Customer Behavior Analysis

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
df = pd.read_csv('/content/drive/MyDrive/CustomerBehavior.csv')
```

```python
df.head(10)
```

	Order Date Time	Order ID	Purchased Product ID	Quantity of SKU in the order	Category ID	Category alias	Brand ID	Price	User ID	Gender	Color	Metal	Main gem
0	2018-12-01 11:40:29 UTC	1.920000e+18	1.840000e+18	1	1.810000e+18	jewelry.earring	0.0	561.51	1.520000e+18	NaN	red	gold	diamond
1	2018-12-01 17:38:31 UTC	1.920000e+18	1.810000e+18	1	1.810000e+18	NaN	NaN	212.14	1.520000e+18	NaN	yellow	gold	NaN
2	2018-12-02 13:53:42 UTC	1.930000e+18	1.840000e+18	1	1.810000e+18	jewelry.pendant	1.0	54.66	1.520000e+18	f	white	gold	sapphire
3	2018-12-02 17:44:02 UTC	1.930000e+18	1.840000e+18	1	1.810000e+18	jewelry.pendant	0.0	88.90	1.520000e+18	f	red	gold	diamond
4	2018-12-02 21:30:19 UTC	1.930000e+18	1.870000e+18	1	1.810000e+18	jewelry.necklace	0.0	417.67	1.520000e+18	NaN	red	gold	amethyst
5	2018-12-02 22:09:34 UTC	1.930000e+18	1.840000e+18	1	1.810000e+18	jewelry.earring	1.0	102.27	1.520000e+18	NaN	red	gold	NaN
6	2018-12-02 22:09:34 UTC	1.930000e+18	1.600000e+18	1	1.810000e+18	jewelry.earring	0.0	150.55	1.520000e+18	NaN	red	gold	diamond

Data Cleaning

```python
df.dtypes
```

	0
Order Date Time	object
Order ID	float64
Purchased Product ID	float64
Quantity of SKU in the order	int64
Category ID	float64
Category alias	object
Brand ID	float64
Price	float64
User ID	float64
Gender	object
Color	object
Metal	object
Main gem	object

dtype: object

```python
#Detect Type
```

```
df['Category ID'] =df['Category ID'].astype(str)
df['Order ID'] =df['Order ID'].astype(str)
df['Purchased Product ID'] =df['Purchased Product ID'].astype(str)
df['Brand ID'] =df['Brand ID'].astype(str)
df['User ID'] =df['User ID'].astype(str)
df['Order Date Time'] =pd.to_datetime(df['Order Date Time'])
```

```
df.dtypes
```

```
                                                 0
              Order Date Time    datetime64[ns, UTC]
                     Order ID                 object
         Purchased Product ID                 object
 Quantity of SKU in the order                  int64
                  Category ID                 object
               Category alias                 object
                     Brand ID                 object
                        Price                float64
                      User ID                 object
                       Gender                 object
                        Color                 object
                        Metal                 object
                     Main gem                 object

dtype: object
```

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

```
                                   0
              Order Date Time      0
                     Order ID      0
         Purchased Product ID      0
 Quantity of SKU in the order      0
                  Category ID      0
               Category alias   9933
                     Brand ID      0
                        Price   5352
                      User ID      0
                       Gender  48168
                        Color   7660
                        Metal   5462
                     Main gem  34058

