

# ETL Pipeline Development

## Objective:

Design and implement an ETL pipeline that extracts data from multiple CSV files, applies transformations based on business rules, and loads the data into a PostgreSQL database. The pipeline should also include robust error handling and logging.

Process	Tech Stack Used
ETL Process	Python (including libraries such as pandas, numpy and pycopg2)
SQL Loading	PostgreSQL

## Setup Library & Installation

### Python Libraries

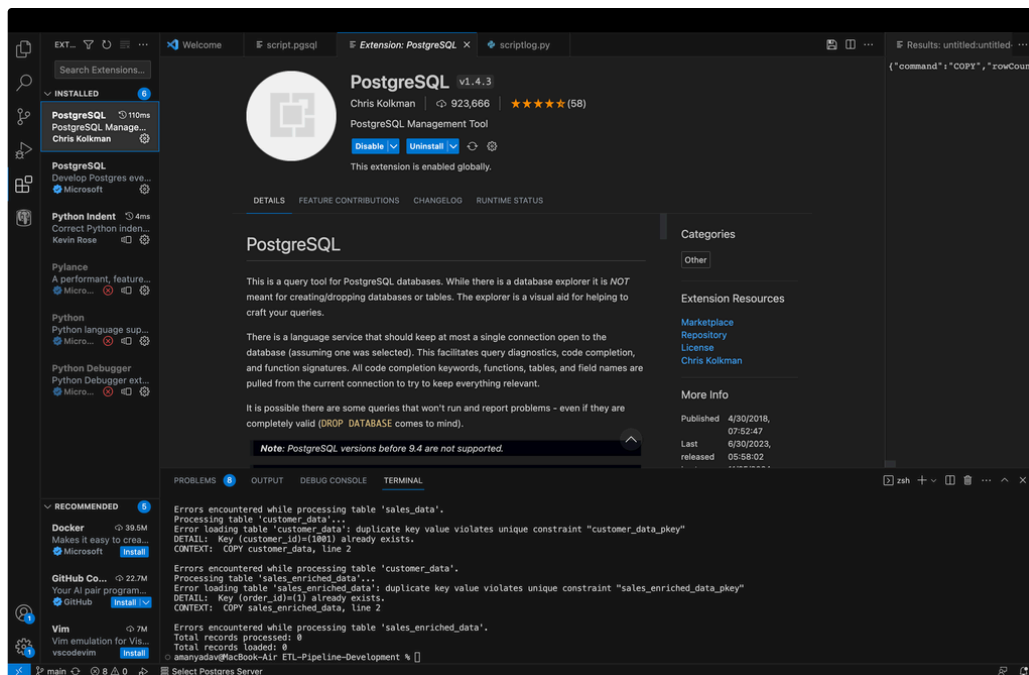
Start by installing python library like pandas, numpy and pycopg2 using the command line or terminal:

1. `pip3 install pandas`
2. `pip3 install numpy`
3. `pip3 install pycopg2`

### PostgreSQL Installation

Download the PostgreSQL by [clicking here](#) and choosing your appropriate OS configuration.

**[Optional]** I have also installed the extension PostgreSQL by ChrisKolkman in VSCode since I'll be connecting the database using the VSCode



## Steps

In the provided document after installing the necessary libraries open the **pipeline.py** file in your preferred code editor.

For demonstration purpose I have used Visual Studio Code. Once the pipeline.py file is open run it through command line/Terminal using `python3 pipeline.py` (Could be different depending on your file path I was already in the project fold so I have used the the provided terminal script)

In the first function in my python file I have extracted and validated the **sales\_data.csv** and **customer\_data.csv** file using the `load_data()` function

In order to store the dataset I have used **pandas dataframe**

```
1 def load_data(file_path):
2     try:
3         # Load the data into a DataFrame
4         data = pd.read_csv(file_path)
5
6         # Log data for debugging
7         #print("\n", data)
8
9         # Validate columns based on file type
10        if 'sales_data.csv' in file_path:
11            missing_columns = [col for col in required_sales_columns if col not in data.columns]
12            if missing_columns:
13                print(f"Error: The following columns are missing in '{file_path}': {'',
14                '.join(missing_columns)}")
15                return None
16            elif 'customer_data.csv' in file_path:
17                missing_columns = [col for col in required_customer_columns if col not in data.columns]
18                if missing_columns:
19                    print(f"Error: The following columns are missing in '{file_path}': {'',
20                    '.join(missing_columns)}")
21                    return None
22            else:
```

```

21         print(f"Error: Unsupported file '{file_path}'")
22         return None
23
24     return data
25
26 except FileNotFoundError:
27     print(f"File {file_path} is missing")
28 except Exception as e:
29     print(f"Error loading data from '{file_path}': {e}")
30
31 return None

