End to end project with sql, python and postgresql, could also extend this with powerbi

Tech stack: vcscode, postgresql, powebi, kaggle
Kaggle dataset: https://www.kaggle.com/datasets/najir0123/walmart-10k-sales-datasets

Downloading the dataset from kaggle using API
Steps:

Setup python environment with vcscode , c

- create project folder and open vscode , open folder in vscode
- Toggle terminal and type cd for current directory

Download kaggle dataset (json file)

- Download API kaggle from settings
- Create .kaggle folder in home directory
- Drag json kaggle file to .kaggle
-

Creating python environment in vscstudio

Return to vscode terminal to set up python environment

- python -m venv myenv1
- Python –version : checks what version of python is installed
- Kill terminal
- Add project.ipynb to add jupyter notebook file
- print("hello world") and select kernel which will suggest the python environment that you created (myenv1)
- **Now if you open terminal you see it has changed to myenv1? Did not see this**
- Pip install kaggle in terminal
- Pip list to verify all libraries
- kaggle datasets download -d najir0123/walmart-10k-sales-datasets : this downloads the dataset from the username and link from the actual link of the kaggle dataset
- Expand-Archive walmart-10k-sales-datasets.zip : unzips the file to csv
- Look in excel to analyze dataset , has 10,000 records . need to remove dollar sign and create new column for total unit price . will also use pandas to check for duplicates and missing values

Data Exploration using Pandas - process data, clean data, creating new columns

Goals: use pandas to remove $ sign in csv dataset (will not work with sql), create new column for total unit price (unit price * quantity purchased), and remove missing values / duplicates

Steps:

- Create new file called requirements.txt and add “pandas” line - this file will include all libraries that we download
- TERMINAL: pip install pandas
- Python : import pandas as pd

```
df = pd.read_csv('Walmart.csv', encoding_errors='ignore')

df.shape
```

Challenges overcame/insights: the walmart.csv file was not moved into the walmart project dataset so when I tried executing the first line it said ‘walmart.csv’ not found. I moved it into the walmart project set and it successfully created the dataframe. The dataset has (10051 rows, 11 columns)

```
df.head()
```

Gave us the headers and columns

```
df.describe()
```

Gave us summary statistics of the dataset - revealed that this only considers 4 columns as number columns (invoice_ id , quantity, rating, and profit_margin) - this also has quantity off when looking at count (10020 compared to 10051). Also unit price should be considered a number data but its not

	invoice_id	quantity	rating	profit_margin
count	10051.000000	10020.000000	10051.000000	10051.000000
mean	5025.741220	2.353493	5.825659	0.393791
std	2901.174372	1.602658	1.763991	0.090669
min	1.000000	1.000000	3.000000	0.180000
25%	2513.500000	1.000000	4.000000	0.330000
50%	5026.000000	2.000000	6.000000	0.330000
75%	7538.500000	3.000000	7.000000	0.480000
max	10000.000000	10.000000	10.000000	0.570000

```
df.info()
```

Revealed that unit price and quantity are showing 10020 non - null compared to all other columns showing 10051, they are missing values. Also unit price is considered an object.

```
df.duplicated().sum()
```

Returned 51 duplicate values (all in dataframe)

```
df.isnull().sum()
```
Revealed that unit price and quantity have 31 nulls each.

```
df.drop_duplicates(inplace = True)
```
Deleted duplicates
```
df.shape
```
Dataset now is (10,000, 11)

```
#dropping all rows with missing records
df.dropna(inplace=True)
df.isnull().sum()
```
There are now no missing values or duplicates in the dataset

```
df.shape
```
Shape has again reduced to (9969, 11)

```
df.dtypes
```
Shows that unit price is object, we need to focus on this now

```
df['unit_price'].astype(float)
ValueError: could not convert string to float: '$74.69'
```

SITUATION: Cannot convert unit price column to a float value because of the $
TASK: Figure out how to replace the $ in the column with a different value
ACTION:

```
df['unit_price'] = df['unit_price'].str.replace('$', '').astype(float)
```

Replaces $ with nothing, also changes column to float, refer back to the original column as well

