RETAIL SALES ANALYSIS

END TO END PROJECT USING

KAGGLE API
PYTHON - PANDAS
SQL

- PRABHAT THAKUR

STEPS FLOWCHART

01

Download Dataset directly in pandas using Kaggle API

02

Data cleaning & processing in the python - pandas

03

Load cleaned data to MySQL DB directly from pandas using SQL libraries

04

Data analysis in MySQL to answer stakeholder's questions

Note:

- We used library like zipfile to extract the zip file in the folder.
- SQL Alchemy library is used to connect our cleaned data to MySQL DB
- Jupyter Notebook is used to perform pandas function and data cleaning.

KAGGLE API IMPORT & DOWNLOAD DATASET

```
[ ]: !pip install kaggle
```

```
[94]: !kaggle -v
```

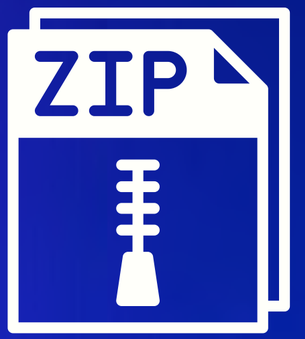
```
Kaggle API 1.6.12
```

```
[93]: !kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
```

UNZIP AND EXTRACT FILE USING ZIPFILE LIBRARY

```
[95]: import zipfile
      zip_ref = zipfile.ZipFile('orders.csv.zip')
      zip_ref.extractall()
      zip_ref.close()
```



CONNECTING AND LOADING CLEANED DATA DIRECTLY TO MYSQL DATABASE


```
import sqlalchemy as sal
#password = 'Qwerty@123'.replace('@', '%40')
connection_string = 'mysql://root:Qwerty%40123@localhost/practice'

# Create the SQLAlchemy engine
engine = sal.create_engine(connection_string)

# Establish a connection to the database
conn = engine.connect()
```

```
df.to_sql('df_orders', con=conn, index = False, if_exists = 'replace')
```

MySQL

JUPYTER NOTEBOOK SNAP FOR THIS PROJECT


```
[ ]: !pip install kaggle
```

```
[ ]: !kaggle -v
```

```
[ ]: !kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
[ ]: import zipfile
zip_ref = zipfile.ZipFile('orders.csv.zip')
zip_ref.extractall()
zip_ref.close()
```

```
[ ]: import pandas as pd
```

```
[ ]: df= pd.read_csv('orders.csv', na_values=['Not Available', 'unknown'])
```

```
[ ]: df['Ship Mode'].unique()
```

```
[ ]: df.rename(columns = {'Order Id':'order_id', 'City': 'city'})
df.columns= df.columns.str.lower()
df.columns= df.columns.str.replace(' ', '_')
```

```
[ ]: df.head()
```

```
[ ]: df['discount']=df['list_price']*(df['discount_percent']*0.01)

df['sale_price']=df['list_price'] - df['discount']

df['profit']=df['sale_price'] - df['cost_price']
```

```
[ ]: df['discount']=df['list_price']*(df['discount_percent']*0.01)

df['sale_price']=df['list_price'] - df['discount']

df['profit']=df['sale_price'] - df['cost_price']
```

```
[ ]: df.dtypes['order_date']

df['order_date'] = pd.to_datetime(df['order_date'], format="%Y-%m-%d")
```

```
[ ]: df.drop(columns=['list_price', 'cost_price', 'discount_percent'], inplace = True)
```

```
[ ]: df
```

```
[ ]: import sqlalchemy as sal
#password = 'Qwerty@123'.replace('@', '%40')
connection_string = 'mysql://root:Qwerty%40123@localhost/practice'

# Create the SQLAlchemy engine
engine = sal.create_engine(connection_string)

# Establish a connection to the database
conn = engine.connect()
```

```
[ ]: df.to_sql('df_orders', con=conn, index = False, if_exists = 'replace')
```

Q1. FIND TOP 10 HIGHEST REVENUE GENERATING PRODUCTS BY CATEGORY?

```sql
SELECT category, product_id, SUM(sale_price) as sales
FROM df_orders
GROUP BY product_id, category
ORDER BY SUM(sale_price) DESC
LIMIT 10;
```

QUERY

category	product_id	sales
Technology	TEC-CO-10004722	59514
Office Supplies	OFF-BI-10003527	26525.3
Technology	TEC-MA-10002412	21734.4
Furniture	FUR-CH-10002024	21096.2
Office Supplies	OFF-BI-10001359	19090.2
Office Supplies	OFF-BI-10000545	18249
Technology	TEC-CO-10001449	18151.2
Technology	TEC-MA-10001127	17906.4
Office Supplies	OFF-BI-10004995	17354.8
Office Supplies	OFF-SU-10000151	16325.8

OUTPUT

Q2. FIND TOP 5 HIGHEST SELLING PRODUCTS IN EACH REGION?