```

The screenshot shows a VS Code editor with a file explorer on the left displaying the project structure: ETL-PIPELINE-DEVELOPMENT, .gitattributes, customer\_data.csv, pipeline.py, README.md, and sales\_data.csv. The main editor window shows the pipeline.py script. The terminal at the bottom displays the output of running the script, showing the data loaded from sales\_data.csv and customer\_data.csv, and confirming successful loading and validation for both files.

```

pipeline.py
39     return None
40
41 # List of files to process
42 files = ['sales_data.csv', 'customer_data.csv']
43
44 sales_df = None
45 customer_df = None
46
47 # Process each file
48 for file in files:
49     df = load_data(file)
50     if df is not None:
51         print(f"File Loaded and Validated Successfully: {file}")

```

```

amanyadav@MacBook-Air ETL-Pipeline-Development % python3 pipeline.py
order_id customer_id product quantity price order_date
0 1 1001 Widget A 2 19.99 2024-01-15 10:00
1 2 1002 Widget B 1 29.99 2024-01-16 11:00
2 3 1003 Widget A 1 19.99 2024-01-16 12:00
3 4 1004 Widget C 0 39.99 2024-01-17 14:00
4 5 1005 Widget B 3 29.99 invalid_date
File Loaded and Validated Successfully: sales_data.csv
customer_id customer_name email signup_date
0 1001 John Doe john.doe@example.com 2023-06-10 9:00
1 1002 Jane Smith jane.smith@example.com invalid_date
2 1003 Alice Johnson alice.j@example.com 2022-12-01 14:00
3 1004 Bob Brown bob.brown@example.com 2023-01-12 16:00
4 1005 Charlie White charlie.w@example.com 2023-11-15 11:00
File Loaded and Validated Successfully: customer_data.csv

```

## Extracting and validating the dataset

Once the data has been extracted I have performed transformation on the dataset using numpy and pandas.

### Data Cleaning:

- Converted order\_date and signup\_date to a standard format (YYYY-MM-DD). Log errors for invalid dates and skip affected records.
- Remove any rows with missing or invalid customer\_id or order\_id.

```

45 sales_df = None
46 customer_df = None
47
48 # Process each file
49 for file in files:
50     df = load_data(file)
51     if df is not None:
52         #Print File Loaded and Validated Successfully: (file)\n\n
53         if file == 'sales_data.csv':
54             sales_df = df
55             sales_df['order_date'] = pd.to_datetime(sales_df['order_date'], errors='coerce').dt.date #converting order_date in sales tab
56             sales_df['order_id'] = pd.to_numeric(sales_df['order_id'], errors='coerce')
57             sales_df['customer_id'] = pd.to_numeric(sales_df['customer_id'], errors='coerce')
58             sales_df_clean = sales_df.dropna(subset=['order_id', 'customer_id']) #Remove any invalid customer id and order id
59             print("\n Successfully Formatted Order_Date and Checked for any invalid customer and order id: \n\n", sales_df)
60         if file == 'customer_data.csv':
61             customer_df = df
62             customer_df['signup_date'] = pd.to_datetime(customer_df['signup_date'], errors='coerce').dt.date #converting signup_date in
63             customer_df['customer_id'] = pd.to_numeric(customer_df['customer_id'], errors='coerce')
64             customer_df_clean = customer_df.dropna(subset=['customer_id']) #Remove any invalid customer id
65             print("\n Successfully Formatted Signup_Date and Checked for any invalid customer id: \n\n", customer_df)

