Import Libraries to perform specifi c task such as data manipulation, data analysis, numerical computations, creatingvarious types of plots, statistical visualizations of plots.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Data (Transaction) and save in variable callad "tran".

```
tran = pd.read_csv("/content/transaction.csv")
tran
```

	transaction_id	product_id	customer_id	transaction_date	online_order	ord
0	1	2	2950	25/02/2017	False	
1	2	3	3120	21/05/2017	True	
2	3	37	402	16/10/2017	False	
3	4	88	3135	31/08/2017	False	
4	5	78	787	01/10/2017	True	
...	...	...	...	...	...	
19995	19996	51	1018	24/06/2017	True	

Some Basic Operations.

```
# Following command gives whole information of given dataset.

tran.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   transaction_id           20000 non-null  int64  
 1   product_id               20000 non-null  int64  
 2   customer_id              20000 non-null  int64  
 3   transaction_date         20000 non-null  object 
 4   online_order             19640 non-null  object 
 5   order_status             20000 non-null  object 
 6   brand                    19803 non-null  object 
 7   product_line             19803 non-null  object 
 8   product_class            19803 non-null  object 
 9   product_size             19803 non-null  object 
 10  list_price               20000 non-null  float64
 11  standard_cost            19803 non-null  object 
 12  product_first_sold_date  19803 non-null  float64
dtypes: float64(2), int64(3), object(8)
memory usage: 2.0+ MB
```

```
# Following command calculates order pair of given dataset.

tran.shape
```

```
(20000, 13)
```

```
# Following command tells if there is any missing valiue in given dataset.

tran.isnull().sum()
```

```
transaction_id              0
product_id                  0
customer_id                 0
transaction_date            0
online_order              360
order_status                0
brand                     197
product_line              197
product_class             197
product_size              197
list_price                  0
standard_cost             197
product_first_sold_date   197
dtype: int64
```

```
# Following command drops missing values of given dataset.

tran.dropna(inplace = True)
tran.isnull().sum()
```

```
transaction_id            0
product_id                0
customer_id               0
transaction_date          0
online_order              0
order_status              0
brand                     0
product_line              0
product_class             0
product_size              0
list_price                0
standard_cost             0
product_first_sold_date   0
dtype: int64
```

```
# Following command calculate order pair of given dataset.
# After droping missing values.

tran.shape
```

```
(19445, 13)
```

```
# Following command gives column names of entire dataset.

tran.columns
```

```
Index(['transaction_id', 'product_id', 'customer_id', 'transaction_date',
       'online_order', 'order_status', 'brand', 'product_line',
       'product_class', 'product_size', 'list_price', 'standard_cost',
       'product_first_sold_date'],
      dtype='object')
```

```
# Following command Explore "online_order" column.

tran.online_order.value_counts()
```

```
True     9739
False    9706
Name: online_order, dtype: int64
```

```
# Following command Explore "order_status" column.

tran.order_status.value_counts()
```

```
Approved     19273
Cancelled      172
Name: order_status, dtype: int64
```

```
# Following command Explore "product_class" column.

tran.product_class.value_counts()
```

```
medium    13587
high       2952
low        2906
Name: product_class, dtype: int64
```

```
# Following command Explore "product_size" column.

tran.product_size.value_counts()
```

```
medium    12767
large      3900
small      2778
Name: product_size, dtype: int64
```

```
# Following command calculate duplicate value in given dataset, if exists.

tran.duplicated().sum()
```

```
0
```

Convert Float to datetime at first_sold column

```
# Following command gives data type of "first_sold" column before converting in given dataset.

tran.product_first_sold_date.dtype
```

```
dtype('float64')
```

```
# Following command replace data type of first_sold column in given dataset.

tran['product_first_sold_date'] = pd.to_datetime(tran['product_first_sold_date'], unit='s')
```

```
# Following command return "first_sold" column after replacing.

tran.product_first_sold_date
```

```
0       1970-01-01 11:27:25
1       1970-01-01 11:35:01
2       1970-01-01 10:06:01
3       1970-01-01 10:02:25
4       1970-01-01 11:43:46
                   ...
19995   1970-01-01 10:30:23
19996   1970-01-01 09:52:40
19997   1970-01-01 11:13:30
19998   1970-01-01 10:36:56
19999   1970-01-01 10:05:34
Name: product_first_sold_date, Length: 19445, dtype: datetime64[ns]
```

```
# Following command gives data type of "first_sold" column after converting in given dataset.

tran.product_first_sold_date.dtype
```

```
dtype('<M8[ns]')
```

Add currency in price_list column of Given dataset

```
# Following command return "list_price" column before editing.

tran.list_price
```

```
0          71.49
1        2091.47
2        1793.43
3        1198.46
4        1765.30
           ...
19995    2005.66
19996     416.98
19997    1636.90
19998     227.88
19999    1775.81
Name: list_price, Length: 19445, dtype: float64
```

```
# Following command edit "list_price" column, add $ sign

tran["list_price"] = tran["list_price"].map("${:,.0f}".format)
```

```
# Following command return "list_price" column after editing.

tran.list_price
```

