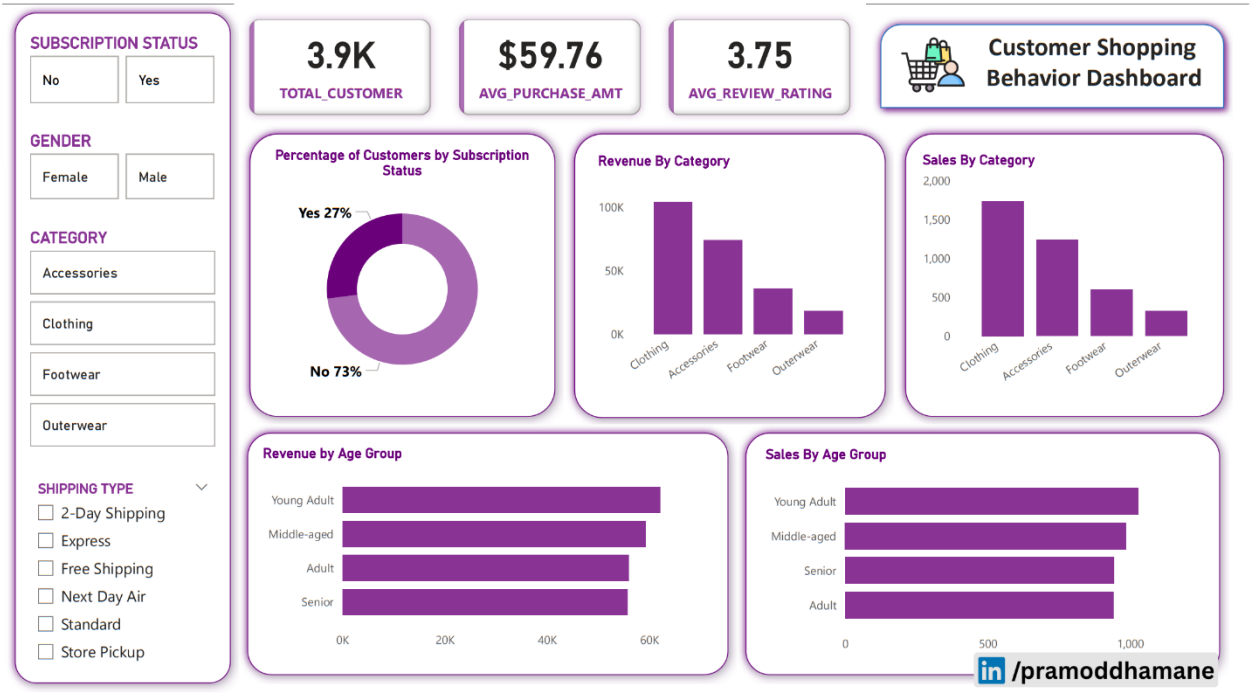
Data Cleaning & Analysis

Customer Shopping Behaviour Analysis

Dashboard



Python

```
import pandas as pd

df = pd.read_csv(r"[File_path]\customer_shopping_Behaviour .csv")

df.head() # view first 5 rows
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Pay Me
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express	Yes	Yes	14	V
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express	Yes	Yes	2	
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping	Yes	Yes	23	
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air	Yes	Yes	49	F
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping	Yes	Yes	31	F

```
df.info()  # checks column and their data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3900 entries, 0 to 3899
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Customer ID             3900 non-null   int64
 1   Age                     3900 non-null   int64
 2   Gender                  3900 non-null   object
 3   Item Purchased          3900 non-null   object
 4   Category                3900 non-null   object
 5   Purchase Amount (USD)   3900 non-null   int64
 6   Location                3900 non-null   object
 7   Size                    3900 non-null   object
 8   Color                   3900 non-null   object
 9   Season                  3900 non-null   object
 10  Review Rating           3863 non-null   float64
 11  Subscription Status     3900 non-null   object
 12  Shipping Type           3900 non-null   object
 13  Discount Applied        3900 non-null   object
 14  Promo Code Used         3900 non-null   object
 15  Previous Purchases      3900 non-null   int64
 16  Payment Method          3900 non-null   object
 17  Frequency of Purchases  3900 non-null   object
dtypes: float64(1), int64(4), object(13)
memory usage: 548.6+ KB
```

```
df.describe()      # checks summary statistics of numeric columns
```

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3900.000000	3900.000000	3900.000000	3863.000000	3900.000000
mean	1950.500000	44.068462	59.764359	3.750065	25.351538
std	1125.977353	15.207589	23.685392	0.716983	14.447125
min	1.000000	18.000000	20.000000	2.500000	1.000000
25%	975.750000	31.000000	39.000000	3.100000	13.000000
50%	1950.500000	44.000000	60.000000	3.800000	25.000000
75%	2925.250000	57.000000	81.000000	4.400000	38.000000
max	3900.000000	70.000000	100.000000	5.000000	50.000000

