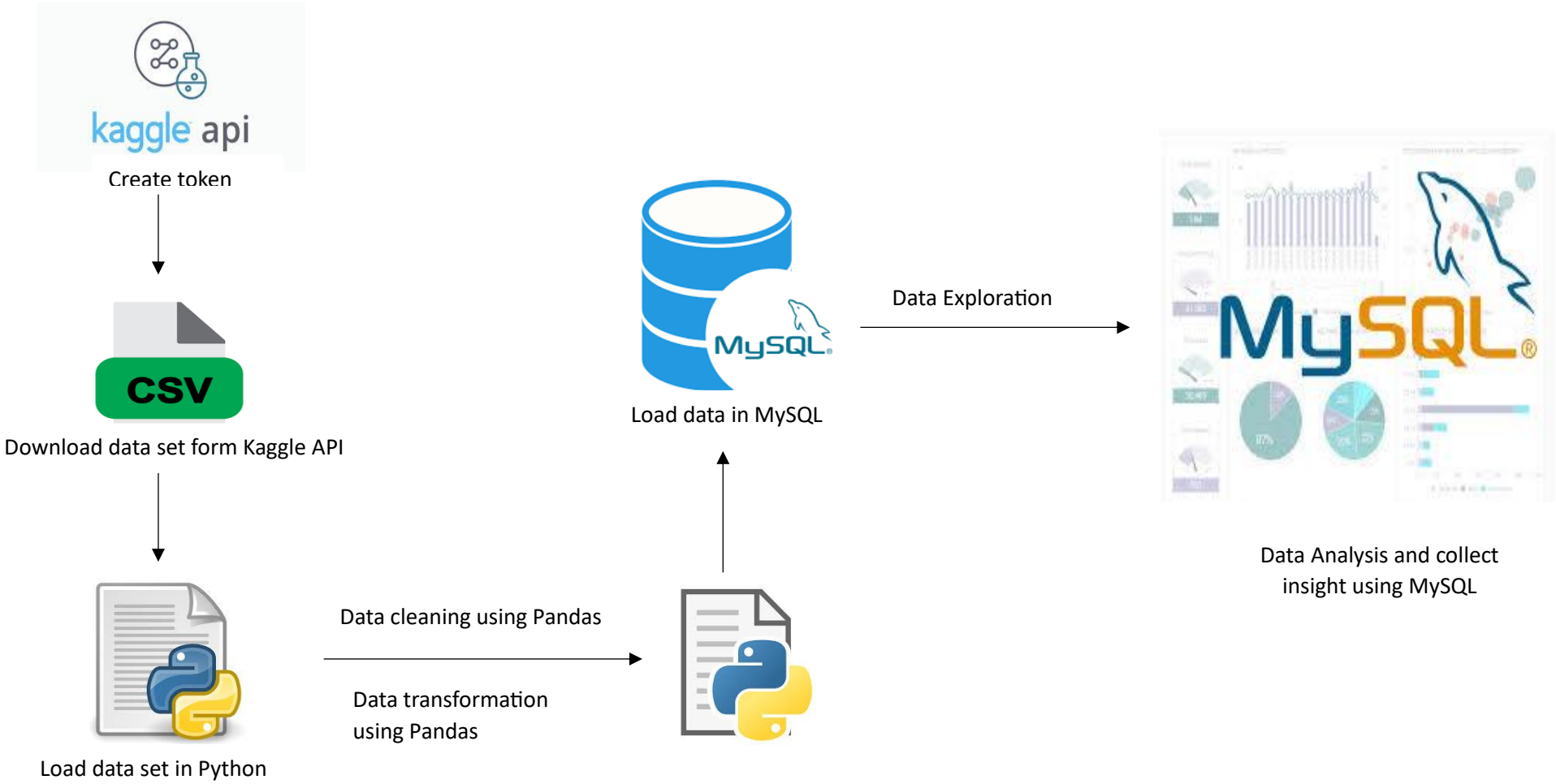
Data Analytics Project on Retail Order

(Kaggle API + Python + My SQL)



Procedure

1.Data Collection

1. Download data set using Kaggle API

```
[3]: #download dataset using kaggle api
import kaggle
!kaggle datasets download ankitbansal06/retail-orders -f orders.csv
```

```
Dataset URL: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
License(s): CC0-1.0
Downloading orders.csv.zip to C:\Users\pavan\projects with kaggle data set\Project_1

  0%|          | 0.00/200k [00:00<?, ?B/s]
100%|##########| 200k/200k [00:00<00:00, 421kB/s]
100%|##########| 200k/200k [00:00<00:00, 420kB/s]
```