Result

```
df.head()
```
✓ 0.0s

	invoice_id	Branch	City	category	unit_price	quantity	date	time	payment_method	rating	profit_margin
0	1	WALM003	San Antonio	Health and beauty	74.69	7.0	05/01/19	13:08:00	Ewallet	9.1	0.48
1	2	WALM048	Harlingen	Electronic accessories	15.28	5.0	08/03/19	10:29:00	Cash	9.6	0.48
2	3	WALM067	Haltom City	Home and lifestyle	46.33	7.0	03/03/19	13:23:00	Credit card	7.4	0.33
3	4	WALM064	Bedford	Health and beauty	58.22	8.0	27/01/19	20:33:00	Ewallet	8.4	0.33
4	5	WALM013	Irving	Sports and travel	86.31	7.0	08/02/19	10:37:00	Ewallet	5.3	0.48

Also with df.dtypes() we can see it changed to float

Now we are creating the unit price times quantity column

```
df['total'] = df['unit_price'] * df['quantity']
```

```
df.head()
```
✓ 0.0s

	invoice_id	Branch	City	category	unit_price	quantity	date	time	payment_method	rating	profit_margin	total
0	1	WALM003	San Antonio	Health and beauty	74.69	7.0	05/01/19	13:08:00	Ewallet	9.1	0.48	522.83
1	2	WALM048	Harlingen	Electronic accessories	15.28	5.0	08/03/19	10:29:00	Cash	9.6	0.48	76.40
2	3	WALM067	Haltom City	Home and lifestyle	46.33	7.0	03/03/19	13:23:00	Credit card	7.4	0.33	324.31
3	4	WALM064	Bedford	Health and beauty	58.22	8.0	27/01/19	20:33:00	Ewallet	8.4	0.33	465.76
4	5	WALM013	Irving	Sports and travel	86.31	7.0	08/02/19	10:37:00	Ewallet	5.3	0.48	604.17

This new total column takes the quantity times the unit_price and lines it up as “total”

Exporting the data to postgresql (pgadmin4)
Steps:

- Add sql toolkit to dependencies and install with terminal
- pymysql, sqlalchemy create_engine, psycopg2 (postgresql only) - pip install in terminal. Verify using pip list, everything is installed
- Import in project as well
- Add to requirements.txt

```
#mysqltoolkit
import pymysql #this will work as adapter
from sqlalchemy import create_engine
import psycopg2 #postgresql only
```

```
df.to_csv('walmart_clean_data.csv', index = False)
```

Directly jump to the sql project now without going through all of this python

Create walmart db in pgadmin 4
and used the following code to connect to pgadmin4