```
df.describe(include='all')  # summary statistics for numeric as well as categorical columns
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	3900	3900	3900	3900
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	2	6	2	2
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	No	Free Shipping	No	No
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	2847	675	2223	2223
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	NaN	NaN	NaN	NaN
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	NaN	NaN	NaN	NaN
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	NaN	NaN	NaN	NaN
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	NaN	NaN	NaN	NaN
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	NaN	NaN	NaN	NaN
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	NaN	NaN	NaN	NaN
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	NaN	NaN	NaN	NaN

```
df.isnull().sum()      # find null value
```

```
Customer ID                0
Age                        0
Gender                     0
Item Purchased             0
Category                   0
Purchase Amount (USD)      0
Location                   0
Size                       0
Color                      0
Season                     0
Review Rating             37
Subscription Status        0
Shipping Type              0
Discount Applied           0
Promo Code Used            0
Previous Purchases         0
Payment Method             0
Frequency of Purchases     0
dtype: int64
```

```
# For sake of simplicity we are inserting null values with category wise median value

df['Review Rating'] = df.groupby('Category')['Review Rating'].transform(lambda x:x.fillna(x.median()))

df.isnull().sum()
```

```
Customer ID                0
Age                        0
Gender                     0
Item Purchased             0
Category                   0
Purchase Amount (USD)      0
Location                   0
Size                       0
Color                      0
Season                     0
Review Rating              0
Subscription Status        0
Shipping Type              0
Discount Applied           0
Promo Code Used            0
Previous Purchases         0
Payment Method             0
Frequency of Purchases     0
dtype: int64
```

```
# will make column names in the snake case for ease for use
df.columns = df.columns.str.lower()    # converting in lower case

df.columns = df.columns.str.replace(' ','_')    # replacing space with
underscore

df.columns
```

```
Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
       'purchase_amount_(usd)', 'location', 'size', 'color', 'season',
       'review_rating', 'subscription_status', 'shipping_type',
       'discount_applied', 'promo_code_used', 'previous_purchases',
       'payment_method', 'frequency_of_purchases'],
      dtype='object')
```

```
df = df.rename(columns={'purchase_amount_(usd)' : 'purchase_amount'})
df.columns
```

```
Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
       'purchase_amount', 'location', 'size', 'color', 'season',
       'review_rating', 'subscription_status', 'shipping_type',
       'discount_applied', 'promo_code_used', 'previous_purchases',
       'payment_method', 'frequency_of_purchases'],
      dtype='object')
```

```
# create a column age_group
labels = ['Young Adult','Adult','Middle-aged', 'Senior']

df['age_group'] = pd.qcut(df['age'],q=4, labels =labels)
# 'qcut' splits ages into four equal size group based on data and assign
labels

df[['age','age_group']].head(10)
```

	age	age_group
0	55	Middle-aged
1	19	Young Adult
2	50	Middle-aged
3	21	Young Adult
4	45	Middle-aged
5	46	Middle-aged
6	63	Senior
7	27	Young Adult
8	26	Young Adult
9	57	Middle-aged

```
#create the column purchase_frequency_days

frequency_mapping = {
    'Weekly': 7,
    'Bi-Weekly': 14,
    'Fortnightly': 14,
    'Monthly': 30,
    'Every 3 Months': 90,
    'Quarterly': 90,
    'Annually': 365
}

df['purchase_frequency_days'] =
df['frequency_of_purchases'].map(frequency_mapping)

df[['purchase_frequency_days','frequency_of_purchases']].head(10)
```

	purchase_frequency_days	frequency_of_purchases
0	14	Fortnightly
1	14	Fortnightly
2	7	Weekly
3	7	Weekly
4	365	Annually
5	7	Weekly
6	90	Quarterly
7	7	Weekly
8	365	Annually
9	90	Quarterly

```
# check 'discount_applied' and 'promo_code_used' columns if they have
resemblance.
df[['discount_applied','promo_code_used']].head(10)
```

	discount_applied	promo_code_used
0	Yes	Yes
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	Yes	Yes
6	Yes	Yes
7	Yes	Yes
8	Yes	Yes
9	Yes	Yes

```
#comparing two columns

(df['discount_applied'] == df['promo_code_used']).all()
```

```
np.True_
```

```
# they are same. will drop one of them

df = df.drop('promo_code_used', axis=1)

df.columns
```

```
Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
       'purchase_amount', 'location', 'size', 'color', 'season',
       'review_rating', 'subscription_status', 'shipping_type',
       'discount_applied', 'previous_purchases', 'payment_method',
       'frequency_of_purchases', 'age_group', 'purchase_frequency_days'],
      dtype='object')
```

```
# Connect to database

from sqlalchemy import create_engine, Numeric

engine =
create_engine('oracle://[username]:[password]@hostname:portnumber/SID')

#replace with your [username],[password], hostname, Port_no, SID'
```

```
#code uploads your DataFrame into SQL and ensures all float columns have
proper numeric formatting in the SQL database.

df.to_sql(
    'customer_shopping',
    engine,
    if_exists='replace',
    index=False,
    dtype={col: Numeric(10, 2)
           for col in df.select_dtypes(include=['float64']).columns}
)
3900


# OR
#Instead of connecting to Database simply export into csv.

# export cleaned file

df.to_csv(r'[folder_location]\export_customer_shopping.csv', index=False)
```

