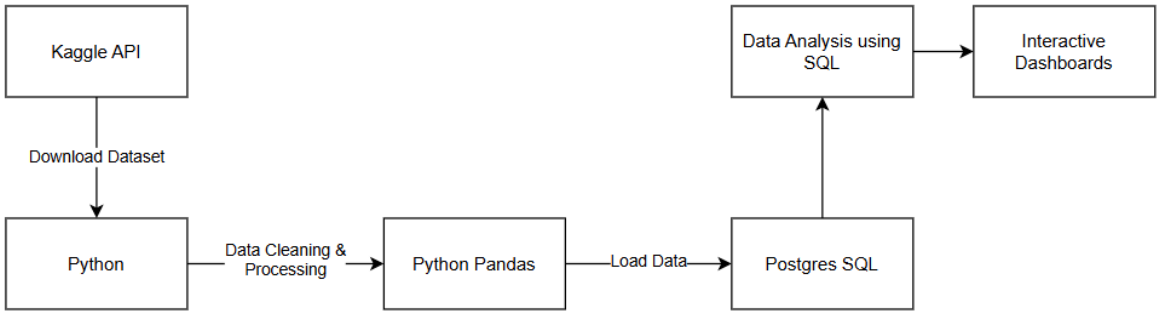
RETAILEDGE: SALES & PROFITS ANALYTICS

In this analytics project mainly involves in ETL, Analytics through SQL and Dashboard creation from the insights derived from analytics.

Techstack used: Python, SQL, PowerBI

ETL(Extract, Transform, Load)



1. Create API token from your kaggle account, it will generate new token in the json format and keep this generated token at "/.kaggle/kaggle.json" in your C drive

2. For the dataset link given : https://www.kaggle.com/datasets/ankitbansal06/retail-orders

3. Launch a Jupyter Notebook, import necessary libraries and download the dataset from kaggle to local machine

```
In [1]: #import libraries
        !pip install kaggle
        import kaggle

Requirement already satisfied: kaggle in c:\users\hp\anaconda3\lib\site-packages (1.7.4.2)
Requirement already satisfied: webencodings in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (0.5.1)
Requirement already satisfied: certifi>=14.05.14 in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (2022.12.7)
Requirement already satisfied: six>=1.10 in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (1.16.0)
Requirement already satisfied: tqdm in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (4.64.1)
Requirement already satisfied: text-unidecode in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (1.3)
Requirement already satisfied: charset-normalizer in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (2.0.4)
Requirement already satisfied: python-dateutil>=2.5.3 in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (2.8.2)
Requirement already satisfied: protobuf in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (5.29.1)
Requirement already satisfied: idna in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (3.4)
Requirement already satisfied: urllib3>=1.15.1 in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (2.2.1)
Requirement already satisfied: requests in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (2.32.3)
Requirement already satisfied: python-slugify in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (8.0.4)
Requirement already satisfied: setuptools>=21.0.0 in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (69.5.1)
Requirement already satisfied: bleach in c:\users\hp\anaconda3\lib\site-packages (from kaggle) (4.1.0)
Requirement already satisfied: packaging in c:\users\hp\anaconda3\lib\site-packages (from bleach->kaggle) (24.2)
Requirement already satisfied: colorama in c:\users\hp\anaconda3\lib\site-packages (from tqdm->kaggle) (0.4.6)

In [2]: import kaggle

In [3]: !kaggle datasets download ankitbansal06/retail-orders

Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
retail-orders.zip: Skipping, found more recently modified local copy (use --force to force download)
```

It will download in the zip format, we have to extract file from zip file and can start performing data cleaning using python and pandas

4. Read data from file and hande null values
5. Renaming the columns names by making them lower case and replace space with underscore
6. Derive new columns discount , sale price and profit.
7. Converting order date column from object data type to datetime.
8. Drop cost price list price and discount percent columns as longer required.

9. Have to load the cleaned dataframe into Postgres SQL DB using sql

By using sqlalchemy, psycopg2 libraries we can import data into postgres sql

```
username = 'postgres'
password = 'root'
host = 'localhost'      # or your remote host/IP
port = '5432'           # default PostgreSQL port
database = 'df_orders'

engine = create_engine(f'postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}')
conn=engine.connect()

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

Analysing through SQL:

Find top 10 highest revenue generating products

```
select product_id, sum(sales_price)  from df_orders
group by product_id
order by sum(sales_price) desc limit 10
```

Find top 5 highest selling products in each region

```
with cte as(select region,product_id,sum(sales_price)as sales,row_number() over(partition by region
order by sum(sales_price) desc) as rn from df_orders
group by region,product_id
order by region,sales desc)

select region,product_id,sales from cte where rn<=5 group by region,product_id,sales order by
region,sales desc
```

Find month over month growth comparison for 2022 and 2023 sales jan 2022 vs jan 2023

