Micro-module 4: Sustainability and Mobility

In this session, we're going to explore the theme of mobility. We explore concepts related to sustainability, sustainable development goals, definition of urban mobility, its several modes, and we will then provide some research examples focused on the bikeshare mode.

In the tutorial part, we will use some example bikeshare data to conduct some basic data analysis and mapping exercises. Through this module, we hope to expose you to the theme of sustainable mobility and see how the data analysis could help us to better comprehend people's cycling patterns statistically and spatially.

1. Bikeshare Ridership Data

- **Data Source**

Many major North American cities provide open data portal that permits open access to bikeshare ridership data, these cities include New York City, Washington DC, Chicago, etc. In North American cities, the prevailing bikeshare system is docked bikeshare system.

In our analysis, we will use Toronto's bikeshare data, and we choose the time window of Aug.1st to Aug.7th , 2022, which spans the entire week in August.

Bikeshare Ridership data

- Open data portal

– bikeshare data of major cities in North America
New York City, Washington DC, Chicago, San Francisco, Toronto, Montreal, etc.
Note: these systems are typically docked bikeshare share system.

- We will use Toronto's Bikeshare data as an example for our analysis

– the first week (8.1 to 8.7) in August 2022 (already processed)
8.1 – 8.5 Weekdays
8.6, 8.7 Weekend

- **Structure of the Tutorial**

We are going to explore following several aspects through some exploratory data analysis. The data provided include columns such as Trip duration, Station Station ID, Start time, and so on.

Bikeshare Ridership data

- We will explore following topics:

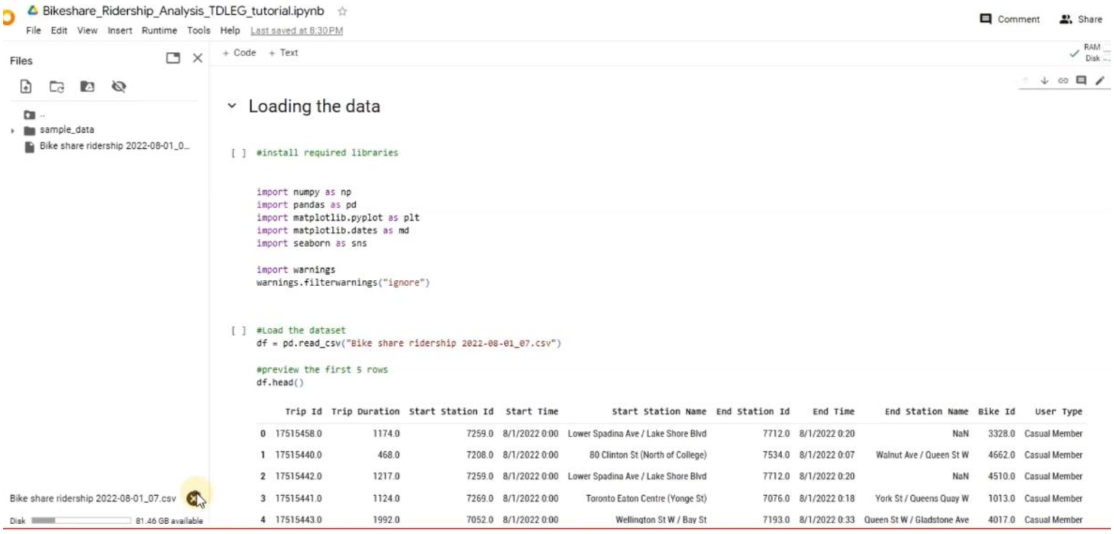
- Total biking duration difference
- Weekend vs weekday difference
- Daily difference in the entire first week in August 2022
- Temporal difference
- Membership difference

A	B	C	D	E	F	G	H	I	J	K	L	M
Trip Id	Trip Duration	Start Station Id	Start Time	Start Station Na	End Station Id	End Time	End Station Name	Bike Id	User Type			
17515458	1174	7259	8/1/2022 0:00	Lower Spadina	7712	##########	NULL	3328	Casual Member			
17515440	468	7208	8/1/2022 0:00	80 Clinton St (N	7534	##########	Walnut Ave / Quee	4662	Casual Member			
17515442	1217	7259	8/1/2022 0:00	Lower Spadina	7712	##########	NULL	4510	Casual Member			
17515441	1124	7269	8/1/2022 0:00	Toronto Eaton	7076	##########	York St / Queens Q	1013	Casual Member			
17515443	1992	7052	8/1/2022 0:00	Wellington St W	7193	##########	Queen St W / Glad	4017	Casual Member			
17515444	2642	7430	8/1/2022 0:00	Marilyn Bell Par	7220	##########	Lake Shore Blvd W	3177	Casual Member			
17515445	451	7208	8/1/2022 0:00	80 Clinton St (N	7534	##########	Walnut Ave / Quee	3550	Casual Member			
17515447	71	7269	8/1/2022 0:00	Toronto Eaton	7269	##########	Toronto Eaton Cen	3892	Casual Member			
17515448	2472	7430	8/1/2022 0:00	Marilyn Bell Par	7220	##########	Lake Shore Blvd W	1805	Casual Member			
17515449	2463	7076	8/1/2022 0:00	York St / Queen	7018	##########	Bremner Blvd / Ree	3520	Casual Member			