2. Extract file from zip file

```
[ ]: #extract file from zip file
import zipfile
zip_ref=zipfile.zipfile('orders.csv.zip')
zip_ref.extractall()
zip_ref.close()
```

3. Read data from the file

```
[2]: #read data from the file and handle null value
import pandas as pd
df=pd.read_csv('orders.csv',)
df.head(4)
```

[2]:

	Order Id	Order Date	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub Category	Product Id	cost price	List Price	Quantity	Discount Percent
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2

2.Data Cleaning

1. Read data from the file and handle null values

[3]:
```
#read data from the file and handle null value
import pandas as pd
df=pd.read_csv('orders.csv',na_values=['Not Available','unknown'])
df.head(4)
```

[3]:

	Order Id	Order Date	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub Category	Product Id	cost price	List Price	Quantity	Discount Percent
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2	5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5	2

2. Rename columns names make them lower case and replace space with underscores

[4]:
```
#rename columns names make them lower case and replace space with underscore
df.columns=df.columns.str.lower()
df.columns=df.columns.str.replace(' ','_')
df.head(4)
```

[4]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	quantity
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5

3. Convert order date form object data type to date time

[5]:
```
#convert order date from object data type to datetime
df['order_date']=pd.to_datetime(df['order_date'],format="%Y-%m-%d")
df.dtypes
```

[5]:
```
order_id                     int64
order_date          datetime64[ns]
ship_mode                   object
segment                     object
country                     object
city                        object
state                       object
postal_code                  int64
region                      object
category                    object
sub_category                object
product_id                  object
cost_price                   int64
list_price                   int64
quantity                     int64
discount_percent             int64
discount                   float64
sale_price                 float64
profit                     float64
dtype: object
```

4.Check data frame

[6]:
```
df
```

[6]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	qu
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	

3.Data Exploration

1. Check short statics

```
[7]: df.describe()
```

[7]:

	order_id	postal_code	cost_price	list_price	quantity	discount_percent	discount	sale_price	profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	201.189714	229.756854	3.789574	3.484090	8.037953	221.718901	20.529188
std	2885.163629	32063.693350	537.743203	623.245839	2.225110	1.114211	22.978004	601.399604	72.514547
min	1.000000	1040.000000	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	-5.000000
25%	2499.250000	23223.000000	20.000000	20.000000	2.000000	2.000000	0.600000	19.200000	-0.400000
50%	4997.500000	56430.500000	50.000000	50.000000	3.000000	3.000000	1.800000	49.000000	6.000000
75%	7495.750000	90008.000000	180.000000	210.000000	5.000000	4.000000	7.000000	201.600000	16.700000
max	9994.000000	99301.000000	18110.000000	22640.000000	14.000000	5.000000	905.600000	21734.400000	3624.400000

2. Check information of data frame

```
[8]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   order_id          9994 non-null   int64
 1   order_date        9994 non-null   object
 2   ship_mode         9988 non-null   object
 3   segment           9994 non-null   object
 4   country           9994 non-null   object
 5   city              9994 non-null   object
 6   state             9994 non-null   object
 7   postal_code       9994 non-null   int64
 8   region            9994 non-null   object
 9   category          9994 non-null   object
 10  sub_category      9994 non-null   object
 11  product_id        9994 non-null   object
 12  cost_price        9994 non-null   int64
 13  list_price        9994 non-null   int64
 14  quantity          9994 non-null   int64
 15  discount_percent  9994 non-null   int64
 16  discount          9994 non-null   float64
 17  sale_price        9994 non-null   float64
 18  profit            9994 non-null   float64
```

4.Data Transformation

1.Derive new columns discount, sales price and profit

[5]:
```
#derive new columns discount, sales price and profit
df['discount']=df['list_price']*df['discount_percent']*.01
df['sale_price']=df['list_price']-df['discount']
df['profit']=df['sale_price']-df['cost_price']
df.head(4)
```

[5]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	cost_price	list_price	quantity
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	240	260	2
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	600	730	3
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	10	10	2
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	780	960	5

2. Drop columns which is not use

[9]:
```
#Drop columns which is not use cost price, list price and discount percent columns
df.drop(columns=['list_price','cost_price','discount_percent'],inplace = True)
df.head(4)
```

[9]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	quantity	discount	sale_price
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	2	5.2	254.8
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	3	21.9	708.1
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	2	0.5	9.5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	5	19.2	940.8

5.Data load into My SQL

1. Create a table in MySQL

```sql
create table df_orders(
    order_id int primary key,
    order_date date,
    ship_mode varchar(20),
    segment varchar(20),
    country varchar(20),
    city varchar(20),
    state varchar(20),
    postal_code varchar(20),
    region varchar(20),
    category varchar(20),
    sub_category varchar(20),
    product_id varchar(50),
    quantity int,
    discount decimal(7,2),
    sale_price decimal(7,2),
    profit decimal(7,2)
);
```