```

```

amanyadav@MacBook-Air ETL-Pipeline-Development % python3 pipeline.py
Successfully Formatted Order_Date and Checked for any invalid customer and order id:
order_id customer_id product quantity price order_date
0 1 1001 Widget A 2 19.99 2024-01-15
1 2 1002 Widget B 1 29.99 2024-01-16
2 3 1003 Widget A 1 19.99 2024-01-16
3 4 1004 Widget C 0 39.99 2024-01-17
4 5 1005 Widget B 3 29.99 NaT
Successfully Formatted Signup_Date and Checked for any invalid customer id:
customer_id customer_name john.doe@example.com 2023-01-10
0 1001 John Doe
1 1002 Jane Smith jane.smith@example.com NaT
2 1003 Alice Johnson alice.j@example.com 2022-12-01
3 1004 Bob Brown bob.brown@example.com 2023-01-17
4 1005 Charlie White charlie.w@example.com 2023-11-15

```

Order\_date and signup\_date in YYYY-MM-DD format

Data Enrichment:

- Calculated the total value for each order (total\_value = quantity \* price).
- Joined the sales data with customer data on customer\_id to enrich the sales table with customer\_name and email.

```

48 # Process each file
49 for file in files:
50     df = load_data(file)
51     if df is not None:
52         #Print File Loaded and Validated Successfully: (file)\n\n
53         if file == 'sales_data.csv':
54             sales_df = df
55             sales_df['order_date'] = pd.to_datetime(sales_df['order_date'], errors='coerce').dt.date #converting order_date in sales tab
56             sales_df['order_id'] = pd.to_numeric(sales_df['order_id'], errors='coerce')
57             sales_df['customer_id'] = pd.to_numeric(sales_df['customer_id'], errors='coerce')
58             sales_df_clean = sales_df.dropna(subset=['order_id', 'customer_id']) #Remove any invalid customer id and order id
59             sales_df['total_value'] = sales_df['quantity'] * sales_df['price'] #calculating total value of each order
60             print("\n Successfully Formatted Order_Date and Checked for any invalid customer and order id: \n\n", sales_df)
61             print("\n Tables after data enrichment \n\n", sales_df)
62         if file == 'customer_data.csv':
63             customer_df = df
64             customer_df['signup_date'] = pd.to_datetime(customer_df['signup_date'], errors='coerce').dt.date #converting signup_date in
65             customer_df['customer_id'] = pd.to_numeric(customer_df['customer_id'], errors='coerce')
66             customer_df_clean = customer_df.dropna(subset=['customer_id']) #Remove any invalid customer id
67             print("\n Successfully Formatted Signup_Date and Checked for any invalid customer id: \n\n", customer_df)
68
69 #Join the sales data with customer data on customer_id to enrich the sales table with customer_name and email.
70 if sales_df is not None and customer_df is not None:
71     enriched_sales_df = pd.merge(sales_df, customer_df[['customer_id', 'customer_name', 'email']],
72                                on='customer_id', how='left')
73     print("\n Enriched Sales \n\n", enriched_sales_df)

```

```

amanyadav@MacBook-Air ETL-Pipeline-Development % python3 pipeline.py
Tables after data enrichment
order_id customer_id product quantity price order_date total_value
0 1 1001 Widget A 2 19.99 2024-01-15 39.98
1 2 1002 Widget B 1 29.99 2024-01-16 29.99
2 3 1003 Widget A 1 19.99 2024-01-16 19.99
3 4 1004 Widget C 0 39.99 2024-01-17 0.00
4 5 1005 Widget B 3 29.99 NaT 89.97
Enriched Sales_df
order_id customer_id product quantity price order_date total_value customer_name email
0 1 1001 Widget A 2 19.99 2024-01-15 39.98 John Doe john.doe@example.com
1 2 1002 Widget B 1 29.99 2024-01-16 29.99 Jane Smith jane.smith@example.com
2 3 1003 Widget A 1 19.99 2024-01-16 19.99 Alice Johnson alice.j@example.com
3 4 1004 Widget C 0 39.99 2024-01-17 0.00 Bob Brown bob.brown@example.com
4 5 1005 Widget B 3 29.99 NaT 89.97 Charlie White charlie.w@example.com

```

Data Enrichment

Business Logic:

- Excluded orders with a quantity of 0 or less.
- Marked orders with total values exceeding \$1000 as "High-Value Orders" in a new column order\_type.
- Add a column customer\_tenure representing the number of days since the customer's signup date.

```

68 sales_df['total_value'] = sales_df['quantity'] * sales_df['price'] #Calculating total value of each order
69 #print("\nSales after data enrichment\n\n",sales_df)
70 sales_df = sales_df[sales_df['quantity'] > 0].copy() #excluding quantity with 0 or less
71 sales_df['order_type'] = np.where(sales_df['total_value'] > 1000, 'High-Value Order', 'Regular Order') #Marking any total
72 print("\n Sales Data Table after Business rule\n\n",sales_df)
73 if file == 'customer_data.csv':
74     customer_df = df
75     customer_df['signup_date'] = pd.to_datetime(customer_df['signup_date'], errors='coerce').dt.date #converting signup_date
76     customer_df['customer_id'] = pd.to_numeric(customer_df['customer_id'], errors='coerce')
77     customer_df_clean = customer_df.dropna(subset=['customer_id']) #Remove any invalid customer id
78     #print("\n Successfully Formatted Signup Date and Checked for any invalid customer id: \n\n",customer_df)
79     customer_df['signup_date'] = pd.to_datetime(customer_df['signup_date'])
80     current_date = pd.to_datetime(datetime.now().date())
81     customer_df['customer_tenure'] = (current_date - customer_df['signup_date']).dt.days
82     print("\nCustomer Data after Business rule\n\n",customer_df)
83
84 #Join the sales data with customer data on customer_id to enrich the sales table with customer_name and email.
85 if sales_df is not None and customer_df is not None:
86     enriched_sales_df = pd.merge(sales_df, customer_df[['customer_id', 'customer_name', 'email']],
87                                on='customer_id', how='left')
88     #print("\nEnriched Sales_df\n\n",enriched_sales_df)

```

```

Sales Data Table after Business rule
order_id customer_id product quantity price order_date total_value order_type
0 1 1001 Widget A 2 19.99 2024-01-15 39.98 Regular Order
1 2 1002 Widget B 1 29.99 2024-01-16 29.99 Regular Order
2 3 1003 Widget A 1 19.99 2024-01-16 19.99 Regular Order
4 5 1005 Widget B 3 29.99 NaN 89.97 Regular Order

Customer Data after Business rule
customer_id customer_name email signup_date customer_tenure
0 1001 John Doe john.doe@example.com 2023-06-10 534.0
1 1002 Jane Smith jane.smith@example.com NaN NaN
2 1003 Alice Johnson alice.j@example.com 2022-12-01 725.0
3 1004 Bob Brown bob.brown@example.com 2023-01-17 676.0
4 1005 Charlie White charlie.w@example.com 2023-11-15 376.0

```

## Business Rule

Data Aggregation:

-Create a summary table with total sales (SUM(total\_value)) per product and order counts (COUNT(order\_id)).

```

81 #Summary table with total sales and per product and order count
82 summary_table = enriched_sales_df.groupby('product').agg(
83     total_sales = ('total_value', 'sum'),
84     order_count = ('order_id', 'count')
85     ).reset_index()
86 print("\nSummary table with total sales per product and order counts:\n\n",summary_table)

```

```

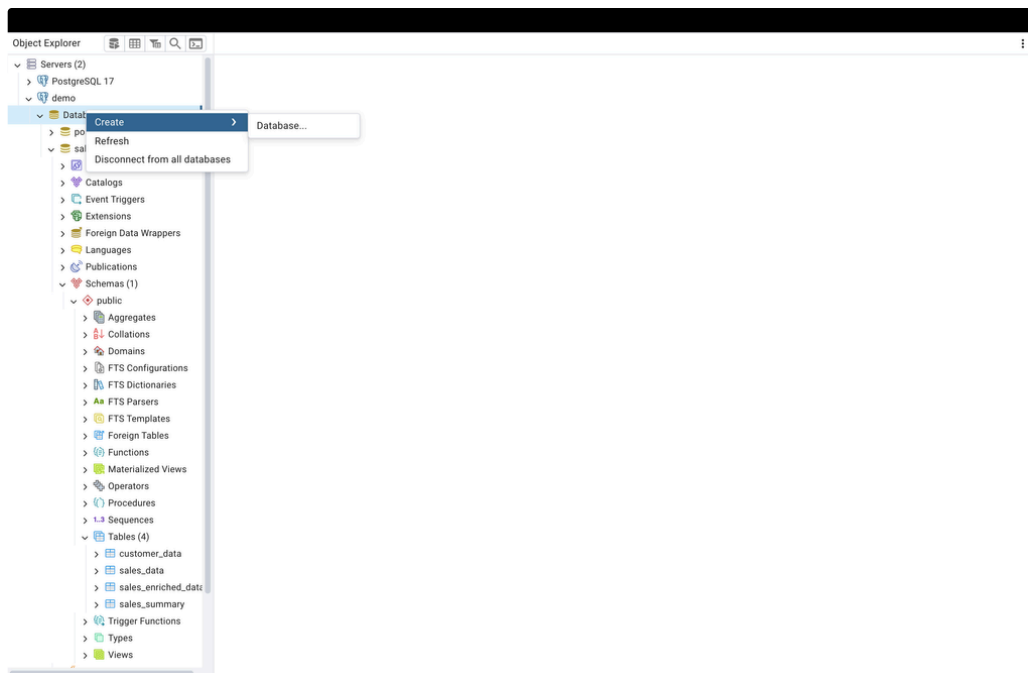
Summary table with total sales per product and order counts:
product total_sales order_count
0 Widget A 59.97 2
1 Widget B 119.96 2