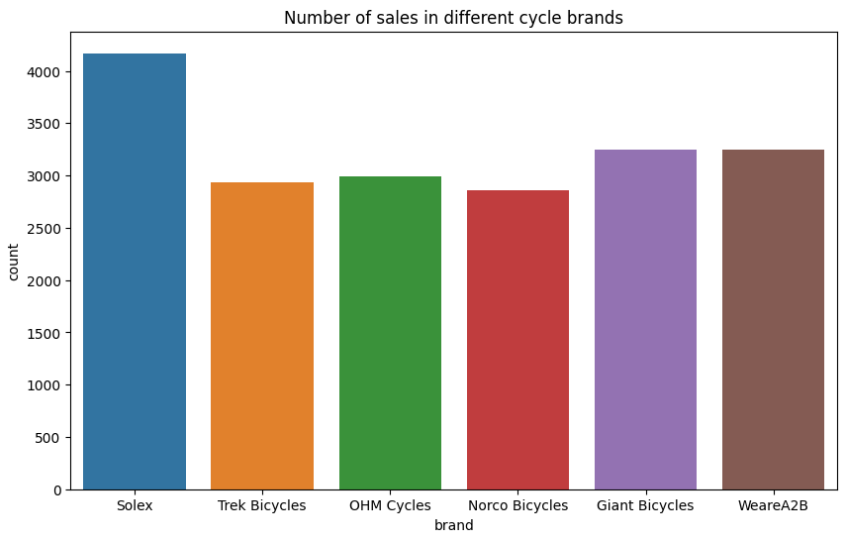
```
0           $71
1        $2,091
2        $1,793
3        $1,198
4        $1,765
```

```
          ...
19995    $2,006
19996      $417
19997    $1,637
19998      $228
19999    $1,776
Name: list_price, Length: 19445, dtype: object
```

Graphical view

```
# Following command shows the graphical view of "Brand" column

plt.figure(figsize=(10,6))
count = sns.countplot(x= tran['brand'], data=tran)
plt.title("Number of sales in different cycle brands")
plt.show()
```



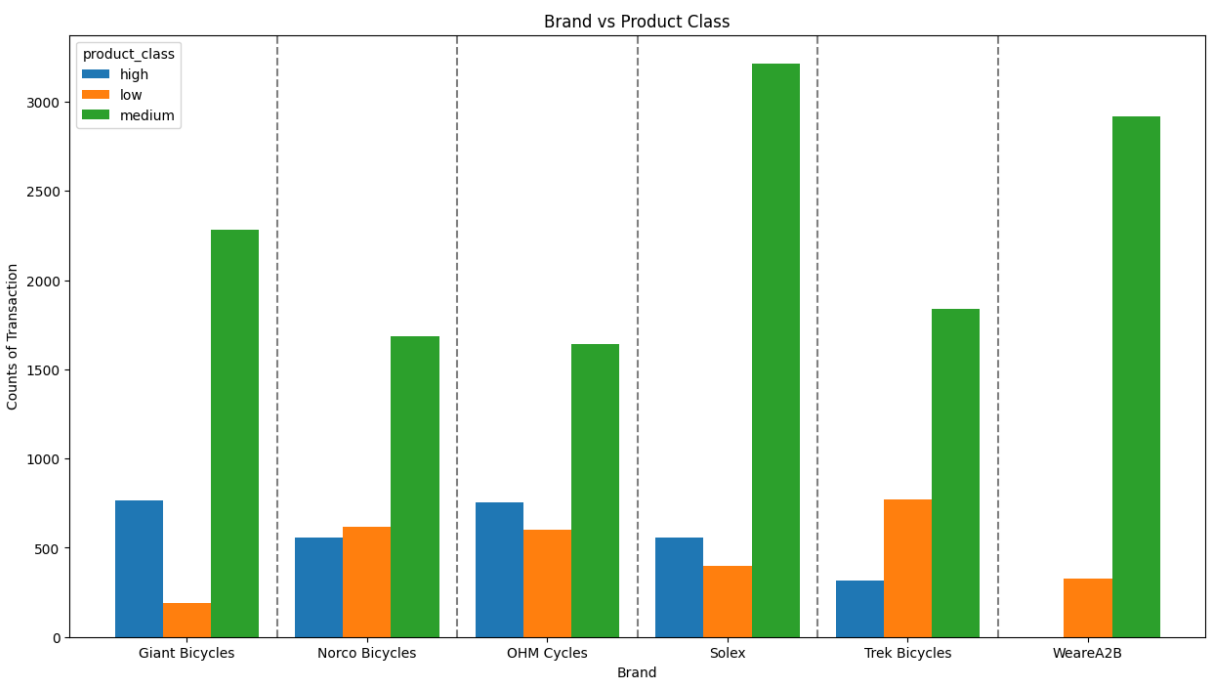

```
# Following commands shows the relationship between different brands with product line

counts = tran.groupby(['brand', 'product_line']).size().reset_index(name='Count')
pivot_counts = counts.pivot(index='brand', columns='product_line', values='Count')
# Create a bar plot
pivot_counts.plot(kind='bar', figsize=(15, 8), width=0.8)
# Creating a line
separation_lines = [1, 2, 3, 4, 5]
for line in separation_lines:
    plt.axvline(x=line - 0.5, color='grey', linestyle='--')
plt.xlabel('Brand')
plt.ylabel('Counts of Transaction')
plt.title('Brand vs Product line')
plt.xticks(rotation = 0)
plt.show()
```



```
# Following commands shows the relationship between different brands with product class

counts = tran.groupby(['brand', 'product_class']).size().reset_index(name='Count')
# Pivot the data for plotting
pivot_counts =counts.pivot(index='brand', columns='product_class', values='Count')
# Create a bar plot
pivot_counts.plot(kind='bar', figsize=(15, 8), width=0.8)
# Creating a line
separation_lines = [1, 2, 3, 4, 5]
for line in separation_lines:
    plt.axvline(x=line - 0.5, color='grey', linestyle='--')
# Set labels and title
plt.xlabel('Brand')
plt.ylabel('Counts of Transaction')
plt.title('Brand vs Product Class')
plt.xticks(rotation = 0)
plt.show()
```



✓ 0s completed at 6:49 PM