2.Load the data and connect database in MySQL

```
•[8]: #load the data into sql server using replace option
import sqlalchemy
import pymysql
#create variables
user ="********"
password ="********"
host ="localhost"
db_name="retail_order"
port=3306
#make connection to our mysql database
sql_engine=sqlalchemy.create_engine(f'mysql+pymysql://{user}:{password}@{host}/{db_name}', pool_recycle=port)
db_connection=sql_engine.connect()
```

```
[10]: df.head(4)
```

[10]:

	order_id	order_date	ship_mode	segment	country	city	state	postal_code	region	category	sub_category	product_id	quantity	discount	sale_price
0	1	2023-03-01	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	FUR-BO-10001798	2	5.2	254.8
1	2	2023-08-15	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	FUR-CH-10000454	3	21.9	708.1
2	3	2023-01-10	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	OFF-LA-10000240	2	0.5	9.5
3	4	2022-06-18	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	FUR-TA-10000577	5	19.2	940.8

3.Load the data into MySQL using append option

```
[11]: df.to_sql('df_orders', con=db_connection, index=False, if_exists='append')
```

```
[11]: 9994
```

6. Analysis in MySQL and Collect Insights

1.Find top 10 highest revenue generating products

```
select product_id, sum(sale_price)as sales
from df_orders
group by product_id
order by sales desc
limit 10
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

product_id	sales
TEC-CO-10004722	59514.00
OFF-BI-10003527	26525.30
TEC-MA-10002412	21734.40
FUR-CH-10002024	21096.20
OFF-BI-10001359	19090.20
OFF-BI-10000545	18249.00
TEC-CO-10001449	18151.20
TEC-MA-10001127	17906.40
OFF-BI-10004995	17354.80
OFF-SU-10000151	16325.80

2. Find top 5 highest selling products in each region

```sql
WITH ranked_products AS (
    SELECT
        region,
        product_id,
        SUM(sale_price) AS sales,
        ROW_NUMBER() OVER (PARTITION BY region ORDER BY SUM(sale_price) DESC) AS `rank`
    FROM df_orders
    GROUP BY region, product_id
)
SELECT region, product_id, sales
FROM ranked_products
WHERE `rank` <= 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

region	product_id	sales
Central	TEC-CO-10004722	16975.00
Central	TEC-MA-10000822	13770.00
Central	OFF-BI-10001120	11056.50
Central	OFF-BI-10000545	10132.70
Central	OFF-BI-10004995	8416.10
East	TEC-CO-10004722	29099.00
East	TEC-MA-10001047	13767.00
East	FUR-BO-10004834	11274.10
East	OFF-BI-10001359	8463.60
East	TEC-CO-10001449	8316.00
South	TEC-MA-10002412	21734.40
South	TEC-MA-10001127	11116.40
South	OFF-BI-10001359	8053.20
South	TEC-MA-10004125	7840.00

Result 3

3.Find month over month growth comparison for 2022 and 2023 sales

```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)
)
select order_month,
sum(case when order_year = 2022 then sales else 0 end) as sales_2022
,sum(case when order_year = 2022 then sales else 0 end) as sales_2022
from cte
group by order_month
order by order_month
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_month	sales_2022	sales_2022
1	94712.50	94712.50
2	90091.00	90091.00
3	80106.00	80106.00
4	95451.60	95451.60
5	79448.30	79448.30
6	94170.50	94170.50
7	78652.20	78652.20
8	104808.00	104808.00
9	79142.20	79142.20
10	118912.70	118912.70
11	84225.30	84225.30
12	95869.90	95869.90

Result 1

4. For each category which month has highest sales

```sql
with cte as(
select category, format(order_date, 'yyyymm') as order_year_month, sum(sale_price) as sales
from df_orders
group by category, format(order_date, 'yyyymm')
)
select * from(
select *, row_number() over(partition by category order by sales desc) as rn
from cte
) a
where rn=1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	order_year_month	sales	rn
Furniture	20,230,208	6247.00	1
Office Supplies	20,230,227	10474.60	1
Technology	20,231,013	23064.40	1

5. Which subcategory 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)
)
, cte2 as (
select sub_category,
sum(case when order_year = 2022 then sales else 0 end) as sales_2022
,sum(case when order_year = 2023 then sales else 0 end) as sales_2023
from cte
group by sub_category
)
select *, (sales_2023 - sales_2022)*100/sales_2022 as growth_percentage
from cte2
order by (sales_2023 - sales_2022)*100/sales_2022 desc
limit 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sub_category	sales_2022	sales_2023	growth_percentage
Supplies	16140.70	28917.40	79.158277