```
WITH cte AS (
   SELECT
      EXTRACT(MONTH FROM order_date) AS month,
```

```
    EXTRACT(YEAR FROM order_date) AS year,
    SUM(sales_price) AS sale
  FROM df_orders
  GROUP BY month, year
  ORDER BY year, month
)

SELECT
  month,
  sum(CASE WHEN year = 2022 THEN sale END) AS sale_22,
  sum(CASE WHEN year = 2023 THEN sale END) AS sale_23
FROM cte
        GROUP BY month
        order by month;
```

For each category which month had highest sales

```
with cte as(
select category,to_char(order_date,'YYYYMM')as YM,sum(sales_price) as sale from df_orders
group by category,YM
        order by sum(sales_price) desc
        )
        select category,ym,sale from(
select *,row_number() over (partition by category order by sale desc) as rn from cte) where rn=1
```

which sub category had highest growth by profit in 2023 compare to 2022

```
with cte as(select sub_category,extract(year from order_date) as yr,sum(sales_price)as sale from
df_orders
group by sub_category,yr),
cte2 as (
select sub_category,sum(case when yr=2022 then sale end) as sale_2022,sum(case when yr=2023
then sale end) as sale_2023
from cte
group by sub_category
order by sub_category)

select *, (sale_2023-sale_2022)as margin from cte2 order by margin desc limit 1
```

Monthly Sales Trend
Insight: See how sales vary by month and year.

```
SELECT
  EXTRACT(YEAR FROM order_date) AS year,
  EXTRACT(MONTH FROM order_date) AS month,
  SUM(sales_price) AS total_sales
FROM df_orders
GROUP BY year, month
```

```
ORDER BY year, month;
```

Top 3 months

```
with cte as (SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    EXTRACT(MONTH FROM order_date) AS month,
    SUM(sales_price) AS total_sales
FROM df_orders
GROUP BY year, month
ORDER BY year, month)
select year,month,total_sales  from(
select *,row_number() over (partition by year order by total_sales desc) as rn from cte ) where rn<=3
```

Profit by Category and Sub Category
Insight: Identify which categories are most profitable.

```
SELECT
    category,
    SUM(sales_price) AS total_sales,
    SUM(profit) AS total_profit
FROM df_orders
GROUP BY category
ORDER BY total_profit DESC;
```

```
SELECT
    category,sub_category,
    SUM(sales_price) AS total_sales,
    SUM(profit) AS total_profit
FROM df_orders
GROUP BY category,sub_category
ORDER BY total_profit DESC;
```

Most Sold Sub-Categories (by quantity)

```
SELECT
    category,sub_category,
    SUM(quantity) AS total_quantity
FROM df_orders
GROUP BY category,sub_category
ORDER BY total_quantity DESC;
```

Ship Mode Popularity

```
select ship_mode,sum(quantity) as orders,sum(sales_price) as total_sales from df_orders
        where ship_mode is not null
group by ship_mode
```

```
order by total_sales desc
```

Discount Impact on Profit
Insight: Are discounts helping or hurting?

```
SELECT
   ROUND(AVG(discount), 2) AS avg_discount,
   ROUND(AVG(profit), 2) AS avg_profit
FROM df_orders;
```

discount vs profit among category and subcategory

```
SELECT
   round(avg(discount),2) as disc, round(avg(profit),2) as pr,category,sub_category
FROM df_orders
group by category,sub_category
order by pr desc
```

```
select round(avg(profit),2) as avg_profit,
case when discount>=0 and discount<=100 then '0-100'
when discount>100 and discount<=200 then '100-200'
when discount>200 and discount<=300 then '200-300'
when discount>300 and discount<=400 then '300-400'
when discount>400 and discount<=500 then '400-500'
when discount>500 and discount<=600 then '500-600'
when discount>600 and discount<=700 then '600-700'
when discount>700 and discount<=800 then '700-800'
when discount>800 and discount<=900 then '800-900'
when discount>900 and discount<=1000 then '900-1000'
end as disc
from df_orders
group by disc
order by disc
```

LOSS VS Sub-Categories

```
WITH CTE AS(SELECT
   order_id,
   TO_CHAR(order_date,'mmYYYY') AS MONTHYR,
         CATEGORY,
         SUB_CATEGORY,
   product_id,
   sales_price,
   discount,
   profit
FROM df_orders
WHERE profit <= 0
ORDER BY profit)
-- SUBCTE AS(
```

```
SELECT SUB_CATEGORY,SUM(PROFIT) AS LOSS FROM CTE GROUP BY SUB_CATEGORY
ORDER BY LOSS

-- SELECT SUB_CATEGORY
```

Dashboard:

Made an interactive dashboard using Power BI desktop. That file uploaded in git. Please check out.