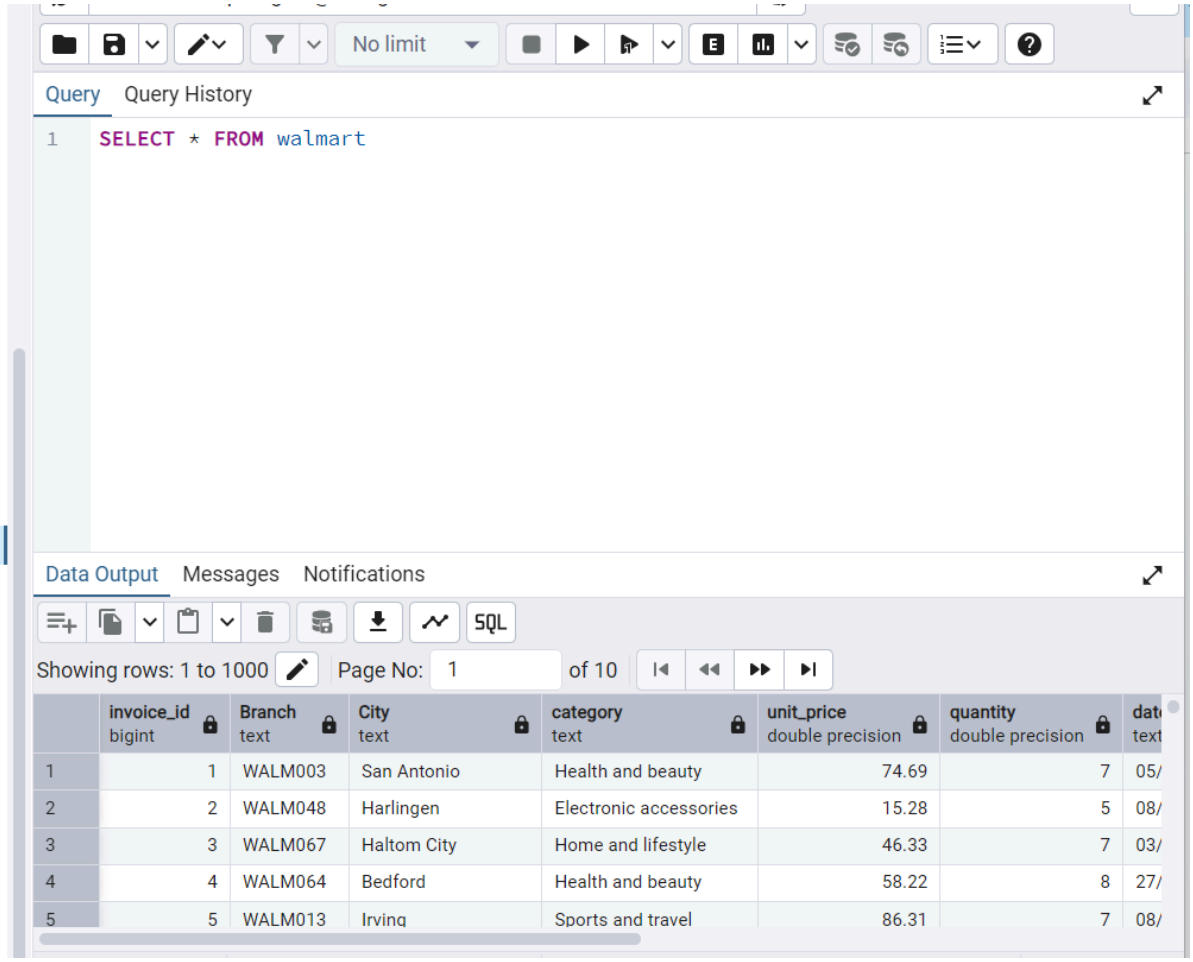
```
engine_psql =
create_engine("postgresql+psycopg2://postgres:Columdos1234%40@localhost:54
32/walmart_db")

try:
    engine_psql
    print("Connection Success")
except:
    print("Conenction Failed")
```

```
df.to_sql(name='walmart', con=engine_psql, if_exists='append',
index=False)
```

The top code wasnt working for a while because my password to postgresql contained an ‘@’ symbol and had the following error : OperationalError: (psycopg2.OperationalError) connection to server on socket "@localhost/.s.PGSQL.5432" failed: Invalid argument (0x00002726/10022)

I used AI and it suggested using an ‘%40’ to replace the ‘@ symbol’

SELECT ALL FROM WALMART IN PG ADMIN 4 AND IT IS SUCCESSFULLY CONNECTED!!



NOW WE CAN ANALYZE DATA IN SQL AND SOLVE BUSINESS PROBLEMS

```
SELECT * FROM walmart;
```

```
SELECT
	payment_method,
	COUNT(*)
FROM walmart
	GROUP BY 1;
```