dtype: int64
```

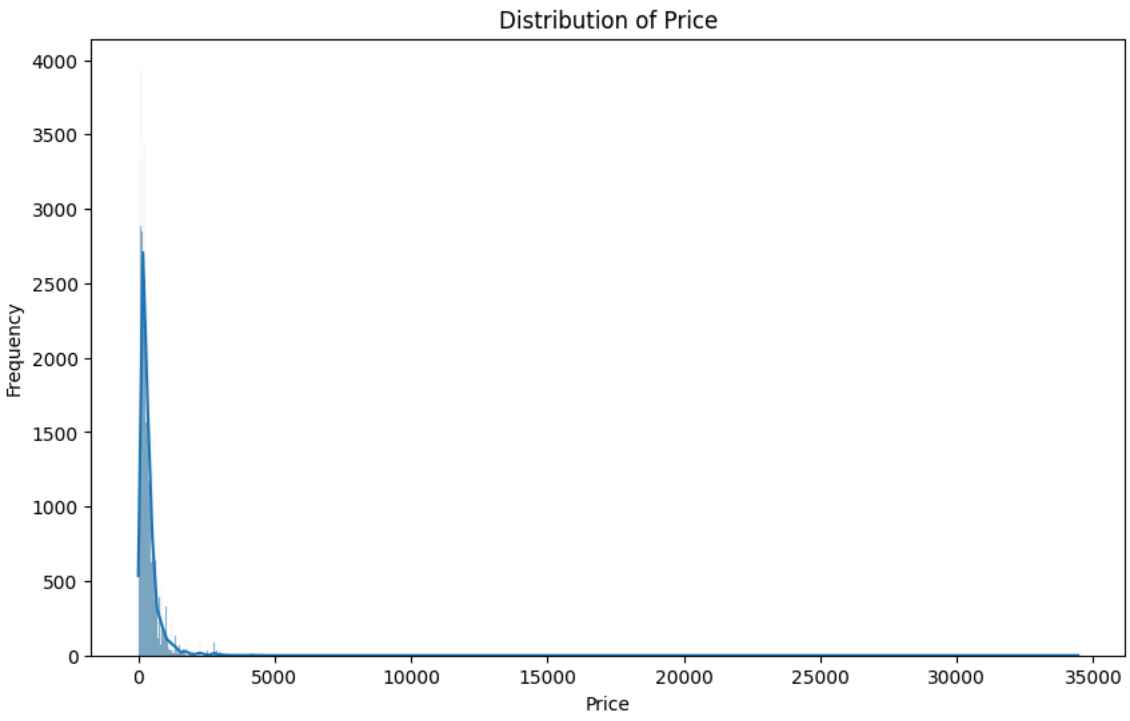
```
df.shape[0]
```

```
95911
```

```
# Visualize the distribution of the 'Price' column
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'].dropna(), kde=True)
plt.title('Distribution of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

```

# Calculate descriptive statistics for the 'Price' column
print("\nDescriptive Statistics for Price:")
print(df['Price'].describe())
```



```
Descriptive Statistics for Price:
count    90559.000000
mean       362.215217
std        444.155706
min          0.990000
25%        145.620000
50%        258.770000
75%        431.370000
max      34448.600000
Name: Price, dtype: float64
```

```
#Data is Skewed ->replace with median
df['Price']=df['Price'].fillna(df['Price'].median())
df['Price'].isnull().sum()
```

```
np.int64(0)
```

```
df['Category alias'] =df['Category alias'].fillna('UnKown')
df['Gender'] =df['Gender'].fillna('UnKown')
df['Color'] = df['Color'].fillna('UnKown')
df['Metal'] = df['Metal'].fillna('UnKown')
df['Main gem']= df['Main gem'].fillna('UnKown')

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

	0
Order Date Time	0
Order ID	0
Purchased Product ID	0
Quantity of SKU in the order	0
Category ID	0
Category alias	0
Brand ID	0
Price	0
User ID	0
Gender	0
Color	0
Metal	0
Main gem	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(2956)
```

```
df.drop_duplicates(inplace=True)
df.duplicated().sum()
```

```
np.int64(0)
```

RFM Analysis & Customer Segmentation

```
df = df[df['Quantity of SKU in the order'] > 0]
df['Revenue'] = df['Price']*df['Quantity of SKU in the order']
df[['Quantity of SKU in the order','Price','Revenue']]
```