```

## Data Aggregation

Once the transformation has performed, as I already install PostgreSQL in my VSCode I will be using that to make the connection between my dataset.

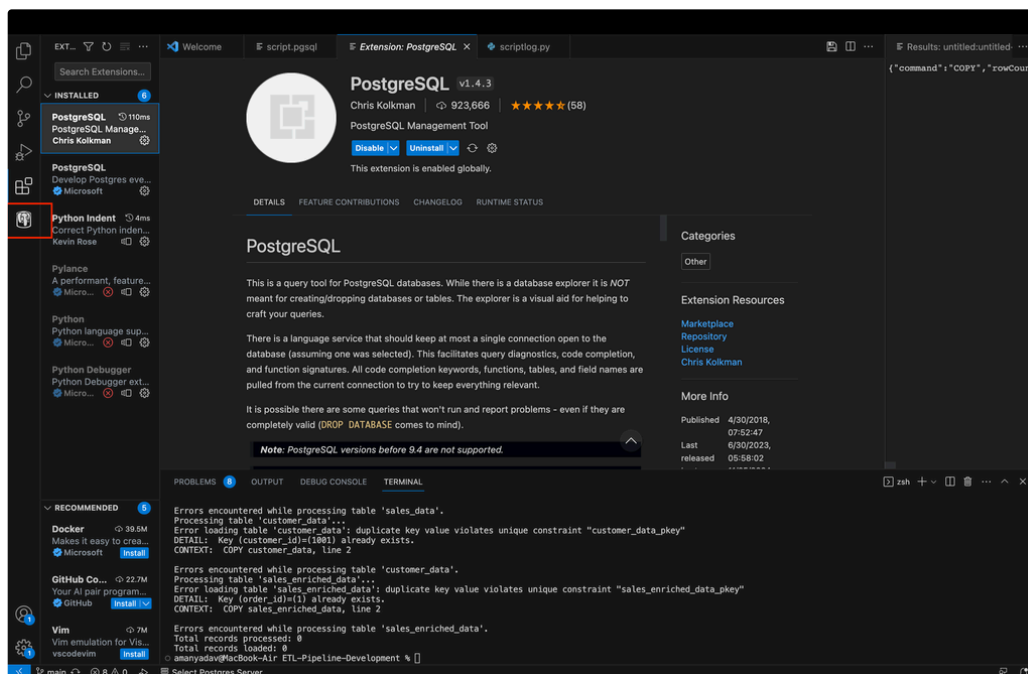
First we will create a database using PgAdmin 4.



