Customer Transactions Analysis with Python

Problem Statement

In this Jupyter notebook, I conducted a thorough analysis of the "Customer Transactions" dataset, covering key aspects like demographics, transaction details, and merchant-related data. The analysis included preprocessing steps, descriptive statistics, and dynamic visualizations to uncover patterns in transaction amounts and customer behavior. Utilizing Python's Pandas and Matplotlib, the notebook provides valuable insights into transaction trends and behaviors, making it a comprehensive exploration of the dataset.

Import Library

In [1]:
```
import pandas as pd
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\Syed Arif\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy
version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.25.
1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Uploading Csv fle

In [3]:
```
df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\Customer transection\Customer Transactions.csv
```

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

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

Out[4]:

	Customer ID	Name	Surname	Gender	Birthdate	Transaction Amount	Date	Merchant Name	Category
0	752858	Sean	Rodriguez	F	2002-10-20	35.47	2023-04-03	Smith-Russell	Cosmetic
1	26381	Michelle	Phelps	NaN	1985-10-24	2552.72	2023-07-17	Peck, Spence and Young	Travel
2	305449	Jacob	Williams	M	1981-10-25	115.97	2023-09-20	Steele Inc	Clothing
3	988259	Nathan	Snyder	M	1977-10-26	11.31	2023-01-11	Wilson, Wilson and Russell	Cosmetic
4	764762	Crystal	Knapp	F	1951-11-02	62.21	2023-06-13	Palmer-Hinton	Electronics

.tail()

tail is used to show rows by Descending order

In [5]:
```
df.tail()
```

Out[5]:

	Customer ID	Name	Surname	Gender	Birthdate	Transaction Amount	Date	Merchant Name	Category
49995	891845	Christine	Leach	F	1997-10-21	108.74	2023-08-30	Alexander Ltd	Market
49996	800560	Anna	Allen	F	1999-10-21	133.66	2023-05-03	Knapp-Calhoun	Cosmetic
49997	133285	Nicole	Franklin	M	1979-10-26	464.29	2023-02-12	Cantrell, Haynes and Ballard	Market
49998	616122	Maria	Keller	M	1981-10-25	472.57	2023-03-25	Wilson, Jackson and Beard	Market
49999	832184	Billy	Walker	F	1958-10-31	270.67	2023-05-20	Combs LLC	Market

.shape

It show the total no of rows & Column in the dataset

In [6]:
```
df.shape
```

Out[6]:
```
(50000, 9)
```

.Columns

It show the no of each Column

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Customer ID', 'Name', 'Surname', 'Gender', 'Birthdate',
       'Transaction Amount', 'Date', 'Merchant Name', 'Category'],
      dtype='object')
```