	Quantity of SKU in the order	Price	Revenue
0	1	561.51	561.51
1	1	212.14	212.14
2	1	54.66	54.66
3	1	88.90	88.90
4	1	417.67	417.67
...	...	...	...
95906	1	258.77	258.77
95907	1	229.99	229.99
95908	1	630.00	630.00
95909	1	83.42	83.42
95910	1	97.12	97.12

95911 rows × 3 columns

```
#RFM_Analysis
reference_date = df['Order Date Time'].max() + pd.Timedelta(days=1)

rfm = df.groupby('User ID').agg({
    'Order Date Time' : lambda x : reference_date - x.max()    #R
    ,'Order ID' : 'nunique'                                    #F
    ,'Revenue' : 'sum'                                         #M
}).reset_index()
```

```python

rfm.columns = ['User ID','Recency','Frequency','Monetary']

rfm['R_Score'] = pd.qcut(rfm['Recency'],5,labels=[5,4,3,2,1])
rfm['F_Score'] = pd.qcut(rfm['Frequency'].rank(method='first'),5,labels=[1,2,3,4,5])
rfm['M_Score'] = pd.qcut(rfm['Monetary'],5,labels=[1,2,3,4,5])

rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) + rfm['M_Score'].astype(str)
rfm.head()
```

	User ID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score
0	1.31e+18	21 days 05:05:56	58	101895.68	3	4	4	344
1	1.32e+18	1 days 02:00:29	73	150777.29	5	5	5	555
2	1.33e+18	10 days 23:53:27	68	118626.97	3	4	4	344
3	1.34e+18	45 days 08:31:30	38	47488.03	2	3	3	233
4	1.35e+18	22 days 17:11:37	34	19918.57	3	2	1	321

```python
#Customer_Segmentation
def Segment(row):

  if row['RFM_Score'] =='555':
    return 'Champion'
  elif row['R_Score'] in ['4','5'] and row['F_Score'] in ['4','5'] :
    return 'Loyal'
  elif row['R_Score'] in ['1','2'] and row['F_Score'] in ['4','5'] :
    return 'At_Risk'
  elif row['R_Score'] in ['1','2'] and row['F_Score'] in ['1','2'] :
    return 'Lost_Customer'
  else:
    return 'Others'

rfm['Customer_Segment'] = rfm.apply(Segment,axis=1)

rfm
```

	User ID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score	Customer_Segment
0	1.31e+18	21 days 05:05:56	58	101895.68	3	4	4	344	Others
1	1.32e+18	1 days 02:00:29	73	150777.29	5	5	5	555	Champion
2	1.33e+18	10 days 23:53:27	68	118626.97	3	4	4	344	Others
3	1.34e+18	45 days 08:31:30	38	47488.03	2	3	3	233	Others
4	1.35e+18	22 days 17:11:37	34	19918.57	3	2	1	321	Others
5	1.36e+18	110 days 19:52:10	18	6973.04	1	1	1	111	Others
6	1.37e+18	8 days 14:03:42	39	40287.62	4	3	3	433	Others
7	1.38e+18	42 days 02:00:05	11	4435.59	2	1	1	211	Others
8	1.45e+18	3 days 20:12:06	54	75359.78	4	4	4	444	Others
9	1.46e+18	1 days 16:49:06	71	134631.57	4	5	5	455	Others
10	1.47e+18	42 days 15:41:23	47	62212.01	2	3	3	233	Others
11	1.48e+18	128 days 12:28:17	34	23874.24	1	2	2	122	Others
12	1.49e+18	2 days 22:04:28	37	34199.30	4	3	2	432	Others
13	1.51e+18	137 days 11:27:40	24	17117.58	1	1	1	111	Others
14	1.52e+18	1 days 00:00:00	81	30957222.30	5	5	5	555	Champion
15	1.53e+18	23 days 00:17:04	23	33798.78	2	1	2	212	Others
16	1.54e+18	401 days 15:04:57	27	45027.17	1	2	3	123	Others
17	1.55e+18	1 days 04:26:38	69	116981.43	5	4	4	544	Others
18	1.5e+18	20 days 19:26:28	31	22546.58	3	2	2	322	Others
19	nan	1 days 00:11:30	70	1278582.57	5	5	5	555	Champion

