Data Loading and Cleaning

```
import numpy as np
import pandas as pd
```

```
data = pd.read_excel("/content/New-york-city-bike-raw-data.xlsx")
```

```
data.head()
```

	Start Time	Stop Time	Start Station ID	Start Station Name	End Station ID	End Station Name	Bike ID	User Type	Birth Year	Age	Age Groups	Trip Duration	Trip_Duration_in_min	Month	Seas
0	2017-01-01 00:38:00	2017-01-01 01:03:00	3194	McGinley Square	3271	Danforth Light Rail	24668	Subscriber	1961	60	55-64	1513	25	1	Win
1	2017-01-01 01:47:00	2017-01-01 01:58:00	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	28	25-34	639	11	1	Win
2	2017-01-01 01:47:00	2017-01-01 01:58:00	3183	Exchange Place	3203	Hamilton Park	26167	Subscriber	1993	28	25-34	639	11	1	Win
3	2017-01-01	2017-01-01	3186	Grove St	3270	Jersey &	24604	Subscriber	1970	51	45-54	258	4	1	Win

```
# Check MISSING values
data.isna().sum()
```

	0
Start Time	0
Stop Time	0
Start Station ID	0
Start Station Name	0
End Station ID	0
End Station Name	1
Bike ID	0
User Type	0
Birth Year	0
Age	0
Age Groups	0
Trip Duration	0
Trip_Duration_in_min	0
Month	0
Season	0
Temperature	0
Weekday	0

dtype: int64

```
# Check DUPLICATE values
data.duplicated().sum()
```

```
3555
```

```
# Drop rows with MISSING values
data_cleaned = data.dropna()
```

```
# Remove DUPLICATEs by dropping duplicate rows
data_cleaned = data_cleaned.drop_duplicates()


# Save the cleaned data to a new Excel file
cleaned_file_path = '/content/Cleaned_Data.xlsx'
data_cleaned.to_excel(cleaned_file_path, index=False)
```

EDA(Exploratory Data Analysis)

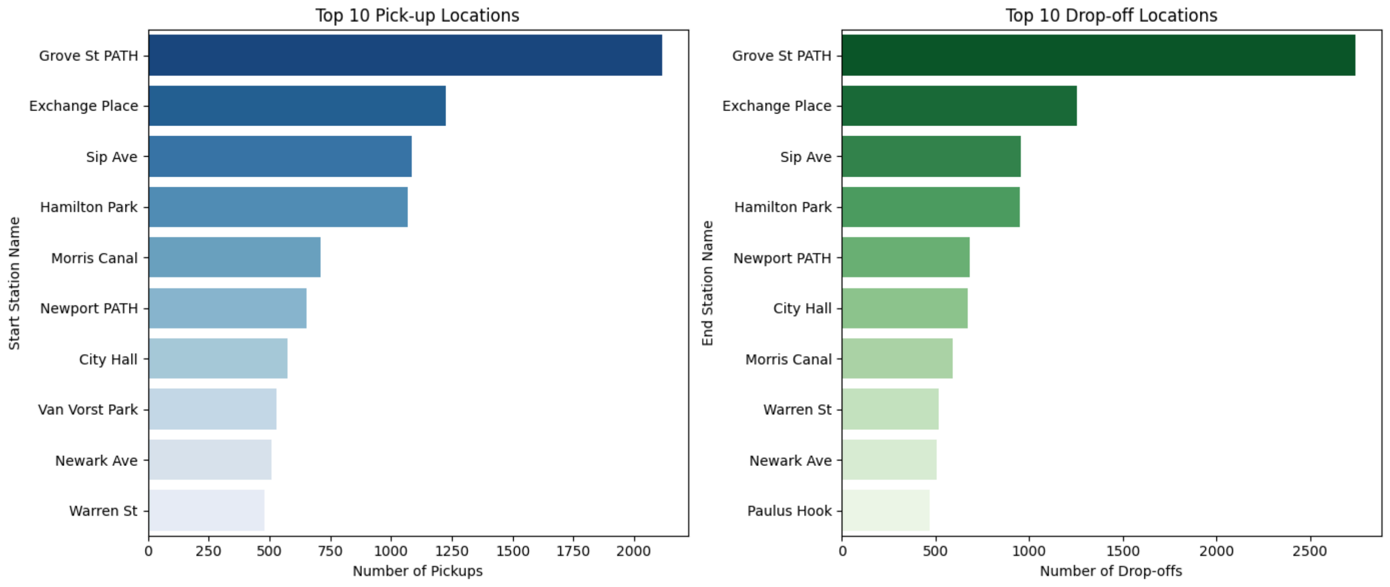
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the cleaned Excel file
file_path = '/content/Cleaned_Data.xlsx'
NY_data = pd.read_excel(file_path)