.dtypes

This Attribute show the data type of each column

In [8]:
```
df.dtypes
```

Out[8]:
```
Customer ID            int64
Name                  object
Surname               object
Gender                object
Birthdate             object
Transaction Amount   float64
Date                  object
Merchant Name         object
Category              object
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [9]:
```
df["Name"].unique()
```

Out[9]:
```
array(['Sean', 'Michelle', 'Jacob', 'Nathan', 'Crystal', 'Monica',
       'Thomas', 'Kelsey', 'Denise', 'Alexander', 'Kimberly', 'Hunter',
       'Julie', 'Christopher', 'Amber', 'Amy', 'Elizabeth', 'Chad',
       'Steven', 'Patrick', 'Daniel', 'Robert', 'Sara', 'Anthony',
       'Jessica', 'Brian', 'William', 'Adam', 'Betty', 'Pamela', 'Sarah',
       'Manuel', 'Aaron', 'Darlene', 'Hannah', 'Benjamin', 'Jason',
       'Scott', 'Emily', 'Katherine', 'Kevin', 'Jose', 'Meghan',
       'Michael', 'Dana', 'Lisa', 'Sydney', 'Melanie', 'Matthew', 'Tammy',
       'Justin', 'Ann', 'Joshua', 'Sabrina', 'Leslie', 'Jeffery', 'Mary',
       'Ashley', 'Joseph', 'Juan', 'Christina', 'Martha', 'John', 'Keith',
       'Alejandro', 'David', 'Jennifer', 'Nicholas', 'Christy', 'Jamie',
       'Douglas', 'Holly', 'Fernando', 'Yvonne', 'Margaret', 'Charles',
       'Derrick', 'Kurt', 'Timothy', 'Courtney', 'Joy', 'Carol',
       'Lawrence', 'Andrew', 'Nicole', 'Stephanie', 'Derek', 'Hector',
       'Jesse', 'Rebecca', 'Catherine', 'Andres', 'Lindsay', 'Raymond',
       'Alison', 'Michele', 'Patricia', 'Samuel', 'Diane', 'Carrie',
       'Shannon', 'Carolyn', 'Travis', 'Regina', 'Wayne', 'Selena',
       'Bianca', 'Brenda', 'Anita', 'Deborah', 'Erika', 'Madison',
       'Kristin', 'James', 'Mark', 'Kyle', 'Harold', 'Sherri', 'Cameron',
```

.nuique()

It will show the total no of unque value from whole data frame

In [10]:
```
df.nunique()
```

Out[10]:
```
Customer ID           50000
Name                    690
Surname                1000
Gender                    2
Birthdate                58
Transaction Amount    34665
Date                    287
Merchant Name         36939
Category                  6
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [11]:
```
df.describe()
```

Out[11]:

	Customer ID	Transaction Amount
count	50000.00000	50000.000000
mean	500136.79696	442.119239
std	288232.43164	631.669724
min	29.00000	5.010000
25%	251191.50000	79.007500
50%	499520.50000	182.195000
75%	749854.25000	470.515000
max	999997.00000	2999.880000

.value_counts

It Shows all the unique values with their count

In [12]:
```
df["Name"].value_counts()
```

Out[12]:
```
Michael        1167
David           761
John            749
James           730
Christopher     711
               ...
Terrance          4
Preston           4
Perry             3
Latasha           2
Leon              1
Name: Name, Length: 690, dtype: int64
```

.isnull()

It shows the how many null values

In [13]:
```
df.isnull()
```

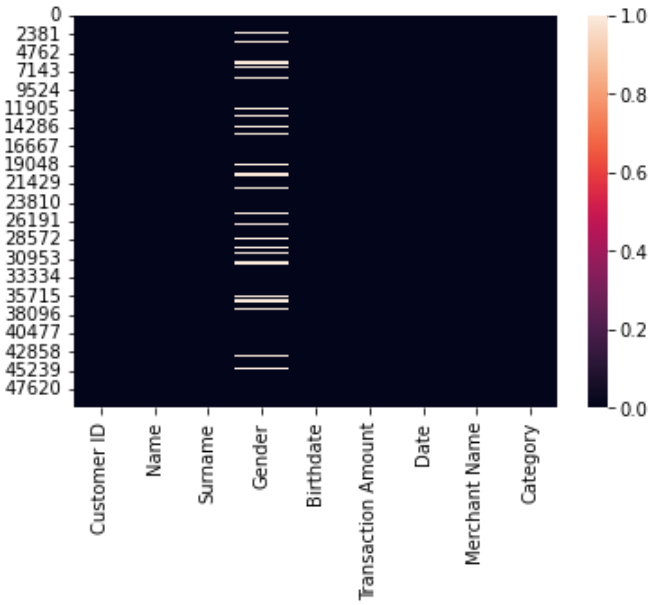
Out[13]:

	Customer ID	Name	Surname	Gender	Birthdate	Transaction Amount	Date	Merchant Name	Category
0	False	False	False	False	False	False	False	False	False
1	False	False	False	True	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
49995	False	False	False	False	False	False	False	False	False
49996	False	False	False	False	False	False	False	False	False
49997	False	False	False	False	False	False	False	False	False
49998	False	False	False	False	False	False	False	False	False
49999	False	False	False	False	False	False	False	False	False

