**Data Loading & Exploration in Vscode using python Virtual environment and getting dataset from Kaggle using API Credentials**

1. import pandas as pd

import psycopg2

from sqlalchemy import create\_engine

1. print(pd.\_\_version\_\_)

O/P : 2.2.3

1. df=pd.read\_csv(**'Walmart.csv'**, encoding\_errors=**'ignore'**)

df.shape

O/P : (10051, 11)

1. df.head()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Invoive\_id | Branch | City | Category | Unit\_price | Quantity | Date | Time | Payment\_method | rating | Profit\_margin |
| 1 | WALM003 | San Antonio | Health and beauty | $74.69 | 7.0 | 05/01/19 | 13:08:00 | Ewallet | 9.1 | 0.48 |
| 2 | WALM048 | Harlingen | Electronic accessories | $15.28 | 5.0 | 08/03/19 | 10:29:00 | Cash | 9.6 | 0.48 |
| 3 | WALM067 | Haltom City | Home and lifestyle | $46.33 | 7.0 | 03/03/19 | 13:23:00 | Credit card | 7.4 | 0.33 |
| 4 | WALM064 | Bedford | Health and beauty | $58.22 | 8.0 | 27/01/19 | 20:33:00 | Ewallet | 8.4 | 0.33 |
| 5 | WALM013 | Irving | Sports and travel | $86.31 | 7.0 | 08/02/19 | 10:37:00 | Ewallet | 5.3 | 0.48 |

1. df.describe()

|  | **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 |

1. df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10051 entries, 0 to 10050

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 invoice\_id 10051 non-null int64

1 Branch 10051 non-null object

2 City 10051 non-null object

3 category 10051 non-null object

4 unit\_price 10020 non-null object

5 quantity 10020 non-null float64

6 date 10051 non-null object

7 time 10051 non-null object

8 payment\_method 10051 non-null object

9 rating 10051 non-null float64

10 profit\_margin 10051 non-null float64

dtypes: float64(3), int64(1), object(7)

memory usage: 863.9+ KB

1. df.duplicated().sum()

O/P : np.int64(51) // contain 51 duplicates

1. df.drop\_duplicates(inplace=True) //removing duplicates

df.duplicated().sum() //summation od duplicate if exists

O/P : np.int64(0) //duplicate removed

1. df.shape

O/P : (10000, 11) // 51 rows reduce

1. df.isnull().sum() //checking if there is any null

O/P : invoice\_id 0

Branch 0

City 0

category 0

unit\_price 31 // 31 null value found

quantity 31 // 31 null value found

date 0

time 0

payment\_method 0

rating 0

profit\_margin 0

dtype: int64

1. df.dropna(inplace=True) // removing null rows

df.isnull().sum() //again checking if null value removed

O/P : invoice\_id 0

Branch 0

City 0

category 0

unit\_price 0 // 0 null value found

quantity 0 // 0 null value found

date 0

time 0

payment\_method 0

rating 0

profit\_margin 0

dtype: int64

1. df.dtypes

O/P : invoice\_id int64

Branch object

City object

category object

unit\_price object //unit is numeric but contain object types bcoz of $ sign so it need to removed

quantity float64

date object

time object

payment\_method object

rating float64

profit\_margin float64

dtype: object

1. df[**'unit\_price'**]=df[**'unit\_price'**].str.replace(**'$',''**).astype(float) //removing and converting

df.info()

<class 'pandas.core.frame.DataFrame'>

Index: 9969 entries, 0 to 9999

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 invoice\_id 9969 non-null int64

1 Branch 9969 non-null object

2 City 9969 non-null object

3 category 9969 non-null object

4 unit\_price 9969 non-null float64 // converted to float

5 quantity 9969 non-null float64

6 date 9969 non-null object

7 time 9969 non-null object

8 payment\_method 9969 non-null object

9 rating 9969 non-null float64

10 profit\_margin 9969 non-null float64

dtypes: float64(4), int64(1), object(6) memory usage: 934.6+ KB

1. df.columns

O/P: Index(['invoice\_id', 'Branch', 'City', 'category', 'unit\_price', 'quantity', 'date', 'time', 'payment\_method', 'rating', 'profit\_margin'],dtype='object')

1. df[**'Total'**]=df[**'unit\_price'**]df[**'quantity'**] //getting total sales

df.dtypes

O/P : invoice\_id int64

Branch object

City object

category object

unit\_price float64

quantity float64

date object

time object

payment\_method object

rating float64

profit\_margin float64

Total float64

dtype: object

# #pgsql requirement

# #host=localhost

# #port= 5432

# #username=  postgres

# #password =

1. df.shape

O/P : (9969, 12) // Total rows after cleaning

1. Pip install kaggle

Import kaggle //installing and importing

Kaggle datasets download -d /path // run in terminal

// after that downloaded file need to be unzipped

1. df.to\_csv(**'walmart\_clean\_datsset.csv'**,index=False) // getting cleaned dataset in csv
2. **help**(df.to\_sql) //getting help

**help**(create\_engine)

1. pgsql\_engine = create\_engine("**postgresql+psycopg2://postgres:ankit@localhost:5432/walmart\_db**") //connecting with SQL, create\_enginge is from sqlalchemy library