# Descriptive statistics for numerical columns
print(NY_data.describe())

# Summary for categorical columns
print(NY_data.describe(include=['object']))
```

```
                          Start Time                      Stop Time  \
count                          16844                          16844
mean   2017-02-19 21:12:05.218475520  2017-02-19 21:21:40.693422080
min              2017-01-01 00:38:00            2017-01-01 01:03:00
25%              2017-01-29 18:06:15            2017-01-29 18:09:30
50%              2017-02-23 18:05:00            2017-02-23 18:13:00
75%              2017-03-09 18:24:00            2017-03-09 18:30:15
max              2017-03-31 23:20:00            2017-03-31 23:30:00
std                              NaN                            NaN

       Start Station ID  End Station ID       Bike ID    Birth Year  \
count      16844.000000    16844.000000  16844.000000  16844.000000
mean        3215.886072     3211.575576  25292.898599   1979.304263
min         3183.000000      152.000000  15084.000000   1931.000000
25%         3186.000000     3186.000000  24523.000000   1974.000000
50%         3203.000000     3202.000000  24678.000000   1982.000000
75%         3267.000000     3217.750000  26219.000000   1986.000000
max         3281.000000     3442.000000  29296.000000   1999.000000
std           34.593994       80.103895    971.139271     10.051886

                Age  Trip Duration  Trip_Duration_in_min         Month  \
count  16844.000000   16844.000000          16844.000000  16844.000000
mean      41.695737     574.706780              9.578307      2.151686
min       22.000000      61.000000              1.000000      1.000000
25%       35.000000     220.000000              4.000000      1.000000
50%       39.000000     312.000000              5.000000      2.000000
75%       47.000000     515.000000              9.000000      3.000000
max       90.000000  390893.000000           6515.000000      3.000000
std       10.051886    4306.726768             71.778274      0.822899

        Temperature
count  16844.000000
mean      14.754096
min        9.000000
25%       13.000000
50%       15.000000
75%       16.000000
max       19.000000
std        2.388317
       Start Station Name End Station Name   User Type Age Groups  Season  \
count               16844            16844       16844      16844   16844
unique                 50               56           2          7       2
top          Grove St PATH    Grove St PATH  Subscriber      35-44  Winter
freq                 2115             2743       16526       7698    9670

          Weekday
count       16844
unique          7
top     Wednesday
freq         3301
```

```
# Count occurrences of each station as a Start and End location
top_pickup = NY_data["Start Station Name"].value_counts().nlargest(10)
```

```
top_dropoff = NY_data["End Station Name"].value_counts().nlargest(10)

plt.figure(figsize=(14, 6))

# Plot Pickup locations
plt.subplot(1, 2, 1)
sns.barplot(x=top_pickup.values, y=top_pickup.index, palette="Blues_r")
plt.xlabel("Number of Pickups")
plt.ylabel("Start Station Name")
plt.title("Top 10 Pick-up Locations")