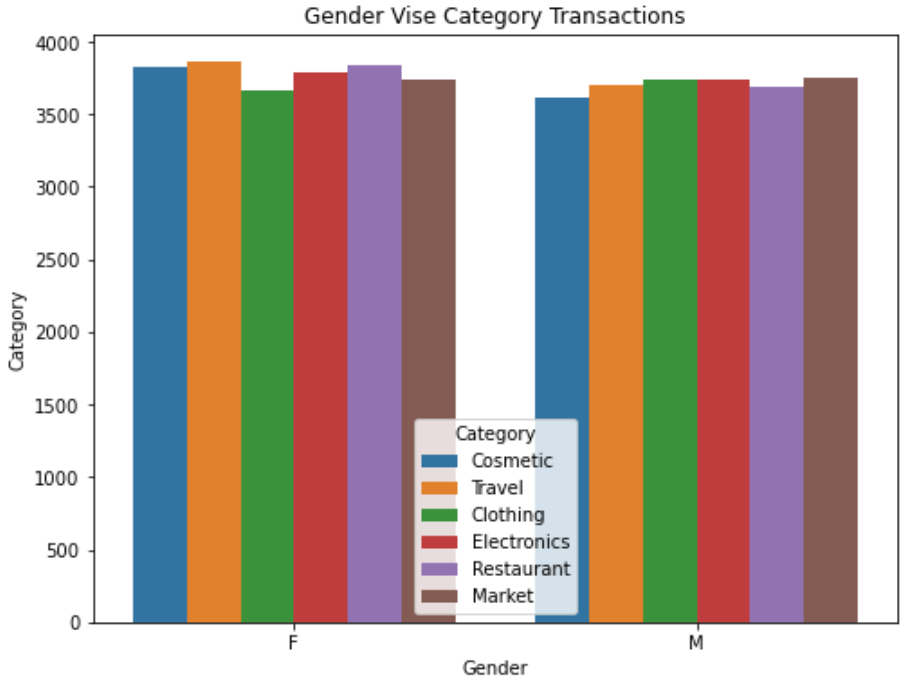
50000 rows × 9 columns

In [14]:
```
sns.heatmap(df.isnull())
```

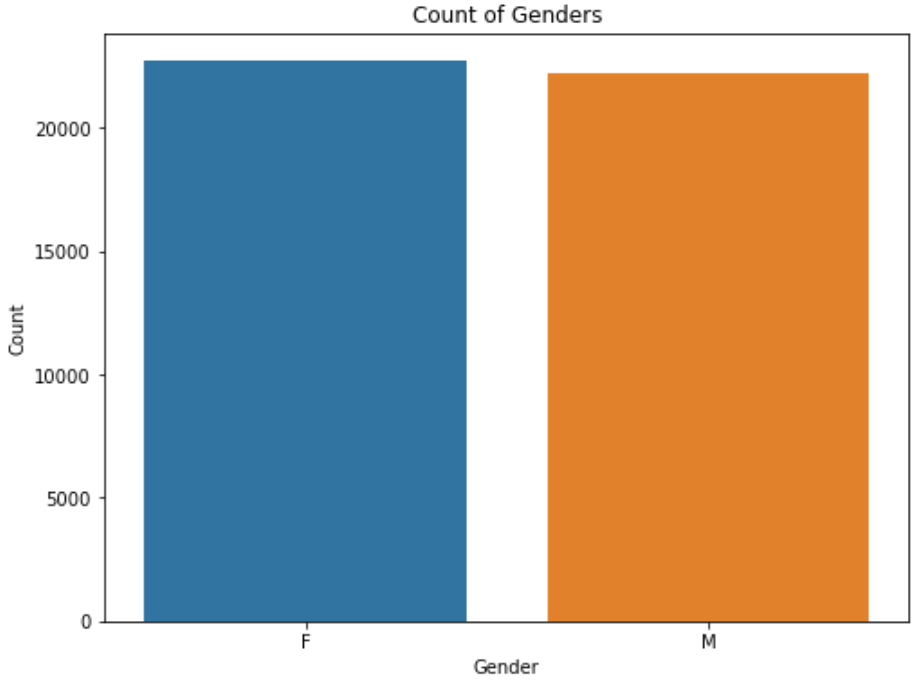
Out[14]: <AxesSubplot:>



In [16]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender', hue ="Category")
plt.xlabel('Gender')
plt.ylabel('Category')
plt.title('Gender Vise Category Transactions')
plt.show()
```

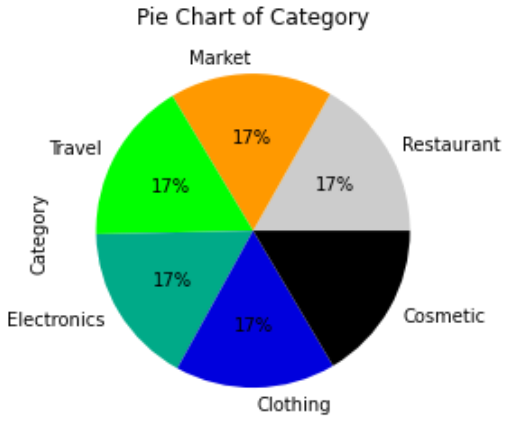


In [17]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Count of Genders')
plt.show()
```



In [18]:
```
df['Category'].value_counts().plot(kind = 'pie' , title = 'Pie Chart of Category',
                                   autopct="%.0f%%", colormap='nipy_spectral_r')
```

Out[18]:
```
<AxesSubplot:title={'center':'Pie Chart of Category'}, ylabel='Category'>
```



What is the average transaction amount per customer?

In [19]:
```
avg_transaction_per_customer = df.groupby('Customer ID')['Transaction Amount'].mean().reset_
print("Average Transaction Amount per Customer:\n", avg_transaction_per_customer)
```

```
Average Transaction Amount per Customer:
        Customer ID  Transaction Amount
0                29               27.02
1                51             1898.56
2                54              166.30
3                83              125.85
4                90               18.16
...             ...                 ...
49995        999904              266.06
49996        999914              295.11