-- Q1. What is the total revenue generated by male vs. female customers?

```
SELECT
   gender,
   SUM(purchase_amount) AS revenue
FROM
   customer_shopping
GROUP BY
   gender;
```

	GENDER	REVENUE
1	Male	157890
2	Female	75191

--Q2. Which customers used a discount but still spent more than the average purchase amount?

```
SELECT
   customer_id,
   purchase_amount
FROM
   customer_shopping
WHERE
   discount_applied = 'Yes'
   AND purchase_amount > (
      SELECT
         round(AVG(purchase_amount), 2)
      FROM
         customer_shopping
   );
```

	CUSTOMER_ID	PURCHASE_AMOUNT
1	182	61
2	183	96
3	188	94
4	190	82
5	191	70
6	192	76
7	194	100

--Q3. Which are the top 5 products with the highest average review rating?

```
SELECT
   *
FROM
   (
      SELECT
         item_purchased,
         AVG(review_rating) AS avg_rating
```

```sql
        FROM
            customer_shopping
        GROUP BY
            item_purchased
        ORDER BY
            avg_rating DESC
    )
WHERE
    ROWNUM <= 5;
```

	ITEM_PURCHASED	AVG_RATING
1	Gloves	3.86142857142857142857142857142857142857
2	Sandals	3.844375
3	Boots	3.81875
4	Hat	3.8012987012987012987012987012987012987
5	Skirt	3.78481012658227848101265822784810126582

```sql
-- or

SELECT
    *
FROM
    (
        SELECT
            item_purchased,
            round(AVG(review_rating), 1) AS avg_rating,
            DENSE_RANK() OVER(
                ORDER BY
                    AVG(review_rating) DESC
            ) AS rating_rank
        FROM
            customer_shopping
        GROUP BY
            item_purchased
    )
WHERE
    rating_rank <= 5;
```

	ITEM_PURCHASED	AVG_RATING	RATING_RANK
1	Gloves	3.9	1
2	Sandals	3.8	2
3	Boots	3.8	3
4	Hat	3.8	4
5	Skirt	3.8	5

--Q4. Compare the average Purchase Amounts between Standard and Express Shipping.

```sql
SELECT
    shipping_type,
    round(AVG(purchase_amount), 2) AS avg_amount
FROM
    customer_shopping
WHERE
    shipping_type IN (
        'Standard',
        'Express'
    )
GROUP BY
    shipping_type;
```

	SHIPPING_TYPE	AVG_AMOUNT
1	Express	60.48
2	Standard	58.46

--Q5. Do subscribed customers spend more? Compare average spend and total revenue

--between subscribers and non-subscribers.

```
SELECT
   subscription_status,
   COUNT(customer_id),
   round(AVG(purchase_amount), 2) AS avg_spend,
   SUM(purchase_amount) AS total_revenue
FROM
   customer_shopping
WHERE

subscription_status
IN (
      'Yes',
      'No'
   )
GROUP BY
   subscription_status;
```

	SUBSCRIPTION_STATUS	COUNT(CUSTOMER_ID)	AVG_SPEND	TOTAL_REVENUE
1	Yes	1053	59.49	62645
2	No	2847	59.87	170436

--Q6. Which 5 products have the highest percentage of purchases with discounts applied?

```
SELECT
   *
FROM
   (
      SELECT
         item_purchased,
         round(100 * SUM(
            CASE
               WHEN discount_applied = 'Yes' THEN
                  1
               ELSE
                  0
            END
         ) / COUNT(*), 2) AS discount_rate
      FROM
         customer_shopping
      GROUP BY
         item_purchased
      ORDER BY
         discount_rate DESC
   )
WHERE
   ROWNUM <= 5;
```

	ITEM_PURCHASED	DISCOUNT_RATE
1	Hat	50
2	Sneakers	49.66
3	Coat	49.07
4	Sweater	48.17
5	Pants	47.37

--Q7. Segment customers into New, Returning, and Loyal based on their total number of previous purchases, and show the count of each segment.

```
WITH customer_type AS (
   SELECT
      customer_id,
      previous_purchases,
      CASE
         WHEN previous_purchases = 1            THEN
            'New'
         WHEN previous_purchases BETWEEN 2 AND 10 THEN
            'Returning'
         ELSE
            'Loyal'
      END AS customer_segment
   FROM
      customer_shopping
)
SELECT
   customer_segment,
   COUNT(*) AS "Number of Customers"
FROM
   customer_type
GROUP BY
   customer_segment;
```