# Plot Drop-off locations
plt.subplot(1, 2, 2)
sns.barplot(x=top_dropoff.values, y=top_dropoff.index, palette="Greens_r")
plt.xlabel("Number of Drop-offs")
plt.ylabel("End Station Name")
plt.title("Top 10 Drop-off Locations")

plt.tight_layout()
plt.show()
```

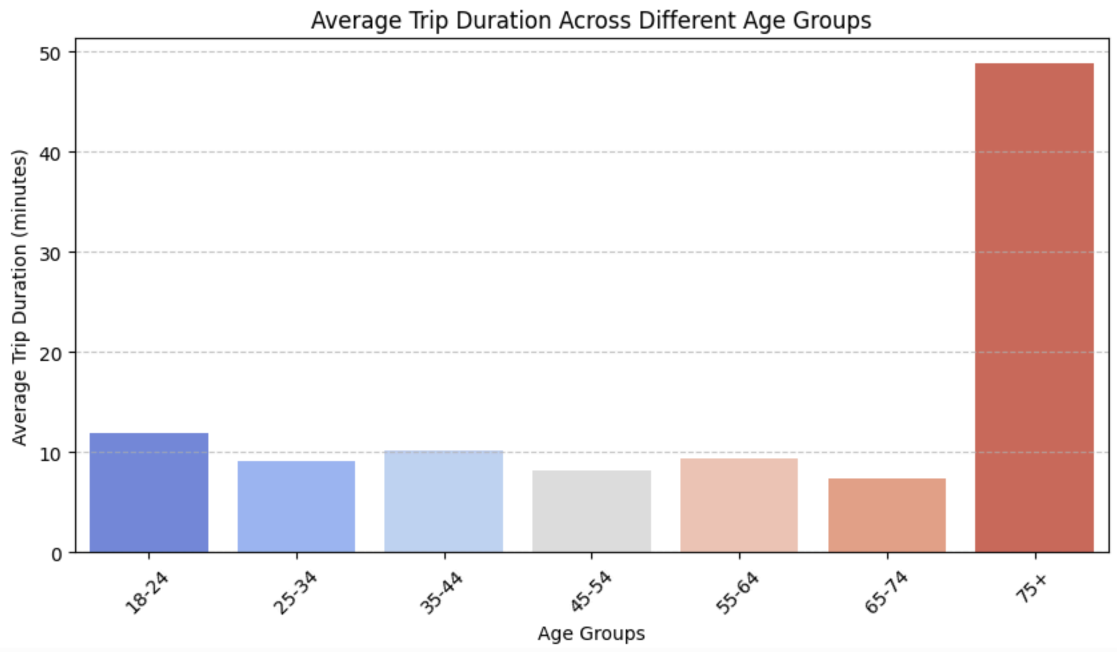
```
<ipython-input-16-a98d369d13d7>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(x=top_pickup.values, y=top_pickup.index, palette="Blues_r")
<ipython-input-16-a98d369d13d7>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(x=top_dropoff.values, y=top_dropoff.index, palette="Greens_r")
```



```
# Convert 'Trip_Duration_in_min' to numeric
NY_data['Trip_Duration_in_min'] = pd.to_numeric(NY_data['Trip_Duration_in_min'], errors='coerce')

# Group by Age Groups and calculate the mean trip duration
age_group_duration = NY_data.groupby('Age Groups')['Trip_Duration_in_min'].mean().reset_index()

# Plot the bar chart
plt.figure(figsize=(10, 5))
sns.barplot(data=age_group_duration, x='Age Groups', y='Trip_Duration_in_min', palette='coolwarm')

# Customizations
plt.xlabel('Age Groups')
plt.ylabel('Average Trip Duration (minutes)')
plt.title('Average Trip Duration Across Different Age Groups')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