QUERY

```sql
with cte as (
SELECT region, product_id, SUM(sale_price) as sales
FROM df_orders
GROUP BY region,product_id),
new_cte as(
Select *,
row_number() OVER(partition by region order by sales desc)as rn
from cte)
Select * from new_cte
Where rn <=5;
```

OUTPUT

region	product_id	sales	rn
Central	TEC-CO-10004722	16975	1
Central	TEC-MA-10000822	13770	2
Central	OFF-BI-10001120	11056.5	3
Central	OFF-BI-10000545	10132.7	4
Central	OFF-BI-10004995	8416.1	5
East	TEC-CO-10004722	29099	1
East	TEC-MA-10001047	13767	2
East	FUR-BO-10004834	11274.1	3
East	OFF-BI-10001359	8463.59	4
East	TEC-CO-10001449	8316	5
South	TEC-MA-10002412	21734.4	1
South	TEC-MA-10001127	11116.4	2
South	OFF-BI-10001359	8053.2	3
South	TEC-MA-10004125	7840	4
South	OFF-BI-10003527	7391.4	5
West	TEC-CO-10004722	13440	1
West	OFF-SU-10000151	12592.3	2
West	FUR-CH-10001215	9604	3
West	OFF-BI-10003527	7804.79	4
West	TEC-AC-10003832	7722.7	5

Q3. FIND MONTH OVER MONTH GROWTH COMPARISON FOR 2022 AND 2023 SALES EG : JAN 2022 VS JAN 2023?

```sql
with cte as (
SELECT YEAR(order_date) as order_year,
month(order_date) as order_month, sum(sale_price) as sales
from df_orders
GROUP BY YEAR(order_date), month(order_date)
order by YEAR(order_date), month(order_date)
)
SELECT order_month,
truncate(sum(case when order_year=2022 then sales else 0 end),2) as sales_2022,
truncate(sum(case when order_year=2023 then sales else 0 end),2) as sales_2023
FROM cte
GROUP BY order_month
ORDER BY order_month;
```

QUERY

order_month	sales_2022	sales_2023
1	94712.49	88632.6
2	90091	128124.2
3	80105.99	82512.29
4	95451.6	111568.6
5	79448.29	86447.89
6	94170.49	68976.5
7	78652.2	90563.79
8	104807.99	87733.6
9	79142.19	76658.59
10	118912.69	121061.49
11	84225.29	75432.79
12	95869.9	102556.1

OUTPUT

Q4. FOR EACH CATEGORY WHICH MONTH & YEAR HAD HIGHEST SALES?

```sql
with cte as
(SELECT category, MONTHNAME(order_date) as month, YEAR(order_date) as year, TRUNCATE(SUM(sale_price),2) as sales
FROM df_orders
GROUP BY category, MONTHNAME(order_date),YEAR(order_date)
Order By month, sales DESC),
n_cte as(
SELECT *,
row_number() OVER (PARTITION BY category ORDER BY sales DESC) as rn
FROM cte
)
SELECT * FROM n_cte
WHERE rn =1;
```

QUERY

category	month	year	sales	rn
Furniture	October	2022	42888.9	1
Office Supplies	February	2023	44118.49	1
Technology	October	2023	53000.1	1

OUTPUT

Q5. WHICH SUB CATEGORY HAD HIGHEST GROWTH BY PROFIT IN 2023 COMPARE TO 2022?

```sql
with cte as (
SELECT sub_category, YEAR(order_date) as order_year, sum(sale_price) as sales
from df_orders
GROUP BY sub_category, YEAR(order_date)
order by YEAR(order_date)
),
n_cte as (
SELECT sub_category,
truncate(sum(case when order_year=2022 then sales else 0 end),2) as sales_2022,
truncate(sum(case when order_year=2023 then sales else 0 end),2) as sales_2023
FROM cte
GROUP BY sub_category
)
SELECT *, truncate((sales_2023 - sales_2022)*100/sales_2022,2) as YOY_growth
FROM n_cte
ORDER BY YOY_growth DESC
LIMIT 1
```

QUERY

sub_category	sales_2022	sales_2023	YOY_growth
Supplies	16140.7	28917.4	79.15

OUTPUT

THANK YOU!