2. Loading the Data

- **Load the csv file and notebook**

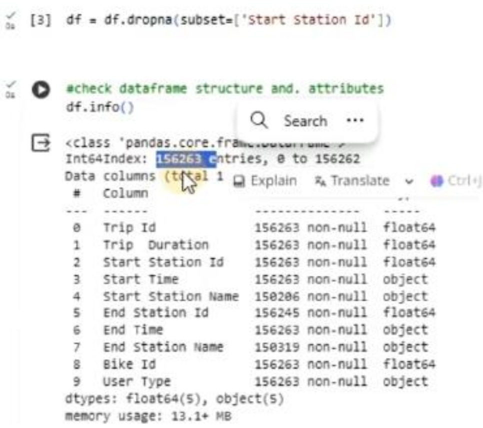
  Through Google colab, we will open the notebook and upload the csv file for analysis.



Then read the csv file in the python notebook, we are able to check the loaded data.

- **Data Preprocessing**

  Delete rows if the 'Start Station ID' contains any missing values.



Also, we will convert the Trip duration from second unit into minute unit.

```
#Convert the Trip Duration column from seconds to minutes
df['Trip  Duration'] = df['Trip  Duration']/ 60

#Rename the Trip Duration column to specify the unit of measurement
df = df.rename(columns={"Trip  Duration":"Trip Duration (Mins)"})
```

```
#preview the modified first 5 rows
df.head()
```

	Trip Id	Trip Duration (Mins)	Start Station Id	Start Time	Start Station Name	End Station Id	End Time	End Station Name	Bike Id	User Type
0	17515458.0	19.566667	7259.0	8/1/2022 0:00	Lower Spadina Ave / Lake Shore Blvd	7712.0	8/1/2022 0:20	NaN	3328.0	Casual Member

Also, we convert the time into timeobject in python.

```
df['Start Time'] = pd.to_datetime(df['Start Time'], format='%m/%d/%Y %H:%M', errors='coerce')

# Print the DataFrame to check the results
df
```

	Trip Id	Trip Duration (Mins)	Start Station Id	Start Time	Start Station Name	End Station Id	End Time	End Station Name	Bike Id	User Type
0	17515458.0	19.566667	7259.0	2022-08-01 00:00:00	Lower Spadina Ave / Lake Shore Blvd	7712.0	8/1/2022 0:20	NaN	3328.0	Casual Member
1	17515440.0	7.800000	7208.0	2022-08-01 00:00:00	80 Clinton St (North of College)	7534.0	8/1/2022 0:07	Walnut Ave / Queen St W	4662.0	Casual Member
2	17515442.0	20.283333	7259.0	2022-08-01 00:00:00	Lower Spadina Ave / Lake Shore Blvd	7712.0	8/1/2022 0:20	NaN	4510.0	Casual Member
3	17515441.0	18.733333	7269.0	2022-08-01 00:00:00	Toronto Eaton Centre (Yonge St)	7076.0	8/1/2022 0:18	York St / Queens Quay W	1013.0	Casual Member
4	17515443.0	33.200000	7052.0	2022-08-01 00:00:00	Wellington St W / Bay St	7193.0	8/1/2022 0:33	Queen St W / Gladstone Ave	4017.0	Casual Member

Then we want to delete those abnormal trips (eliminating those duration <1min or >45min). The choice of these thresholds is based on previous experience and domain knowledge.

```
# delete those random trips (<1min ride, >45min ride)

df = df[(df['Trip Duration (Mins)'] >= 1.0) & (df['Trip Duration (Mins)'] <= 45.0)]

df
```