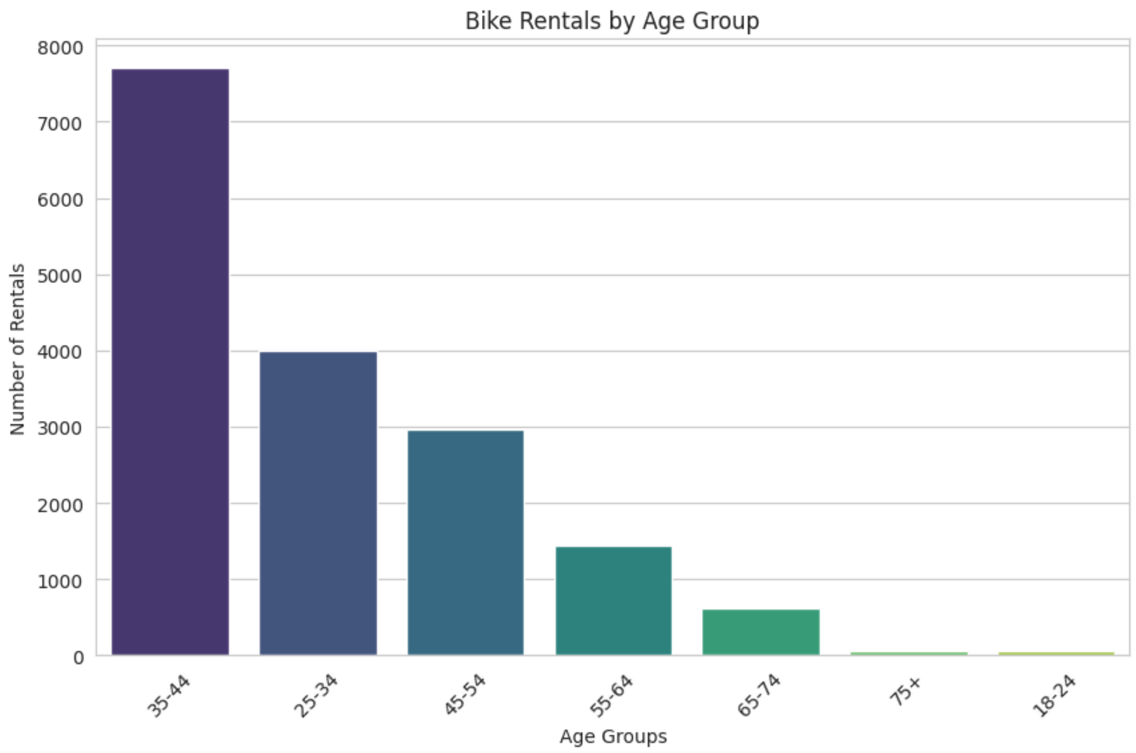
```
# Show the plot
plt.show()
```

```
<ipython-input-17-1a3a80344a3a>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(data=age_group_duration, x='Age Groups', y='Trip_Duration_in_min', palette='coolwarm')
```



