is essential that you deploy the application with the core logic.
4. Run the project after setting all the environment variables.
5. Take appropriate steps to handle the exception

Validation Steps

- You should check whether the data in the files has been converted properly.
- Make sure the target folder has been created and populated with JSON files and confirm that the schema structure was accurately reflected from the CSV file. (**Hint**: Refer to schemas.json)
- Take the count of records in the CSV files and compare it to the number of records in the JSON files.

```
import pandas as pd
# ###### Read orders JSON File using PANDAS
orders_data_json= pd.read_json(
   'data/retail_db/orders_json/part-00000',
   lines=True
)
```

```
# To find count of rows
orders_data_json.count()
# ###### Read order_items JSON File using PANDAS
order_items_data_json= pd.read_json(
   'data/retail_db/order_items_json/part-00000',
   lines=True
)
# To find count of rows
order_items_data_json.count()
```

Technologies Used

- Programming Language – Python
- Pandas – For Converting CSV to Dataframe and then Dataframe into JSON.

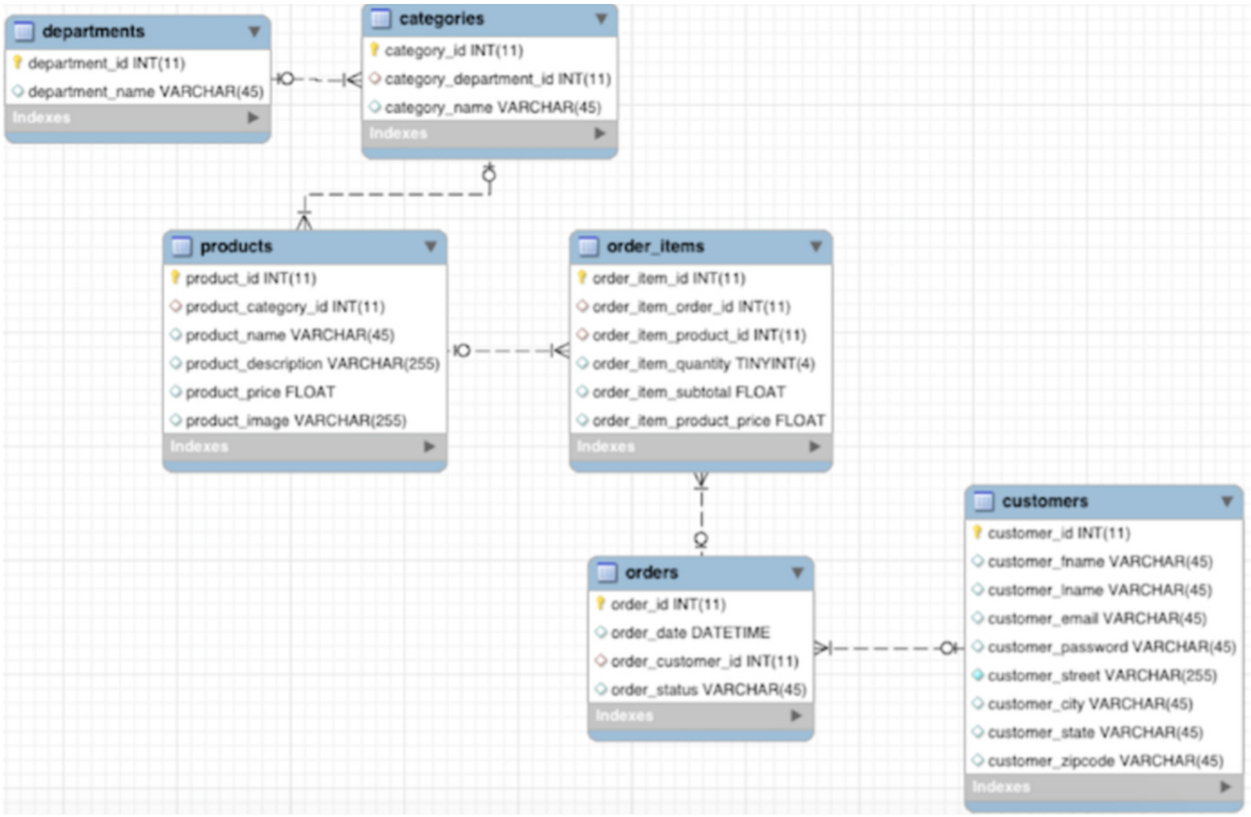
Project 2 - Files To Database Loader Handout

Overview

The objective of this project is to develop solutions based on the design provided. In this case, the source data was obtained in the form of files from a MySQL DB.