Creating database

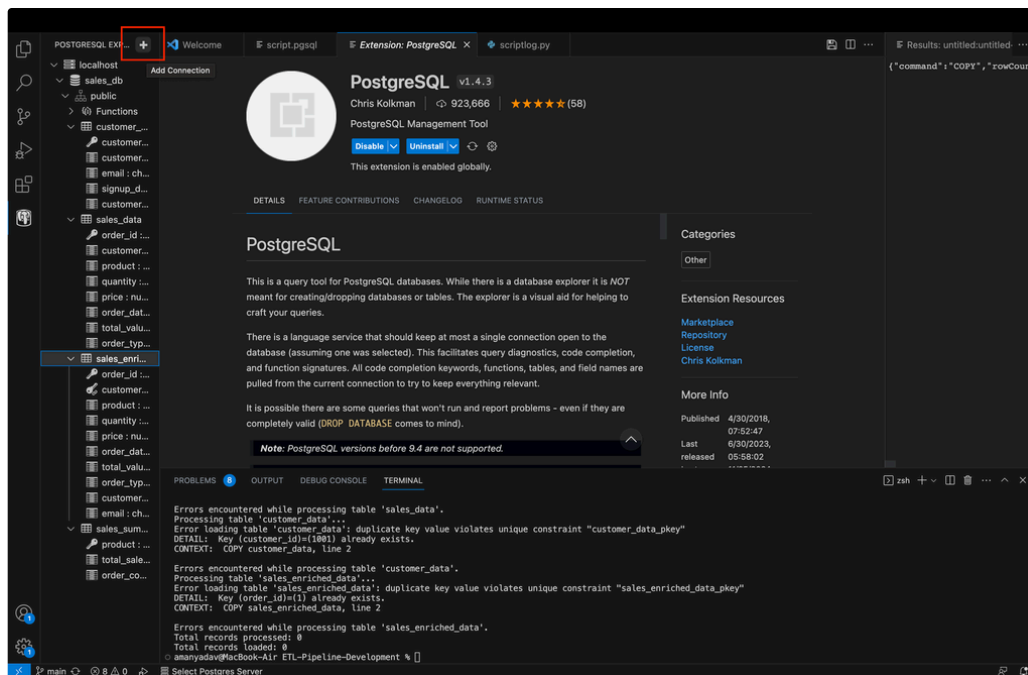
I already have created a database in name of sales\_db.

Now back to VS Code by clicking on the PostgreSQL logo in the VSCode I will make the database connection



Creating connection

Once the PostgreSQL tab is open click on the "+" and choose your server name → enter your username → password → select database



After this I've created the PostgreSQL script to create table

```

1  -- Create the sales table
2  CREATE TABLE sales_data (
3      order_id SERIAL PRIMARY KEY,
4      customer_id INT NOT NULL,
5      product VARCHAR(255) NOT NULL,
6      quantity INT NOT NULL,
7      price DECIMAL(10, 2) NOT NULL,
8      order_date DATE,
9      total_value DECIMAL(10, 2) NOT NULL,
10     order_type VARCHAR(50)
11 );
12
13
14 -- Create the sales_summary table
15 CREATE TABLE sales_summary (
16     product VARCHAR(255) PRIMARY KEY,
17     total_sales DECIMAL(10, 2) NOT NULL,
18     order_count INT NOT NULL
19 );
20
21 --Create customer_data table
22 CREATE TABLE customer_data (
23     customer_id SERIAL PRIMARY KEY,
24     customer_name VARCHAR(255),
25     email VARCHAR(255),
26     signup_date DATE,
27     customer_tenure VARCHAR(255) DEFAULT NULL
28 );
29
30 --Create the sales_enriched_data
31 CREATE TABLE sales_enriched_data (
32     order_id SERIAL PRIMARY KEY,
33     customer_id INT NOT NULL,
34     product VARCHAR(255) NOT NULL,
35     quantity INT NOT NULL,

```

```

36 price DECIMAL(10, 2) NOT NULL,
37 order_date DATE,
38 total_value DECIMAL(10, 2) NOT NULL,
39 order_type VARCHAR(50),
40 customer_name VARCHAR(255),
41 email VARCHAR(255),
42 CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES customer_data(customer_id) ON DELETE CASCADE
43 );

```

Once the table has created I have used copy command to copy the transformed dataset to load in the table which is created.

```

1 --Copied the data from the csv file to our table
2 COPY sales_summary FROM '/private/tmp/summary_table.csv'
3 DELIMITER ','
4 CSV HEADER
5 ENCODING 'UTF8';
6
7 COPY sales_data FROM '/private/tmp/transformed_sales_data.csv'
8 DELIMITER ','
9 CSV HEADER
10 ENCODING 'UTF8';
11
12 COPY customer_data FROM '/private/tmp/transformed_customer_data.csv'
13 DELIMITER ','
14 CSV HEADER
15 ENCODING 'UTF8';
16
17 COPY sales_enriched_data FROM '/private/tmp/enriched_sales_df.csv'
18 DELIMITER ','
19 CSV HEADER
20 ENCODING 'UTF8';