	CUSTOMER_SEGMENT	Number of Customers
1	Returning	701
2	New	83
3	Loyal	3116

--Q8. What are the top 3 most purchased products within each category?

```
SELECT
   category,
   item_purchased,
   total_purchases,
   item_rank
FROM
   (
      SELECT
         category,
         item_purchased,
         COUNT(*) AS total_purchases,
         ROW_NUMBER() OVER(
            PARTITION BY category
            ORDER BY
               COUNT(*) DESC
         ) AS item_rank
      FROM
         customer_shopping
      GROUP BY
         category,
         item_purchased
   )
WHERE
   item_rank <= 3
```

	CATEGORY	ITEM_PURCHASED	TOTAL_PURCHASES	ITEM_RANK
1	Accessories	Jewelry	171	1
2	Accessories	Belt	161	2
3	Accessories	Sunglasses	161	3
4	Clothing	Pants	171	1
5	Clothing	Blouse	171	2
6	Clothing	Shirt	169	3
7	Footwear	Sandals	160	1

```
ORDER BY
   category,
   total_purchases DESC;

-- With CTE

WITH item_count AS (
   SELECT
      category,
      item_purchased,
      COUNT(*) AS total_purchases,
      ROW_NUMBER() OVER(
         PARTITION BY category
         ORDER BY
            COUNT(*) DESC
      ) AS item_rank
   FROM
      customer_shopping
   GROUP BY
      category,
      item_purchased
)
SELECT
   category,
   item_purchased,
   total_purchases,
   item_rank
FROM
   item_count
WHERE
   item_rank <= 3;
```

	CATEGORY	ITEM_PURCHASED	TOTAL_PURCHASES	ITEM_RANK
1	Accessories	Jewelry	171	1
2	Accessories	Belt	161	2
3	Accessories	Sunglasses	161	3
4	Clothing	Pants	171	1
5	Clothing	Blouse	171	2
6	Clothing	Shirt	169	3
7	Footwear	Sandals	160	1

--Q9. Are customers who are repeat buyers (more than 5 previous purchases) also likely to subscribe?

```
SELECT
   subscription_status,
   COUNT(*) AS repeat_buyers
FROM
   customer_shopping
WHERE
   previous_purchases >= 5
GROUP BY
   subscription_status;
```

	SUBSCRIPTION_STATUS	REPEAT_BUYERS
1	Yes	980
2	No	2583

--Q10. What is the revenue contribution of each age group?

```
SELECT
```

```
   age_group,
   SUM(purchase_amount) AS revenue_contribution
FROM
   customer_shopping
GROUP BY
   age_group
ORDER BY
   revenue_contribution DESC;
```

	AGE_GROUP	REVENUE_CONTRIBUTION
1	Young Adult	62143
2	Middle-aged	59197
3	Adult	55978
4	Senior	55763

Power BI DAX

```
AVG_PURCHASE_AMT = AVERAGE(CUSTOMER_SHOPPING[PURCHASE_AMOUNT])

AVG_REVIEW_RATING = AVERAGE(CUSTOMER_SHOPPING[REVIEW_RATING])

TOTAL_CUSTOMER = COUNT(CUSTOMER_SHOPPING[CUSTOMER_ID])
```

Github

Basic prompts:

```
ls
```

 # Shows all files and folders in the current directory.

```
git init
```

 # Creates a new empty Git repository in your folder.

```
git status
```

 # Shows which files are tracked, untracked, or modified.

```
git add .
```

 # Stages all files in the folder for the next commit.

```
git status
```

 # Confirms which files are now staged and ready to commit.

```
git commit -m "first commit"
```

 # Saves the staged changes with a message.

```
git branch -M main
```

 # Renames the current branch to 'main' (forcefully).

```
git remote add origin
https://github.com/[your_acc_name]/customer_shopping_behavior_analysis_py
thon_sql_powerbi.git
```

 # Connects your local repository to a GitHub remote repo.

```
git push -u origin main
```

 # Uploads your local 'main' branch to GitHub and sets it as default.

Using " .gitignore " (if you want to keep files locally but exclude them from Git)

```
nano .gitignore
```

 # Opens or creates " .gitignor " file in the folder.

#Add entries which you want locally but not on repo like:

.ipynb_checkpoints/
pycache/
logs/
data/

Ctrl + O – Saves the file

Ctrl + X – Exits nano editor

Unstage / Stop tracking a file or folder (without deleting locally)

```
git rm --cached <filename>
```

Removes the file or folder from Git tracking but keeps it on your system.