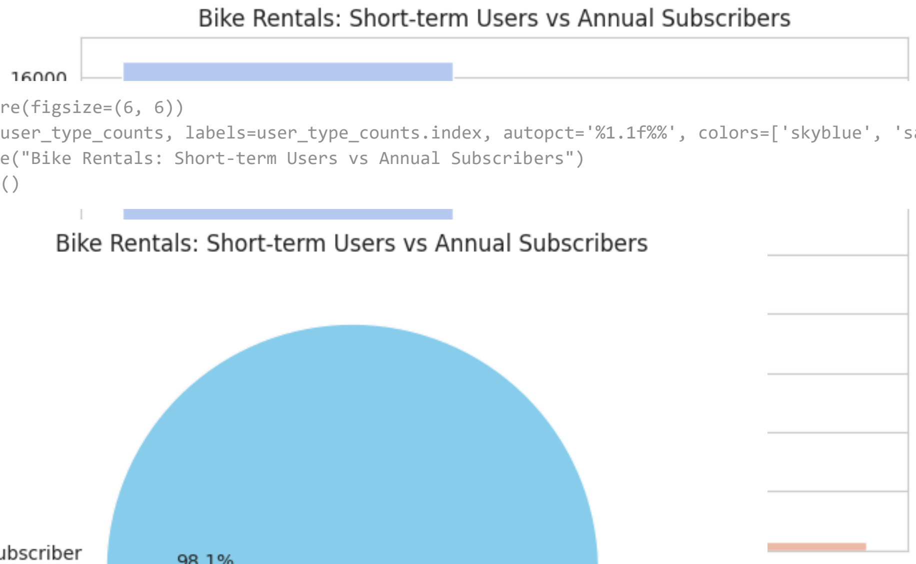
```
# Count bike rentals by age group
age_group_counts = NY_data["Age Groups"].value_counts().sort_values(ascending=False)

# Plot the data
plt.figure(figsize=(10, 6))
sns.barplot(x=age_group_counts.index, y=age_group_counts.values, palette="viridis")

# Customize the plot
plt.xlabel("Age Groups")
plt.ylabel("Number of Rentals")
plt.title("Bike Rentals by Age Group")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

# Show the plot
plt.show()
```

```
<ipython-input-19-e30710246738>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x=age_group_counts.index, y=age_group_counts.values, palette="viridis")
```

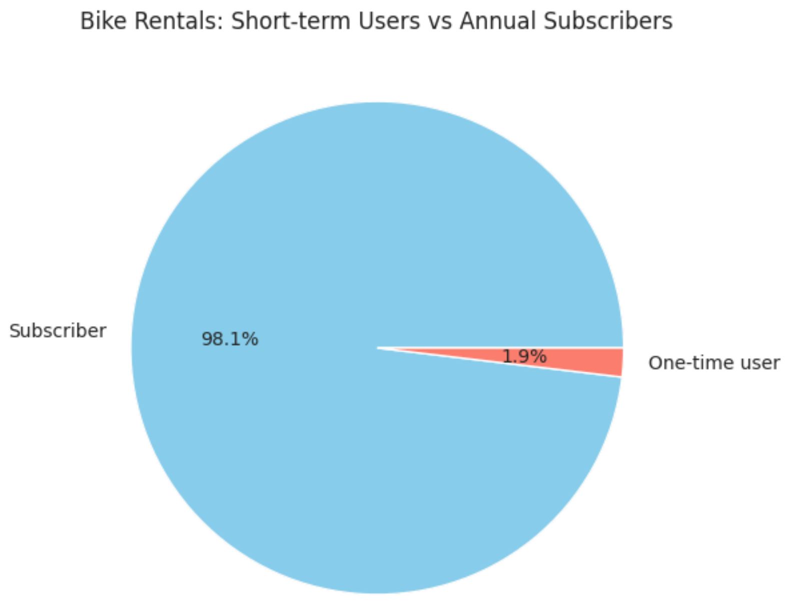


```
# Count bike rentals by user type
user_type_counts = NY_data['User Type'].value_counts()

# Plot the bar chart
plt.figure(figsize=(8, 5))
sns.barplot(x=user_type_counts.index, y=user_type_counts.values, palette='coolwarm')

# Add labels
plt.xlabel("User Type")
plt.ylabel("Number of Rentals")
plt.title("Bike Rentals: Short-term Users vs Annual Subscribers")
plt.xticks(rotation=0)

# Show the plot
plt.show()
```

```
<ipython-input-20-22211db7b0f8>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x=user_type_counts.index, y=user_type_counts.values, palette='coolwarm')
```



```
plt.figure(figsize=(6, 6))
plt.pie(user_type_counts, labels=user_type_counts.index, autopct='%1.1f%%', colors=['skyblue', 'salmon'])
plt.title("Bike Rentals: Short-term Users vs Annual Subscribers")
plt.show()
```



Start coding or generate with AI.