```

Once the transferred has performed I have run the query to view the columns

The screenshot shows a PostgreSQL IDE interface. The top pane displays a query result with 2 rows returned:

product	total_sales	order_count
Widget A	59.97	2
Widget B	119.96	2

The bottom pane shows error messages:

```

Errors encountered while processing table 'sales_data'.
Processing table 'customer_data'...
Error loading table 'customer_data': duplicate key value violates unique constraint "customer_data_pkey"
DETAIL: Key (customer_id)=(1001) already exists.
CONTEXT: COPY customer_data, line 2

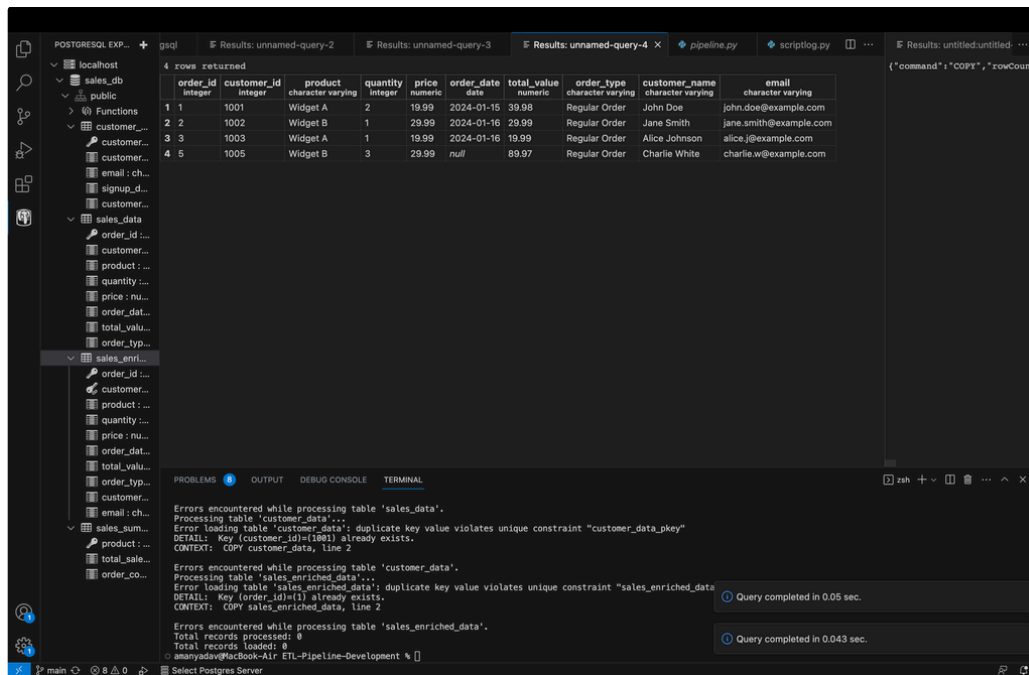
Errors encountered while processing table 'customer_data'.
Processing table 'sales_enriched_data'...
Error loading table 'sales_enriched_data': duplicate key value violates unique constraint "sales_enriched_data_pkey"
DETAIL: Key (order_id)=(1) already exists.
CONTEXT: COPY sales_enriched_data, line 2

Errors encountered while processing table 'sales_enriched_data'.
Total records processed: 0
Total records loaded: 0
Query completed in 0.026 sec.

```

Summary Table





Sales Data Table

Challenges/Errors Faced	Resolution
Was having issue where column was missing/mismatch in sales_data.csv	Checked the error in the output terminal turned out to there was typo while creating the original dataset file
While validating the columns, columns were bring crossed check with both the dataset meaning load_data() function was also checking for missing sales_data columns in the customer_data and vice-versa	Implemented if condition to avoid that error.
While formatting date into YYYY-MM-DD format was unable to format the invalid_date.	Used <code>errors='coerce'</code> while converting date into YYYY-MM-DD Format so that invalid data will be set to NaN
While copying the dataset to PostgreSQL was having read access permission even though there was read permission given	<ol style="list-style-type: none"> <li>1. moved the transformed file into tmp folder and then copied the data.</li> <li>2. Directly exported from PgAdmin 4.</li> </ol>