**try**:

    pgsql\_engine

**print**("**connected succesfully**")

**except**:

**print**("**Unable to connect**")

1. df.to\_sql(**name**=**'walmart'**, **con**=pgsql\_engine, **if\_exists=**'**append**', **index**=False)

//loading data to sql

// Before making connection with sql, create a Database ‘walmart\_db’ then connect

**// Solving business Problem in SQL**

**--1. find the different payment method, no of transaction and number of qty sold**

**select**

payment\_method,

**count**(\*) **as** paycount,

**sum(**quantity) **as** qtysold

**from** walmart

**group** **by** 1

**-- 2. Identify the highest rated category in each branch, displaying the branch, category, avg rating**

**select** \* **from** (**select**

branch,

category,

**avg**(rating) **as** avg\_rating,

**rank**() **over**(**partition** **by** branch order by **avg**(rating) **desc**) **as** rating

**from** walmart

**group** **by** 1,2 )

**where** rating =1

**-- 3. Identify the busiest day for each branch based on the number of transaction**

**select** \* **from** (**select**

branch,

to\_char(to\_date(date,**'DD/MM/YY'),'day'**),

**count**(\*) **as** no\_of\_transaction,

**rank()** **over**(**partition** **by** branch **order** **by** **count**(\*) **desc**) **as** ranking

**from** walmart

**group by** 1,2)

**where** ranking =1

**-- 4. calculate the total quantity of items sold per payment method.**

**-- list payment method and total quantity**

**select**

payment\_method,

**sum(**quantity) **as** Total\_quantity

**from** walmart

**group by** 1

**-- 5. Determine the avg, min, and max rating of product for each city.**

**select**

"City",

**avg(**rating) **as** avg\_rating,

**min**(rating) **as** min\_rating,

**max**(rating) **as** max\_rating

**from** walmart

**group by** 1

**-- 6. calculate the total profit for each category by considering total profit as (unit \* qty \* profit\_margin])**

**select**

category,

**round**(**sum**("Total")::**numeric**,2)

**from** walmart

**group by** 1

**-- 7. Determine the most common payment method for each branch.**

**-- display branch and preferred payment method**

**select**

branch, payment\_method

**from** (**select**

branch,

payment\_method,

**count**(\*),

**RANK**() **over**(**partition** **by** branch **order** **by** **count**(\*) **desc**) **as** ranking

**from** walmart

**group** **by** 1,2)

**where** ranking =1

**-- 8. Categories sales into three group morning, evening, afternoon.**

**-- find out each of the shift and number of transcation**

**select**

(**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**) **as** shift,

**count**(\*)

**from** walmart

**group by** 1

**-- 9. identify 5 branch with highest decrease ratio in revenue compare to last year**

**-- current year 2023 vs last year 2022**

**select** **branch**, ((revenue2022-revenue2023)::**numeric**/revenue2022::**numeric**\*100) **as** rev\_ratio

**from** (**select** branch,

**sum**(**case** **when** **extract**(**year** **from** to\_date(date,**'dd-mm-yy'**))=**'2022' then** "Total" **end**) **as** revenue2022,

**sum**(**case** **when** **extract**(**year** **from** to\_date(date,**'dd-mm-yy'**))='**2023'** **then** "Total" **end**) **as** revenue2023

**from** walmart

**group** **by** **1**)

**order** **by** **2** **desc**

**limit** **5**

**-- 10. which product category generates the highest revenue and profit margin**

**select**

category,

**round**(**sum**("Total")::numeric,**2**) **as** Revenue,

**round**(**sum**(profit\_margin)::numeric,**2**) **as** profitmargin

**from** walmart

**group** **by** **1**

**order** **by** **2,3** **desc**

**-- 11. How do customer purchasing pattern changes based on time of day and branch location**

**select** branch,

**case**

**when** **extract**(**hour** **from** time::time) **between** **6** **and** **12** **then** 'Morning'

**when** **extract**(**hour** **from** time::time) **between** **12** **and** **16** **then** 'Afternoon'

**when** **extract**(**hour** **from** time::time) **between** **16** **and** **20** **then** 'Evening'

**else** 'Night' **end** **as** Time\_of\_day,

**count**(\*) **as** total\_orders,

**sum**("Total") **as** total\_sales

**from** walmart

**group** **by** **1**,**2**

**order** **by** **1**,**2** **desc**;

**-- 12. Is there any relationship between payment method and customer rating**

**select** payment\_method, **round**(**avg**(rating)::numeric,**2**) **as** average\_rating **from** walmart

**group** **by** **1**

**order** **by** **2** **desc**

**-- 13. which branch are underperforming in terms of total sales and profit**

**select** branch, **count**(\*) **as** total\_orders, **sum**("Total") **as** profit **from** walmart

**group** **by** **1**

**order** **by** **2,3** **asc**

**-- 14. identifying which Year,month have most sales**

**select**

**extract**(**year** **from** to\_date(date,**'dd-mm-yy'**)),

to\_char(to\_date(date,'dd-mm-yy'),**'Month'**),

**sum**(invoice\_id) **as** total\_order **from** walmart

**group by** **1**,**2**

**order by** **3** **desc**