49997        999942              153.78
49998        999949              636.09
49999        999997               15.47

[50000 rows x 2 columns]
```

Which gender has a higher average transaction amount?

In [20]:
```
avg_transaction_by_gender  = df.groupby('Gender')['Transaction Amount'].mean().reset_index(
print("Average Transaction Amount by Gender:\n", avg_transaction_by_gender)
```

```
Average Transaction Amount by Gender:
   Gender  Transaction Amount
0      F          445.521078
1      M          440.417393
```

How many transactions occur on weekdays vs. weekends?

In [21]:
```
# Assuming 'df' is your DataFrame
df['Birthdate'] = pd.to_datetime(df['Birthdate'])
df.set_index('Birthdate', inplace=True)  # Set 'Birthdate' as the datetime index

df['Day_of_Week'] = df.index.day_name()

transaction_counts_by_day = df['Day_of_Week'].value_counts()

# Print the transaction counts
print("Transaction Counts by Day:\n", transaction_counts_by_day)
```

```
Transaction Counts by Day:
 Tuesday      7854
Wednesday    7725
Monday       6966
Thursday     6899
Saturday     6878
Friday       6846
Sunday       6832
Name: Day_of_Week, dtype: int64
```

In [22]:
```
transaction_counts_by_day = df['Day_of_Week'].value_counts()
transaction_counts_by_day.plot(kind='bar', color='cyan')
plt.xlabel('Day of Week')
plt.ylabel('Count')
plt.title('Transaction Counts by Day')
```

Out[22]:
```
Text(0.5, 1.0, 'Transaction Counts by Day')
```