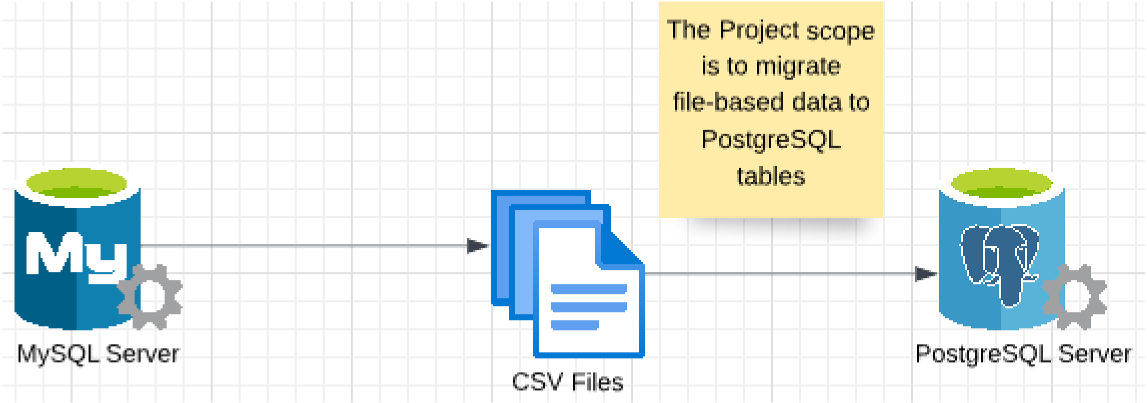
We require the data to be loaded into PostgreSQL, which is a common scenario in companies who want to change underlying database technologies, in which case the data from one DB needs to be migrated into another DB. The Project scope is to migrate file-based data to PostgreSQL tables

Data Model Details



departments
department_id INT(11)
department_name VARCHAR(45)
Indexes
categories
category_id INT(11)
category_department_id INT(11)
category_name VARCHAR(45)
Indexes
products
product_id INT(11)
product_category_id INT(11)
product_name VARCHAR(45)
product_description VARCHAR(255)
product_price FLOAT
product_image VARCHAR(255)
Indexes
order_items
order_item_id INT(11)
order_item_order_id INT(11)
order_item_product_id INT(11)
order_item_quantity TINYINT(4)
order_item_subtotal FLOAT
order_item_product_price FLOAT
Indexes
customers
customer_id INT(11)
customer_fname VARCHAR(45)
customer_lname VARCHAR(45)
customer_email VARCHAR(45)
customer_password VARCHAR(45)
customer_street VARCHAR(255)
customer_city VARCHAR(45)
customer_state VARCHAR(45)
customer_zipcode VARCHAR(45)
Indexes
orders
order_id INT(11)
order_date DATETIME
order_customer_id INT(11)
order_status VARCHAR(45)
Indexes

Design


The Project scope is to migrate file-based data to PostgreSQL tables
MySQL Server
CSV Files
PostgreSQL Server

Setup Instructions

The previous project can be used as a reference for the following steps. Watch the relevant videos to take care of the setup.

1. Setup the Project Using VSCode
2. Make sure you have set up a virtual environment (creating venv, requirements.txt, etc.,) and installed dependencies for the project.
3. It is essential that you deploy the application with the core logic.
4. Be sure to drop the tables and recreate them using the scripts OR simply truncate the tables with the truncate command before running the scripts
5. Run the project after setting all the environment variables.

Validation Steps

- You should check whether the data in the tables have been populated by running queries.
- In postgres tables, we need to confirm that the schema structure (column name, data type, etc.) was accurately reflected from the CSV file. (**Hint**: Refer to schemas.json)
- Take the count of records in the CSV files and compare it to the number of records in the PostgreSQL tables. The count should match the numbers below.

```
select count(*) from orders; --68883
select count(*) from order_items; --172198
select count(*) from categories; --58
select count(*) from customers; --12435
select count(*) from departments; --6
select count(*) from products; --1345
```

Technologies Used

- Programming Language – Python
- Pandas – For Converting CSV to Dataframe and then load the Dataframe into Postgres Database