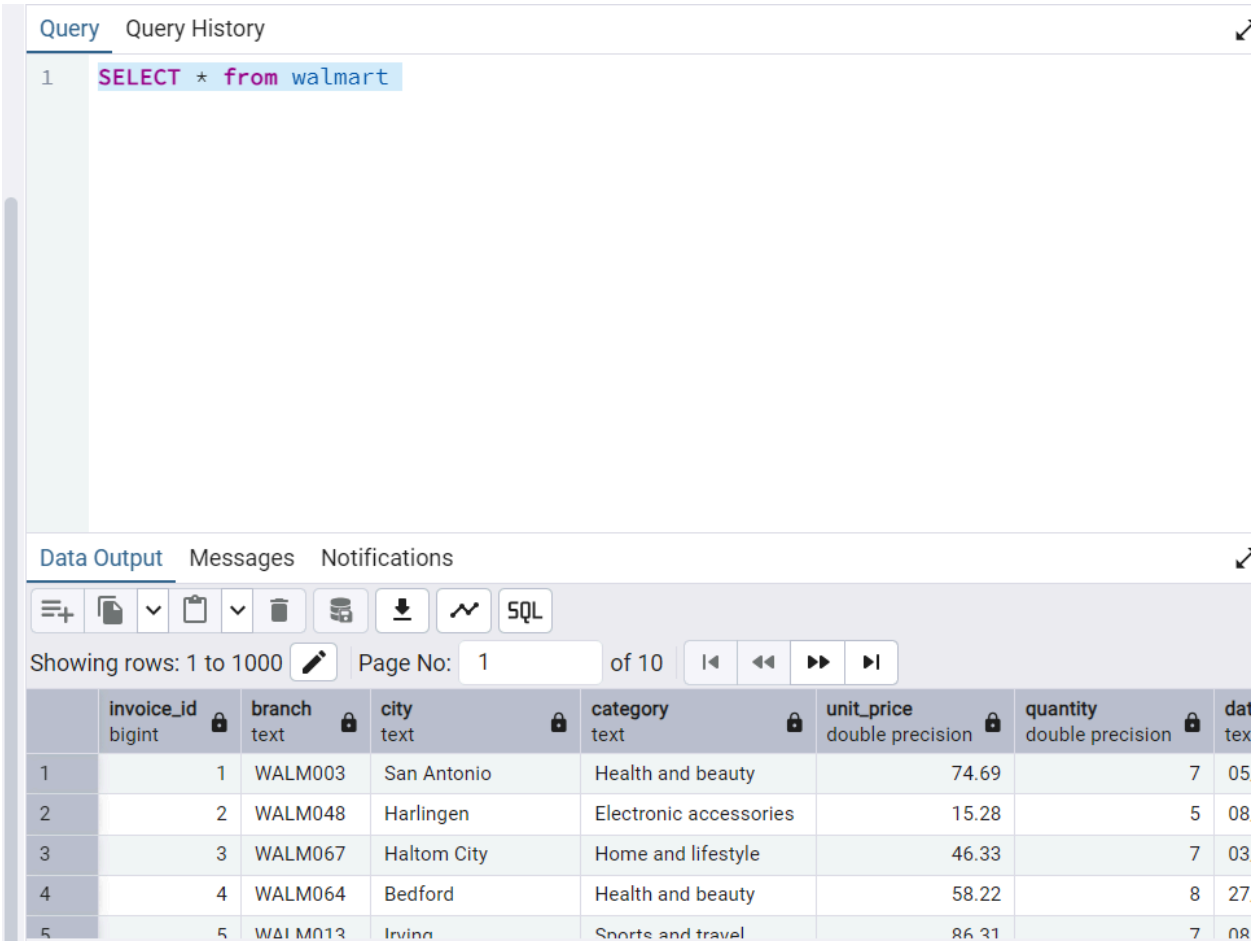
```
SELECT COUNT(DISTINCT branch);
```

```
--branch does not exist because columns need to be in lowercase. need to
drop walmart table and fix in vscstudio and  then export it back--
```

```
DROP TABLE walmart;
```

```
df.columns = df.columns.str.lower()
df.columns
```

This resolves the issue, run all in vsc studio and refresh pgadmin 4 . resolved



	invoice_id bigint	branch text	city text	category text	unit_price double precision	quantity double precision	dat tex
1	1	WALM003	San Antonio	Health and beauty	74.69	7	05
2	2	WALM048	Harlingen	Electronic accessories	15.28	5	08
3	3	WALM067	Haltom City	Home and lifestyle	46.33	7	03
4	4	WALM064	Bedford	Health and beauty	58.22	8	27
5	5	WALM013	Irving	Sports and travel	86.31	7	08