Sales

```
df
```

	Order Date Time	Order ID	Purchased Product ID	Quantity of SKU in the order	Category ID	Category alias	Brand ID	Price	User ID	Gender	Color	Metal	Main gem
0	2018-12-01 11:40:29+00:00	1.92e+18	1.84e+18	1	1.81e+18	jewelry.earring	0.0	561.51	1.52e+18	UnKown	red	gold	diamond
1	2018-12-01 17:38:31+00:00	1.92e+18	1.81e+18	1	1.81e+18	UnKown	nan	212.14	1.52e+18	UnKown	yellow	gold	UnKown
2	2018-12-02 13:53:42+00:00	1.93e+18	1.84e+18	1	1.81e+18	jewelry.pendant	1.0	54.66	1.52e+18	f	white	gold	sapphire
3	2018-12-02 17:44:02+00:00	1.93e+18	1.84e+18	1	1.81e+18	jewelry.pendant	0.0	88.90	1.52e+18	f	red	gold	diamond
4	2018-12-02 21:30:19+00:00	1.93e+18	1.87e+18	1	1.81e+18	jewelry.necklace	0.0	417.67	1.52e+18	UnKown	red	gold	amethyst
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95906	2021-12-01 09:47:37+00:00	2.72e+18	1.52e+18	1	nan	0	1.52e+18	258.77	nan	UnKown	UnKown	UnKown	UnKown
95907	2021-12-01 09:50:37+00:00	2.72e+18	1.96e+18	1	1.81e+18	UnKown	0.0	229.99	1.52e+18	UnKown	red	gold	UnKown
95908	2021-12-01 09:52:06+00:00	2.72e+18	1.96e+18	1	1.81e+18	jewelry.earring	1.0	630.00	1.52e+18	f	white	gold	diamond
95909	2021-12-01 09:55:35+00:00	2.72e+18	1.84e+18	1	1.81e+18	jewelry.earring	0.0	83.42	1.52e+18	UnKown	white	gold	fianit
95910	2021-12-01 09:59:07+00:00	2.72e+18	1.96e+18	1	1.81e+18	jewelry.pendant	1.0	97.12	1.52e+18	UnKown	red	gold	UnKown

92955 rows × 13 columns

```
count_of_orders = df['Order ID'].nunique()
total_quantity = df['Quantity of SKU in the order'].sum()
avg_quantity = df['Quantity of SKU in the order'].mean()
```

```
total_revenue = df['Revenue'].sum()
avg_total_price = df['Price'].mean()

print("Number of Orders:", count_of_orders)
print("Total Quantity:", total_quantity)
print("Average Quantity:", avg_quantity)
print("Total Revenue:", total_revenue)
print("Average Total Price:", avg_total_price)
```

```
Number of Orders: 81
Total Quantity: 92955
Average Quantity: 1.0
Total Revenue: 33291956.099999998
Average Total Price: 358.1513216072293
```

```
df['Year'] = df['Order Date Time'].dt.year
sales_per_year = df.groupby('Year')['Revenue'].sum()
sales_per_year
```

```
           Revenue
Year
2018      33143.57
2019    4136225.75
2020    9042261.20
2021   20080325.58

dtype: float64
```

```
df['Month'] = df['Order Date Time'].dt.month
sales_per_month = df.groupby('Month')['Revenue'].sum()
sales_per_month
```

```
          Revenue
Month
  1    1888248.28
  2    1963446.45
  3    2247752.74
  4    1698547.76
  5    1738255.59
  6    2020141.14
  7    2675748.23
  8    4337431.41
  9    3069369.17
 10    2987954.16
 11    5596249.47
 12    3068811.70

dtype: float64
```

```
df['Hour'] = df['Order Date Time'].dt.hour
most_hour_in_sales = df['Hour'].mode()[0]
int(most_hour_in_sales)
```

```
11
```