CONDUCTED EDA IN PGADMIN 4 , UP TO QUESTION 1 ON VIDEO

```
SELECT * from walmart;


--Exploratory Data Analaysis--

SELECT COUNT(*)
FROM walmart;

--returns 9969 total records in this dataset--

SELECT COUNT(DISTINCT payment_method)
```

```
FROM walmart;

--ouput shows us that there are 3 distinct payment methods--

SELECT payment_method,
            COUNT(*)
FROM walmart
GROUP BY 1;

--credit card has 4256 transactions , ewallet has 3881 transactions, cash
has 1832 transactions--

SELECT COUNT(DISTINCT branch)
FROM walmart;

--output shows 100 distinct branches in this dataset--

SELECT MAX(quantity) FROM walmart;

--max quantity is 10--

SELECT MIN(quantity) FROM walmart;

--min quantity is 1--

SELECT DISTINCT(category)
FROM walmart;

--output shows us that categories are fashion, electronics, health and
beauty, food/beverages, sports/travel, home/lifestyle

SELECT COUNT(DISTINCT city)
FROM walmart;

--there are 98 different cities--


--find distinct payment methods and number of quantity sold--

SELECT
payment_method,
COUNT(*) as no_payments,
SUM(quantity) as no_qty_sold
FROM walmart
GROUP BY 1
```

```
--credit card has 4256;9567, ewallet has 3881;8932, cash has 1832;4984--

--highest rated category in each branch, rating, category--
--avg rating--subquery and window function--

SELECT * FROM

(SELECT branch,
category,
AVG(rating) as avg_rating,
RANK() OVER( PARTITION BY branch ORDER BY AVG(rating) DESC)
FROM walmart
GROUP BY 1,2
)
WHERE rank = 1


--busiest day for each branch based on number of transactions--cte

WITH date_cte AS(
SELECT
      branch,
      TO_CHAR(TO_DATE(date, 'DD/MM/YY'), 'Day') as day_name,
      COUNT(*) as no_transactions,
      RANK() OVER(PARTITION BY branch ORDER BY(COUNT(*)) DESC) as rank
FROM walmart
GROUP BY 1,2
ORDER BY 1, 3 DESC
)

SELECT *
FROM date_cte
WHERE rank = 1


--quantity of items sold per payment method--

SELECT payment_method,
SUM(quantity) as total_qty_sold
FROM walmart
GROUP BY 1

--avg, minimum, maximum rating of each category per city--
```

```
SELECT city,
category,
MAX(rating) as max_rating,
MIN(rating) as min_rating,
AVG(rating) as avg_rating
FROM walmart
GROUP BY 1, 2
ORDER BY 1

--total revenue and profit for each category--

SELECT
category,
SUM(total) as total_revenue,
SUM(total * profit_margin) as profit
FROM walmart
GROUP BY 1

--most common payment method for each branch, cte and window function--

WITH payment_method_cte AS

(SELECT
branch,
payment_method,
RANK() OVER(PARTITION BY branch ORDER BY COUNT(*) DESC) as
preferred_payment_method
FROM walmart
GROUP BY 1,2
)
SELECT * FROM
payment_method_cte
WHERE preferred_payment_method = 1

--create morning,afternoon,night categories to better segment sales --

ALTER TABLE walmart
ADD COLUMN time_of_day VARCHAR(15)

--added time of day column--

UPDATE walmart
SET time_of_day=
CASE
      WHEN EXTRACT(HOUR FROM (time::time)) < 12 THEN 'Morning'
```

```
      WHEN EXTRACT(HOUR FROM (time::time)) BETWEEN 12 AND 17 THEN
'Afternoon'
      ELSE 'Evening'
END;

--used case statement and added segmented column based on time_of_day with
morning, afternoon, evening--

--END OF SQL--
```

SQL PORTION COMPLETE - CONDUCTED EDA, ADDED TIME OF DAY COLUMN, WINDOW FUNCTIONS AND CTE EXAMPLES.

POWER BI

- ADDED SLICERS FOR TIME OF DAY AND STATE
- ADDED TITLE CARD THAT DISPLAYS CURRENT STATE SELECTED
- ADDED PIE CHART THAT SHOWS TOTAL SPENT PER PAYMENT TYPE
- ADDED COLUMN CHART THAT SHOWS TOTAL REVENUE PER CATEGORY
- ISSUE : TRIED TO MAKE LINE CHART BASED ON DATE BUT THERE ARE TOO MANY DATE VALUES (1/1-3/30) FOR IT TO BE IMPACTFUL. GOING TO POWER QUERY EDITOR TO FIX
- POWER QUERY EDITOR ISSUE - DATE IMPORTED IS IN DD-MM-YYYY FORMAT AND IS A TEXT DATA TYPE, CANNOT FORMAT IT TO MM-DD-YYYY BECAUSE IT CANNOT BE DETECTED. USED SPLIT BY DELIMITER TO SPLIT EACH DAY, MONTH, YEAR INTO A DIFFERENT ROW AND RENAMED ROWS, DAY, MONTH, YEAR
- REARRANGED COLUMNS AND MOVED MONTH BEFORE YEAR
- MERGED COLUMNS WITH “-” AS SEPARATOR AND DETECTED IT AS A DATE TYPE. RESOLVED ISSUE
- ADDED COLUMN BASED ON THE FORMATTED DATE COLUMN “MONTH NAME” IN POWER QUERY EDITOR , ALSO ADDED A YEAR COLUMN BASED ON YEAR- ISSUE RESOLVED
- Added line graph average rating by month and added slicer where you can filter by year,