Product

```python
num_of_products=df['Purchased Product ID'].nunique()
num_of_categories=df['Category ID'].nunique()
most_color = df['Color'].mode()[0]
most_metal = df['Metal'].mode()[0]
most_main_gem = df['Main gem'].mode()[0]


print("Number of Products:", num_of_products)
print("Number of Categories:", num_of_categories)
print("Most Color:", most_color)
print("Most Metal:", most_metal)
print("Most Main Gem:", most_main_gem)
```

```
Number of Products: 103
Number of Categories: 3
Most Color: red
Most Metal: gold
Most Main Gem: UnKown
```

```python
most_product_selling = df['Purchased Product ID'].mode()[0]
product_details = df[df['Purchased Product ID'] == most_product_selling]
product_details.head(1)
```

	Order Date Time	Order ID	Purchased Product ID	Quantity of SKU in the order	Category ID	Category alias	Brand ID	Price	User ID	Gender	Color	Metal	Main gem	Revenue	Year	Month
148	2019-01-13	1.96e+18	1.96e+18	1	1.81e+18	jewelry.pendant	1.0	123.15	1.52e+18	f	red	gold	diamond	123.15	2019	1

```python
Category_sales = df.groupby('Category alias')['Revenue'].sum()
Category_sales.sort_values(ascending=False).head(5)
```

	Revenue
Category alias	
jewelry.earring	11503531.02
jewelry.ring	9679520.82
UnKown	4336725.60
jewelry.bracelet	3020484.70
jewelry.pendant	2198554.30

dtype: float64

Customer

```python
df
```

	Order Date Time	Order ID	Purchased Product ID	Quantity of SKU in the order	Category ID	Category alias	Brand ID	Price	User ID	Gender	Color	Metal	Main gem	Revenue	Yea
0	2018-12-01 11:40:29+00:00	1.92e+18	1.84e+18	1	1.81e+18	jewelry.earring	0.0	561.51	1.52e+18	UnKown	red	gold	diamond	561.51	201
1	2018-12-01 17:38:31+00:00	1.92e+18	1.81e+18	1	1.81e+18	UnKown	nan	212.14	1.52e+18	UnKown	yellow	gold	UnKown	212.14	201
2	2018-12-02 13:53:42+00:00	1.93e+18	1.84e+18	1	1.81e+18	jewelry.pendant	1.0	54.66	1.52e+18	f	white	gold	sapphire	54.66	201
3	2018-12-02 17:44:02+00:00	1.93e+18	1.84e+18	1	1.81e+18	jewelry.pendant	0.0	88.90	1.52e+18	f	red	gold	diamond	88.90	201
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95906	2021-12-01	2.72e+18	1.52e+18	1	nan	0	1.52e+18	258.77	nan	UnKown	UnKown	UnKown	UnKown	258.77	202
95908	2021-12-01 09:52:06+00:00	2.72e+18	1.96e+18	1	1.81e+18	jewelry.earring	1.0	630.00	1.52e+18	f	white	gold	diamond	630.00	202
	2021-12-01 00:00	2.72e+18	1.84e+18	1	1.81e+18	jewelry.earring	0.0	83.42	1.52e+18	UnKown	white	gold	fianit	83.42	202
95910	2021-12-01 09:50:07+00:00	2.72e+18	1.96e+18	1	1.81e+18	jewelry.pendant	1.0	97.12	1.52e+18	UnKown	red	gold	UnKown	97.12	202

92955 rows × 17 columns

```
num_of_customers=df['User ID'].nunique()
num_of_customers
```

```
20
```

```
gender_values = df['Gender'].value_counts()
gender_values
```

Gender	count
f	46655
UnKown	45941
m	359

dtype: int64

```
avg_revenue_gender = (
    df.groupby('Gender')
      .agg(avg_revenue_per_customer=('Revenue', 'mean'))
)

print("Average revenue per customer by Gender:")
print(avg_revenue_gender)
```