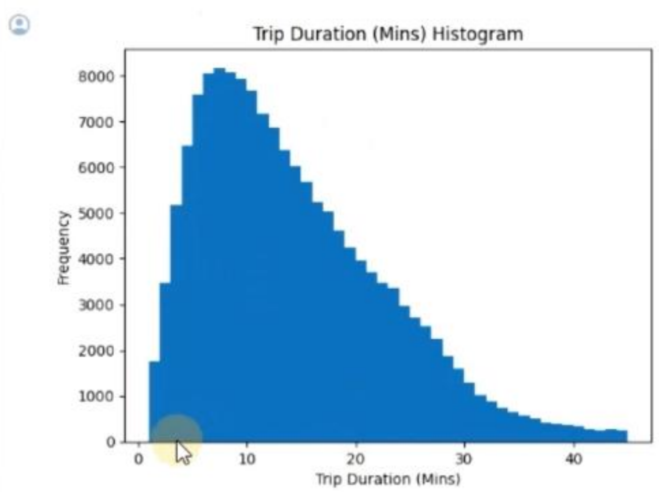
	Trip Id	Trip Duration (Mins)	Start Station Id	Start Time	Start Station Name	End Station Id	End Time	End Station Name	Bike Id	User Type
0	17515458.0	19.6	7259.0	2022-08-01 00:00:00	Lower Spadina Ave / Lake Shore Blvd	7712.0	2022-08-01 00:20:00	NaN	3328.0	Casual Member
1	17515440.0	7.8	7208.0	2022-08-01 00:00:00	80 Clinton St (North of College)	7534.0	2022-08-01 00:07:00	Walnut Ave / Queen St W	4662.0	Casual Member
2	17515442.0	20.3	7259.0	2022-08-01 00:00:00	Lower Spadina Ave / Lake Shore Blvd	7712.0	2022-08-01 00:20:00	NaN	4510.0	Casual Member
3	17515441.0	18.7	7269.0	2022-08-01 00:00:00	Toronto Eaton Centre (Yonge St)	7076.0	2022-08-01 00:18:00	York St / Queens Quay W	1013.0	Casual Member
4	17515443.0	33.2	7052.0	2022-08-01 00:00:00	Wellington St W / Bay St	7193.0	2022-08-01 00:33:00	Queen St W / Gladstone Ave	4017.0	Casual Member
...	...	...	...	...	...	...	...	...	...	...
156258	17694272.0	13.9	7195.0	2022-08-07 [illegible]	Ulster St / Bathurst St	7458.0	2022-08-08 [illegible]	Church St / Lombard St	4802.0	Annual [illegible]

3. Analysing the Trip Duration

- Plot a histogram

We will use the Trip duration column to plot a histogram of our data.

```
#Create a historgam

plt.hist(df['Trip Duration (Mins)'], bins=44)  # 'bins' defines the number of intervals

# Add titles and labels
plt.title('Trip Duration (Mins) Histogram')
plt.xlabel('Trip Duration (Mins)')
plt.ylabel('Frequency')


plt.show()
```



many rides are between 5 to 20 mins

We will also numerically describe the data by comparing its mean and median.

```
df['Trip Duration (Mins)'].describe()

count    152243.000000
mean         14.314516
std           8.549454
min           1.000000
25%           7.600000
50%          12.600000
75%          19.600000
max          45.000000
Name: Trip Duration (Mins), dtype: float64
```

4. Analysing the Trip difference between weekdays and weekends

- **Assign labels to data to differentiate weekdays and weekends**

  We will create a new column called 'Day Type' to determine whether the date belongs to weekdays or weekends.

```
import datetime

# Define the date range
start_date = datetime.date(2022, 8, 1)
end_date = datetime.date(2022, 8, 7)

# Iterate through the date range
for single_date in (start_date + datetime.timedelta(days=n) for n in range((end_date - start_date).days + 1)):
    day_of_week = single_date.weekday()
    day_type = "Weekday" if day_of_week < 5 else "Weekend"
    print(f"Date: {single_date}, Day of Week: {day_of_week} ({day_type})")

Date: 2022-08-01, Day of Week: 0 (Weekday)
Date: 2022-08-02, Day of Week: 1 (Weekday)
Date: 2022-08-03, Day of Week: 2 (Weekday)
Date: 2022-08-04, Day of Week: 3 (Weekday)
Date: 2022-08-05, Day of Week: 4 (Weekday)
Date: 2022-08-06, Day of Week: 5 (Weekend)
Date: 2022-08-07, Day of Week: 6 (Weekend)
```

```
# Classify each row as 'Weekday' or 'Weekend'
df['Day Type'] = df['Start Time'].apply(lambda x: 'Weekday' if x.weekday() < 5 else 'Weekend')
```

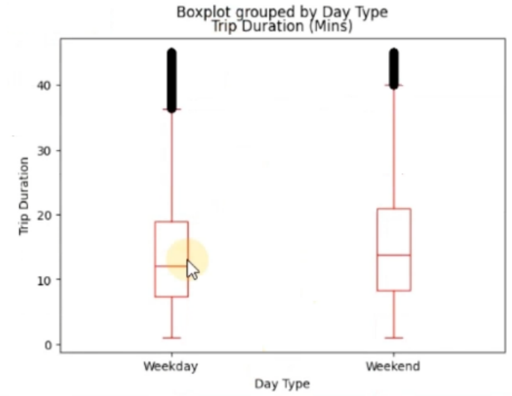
Run cell (Ctrl+Enter)
cell has not been executed in this session

```
[ ] df
```

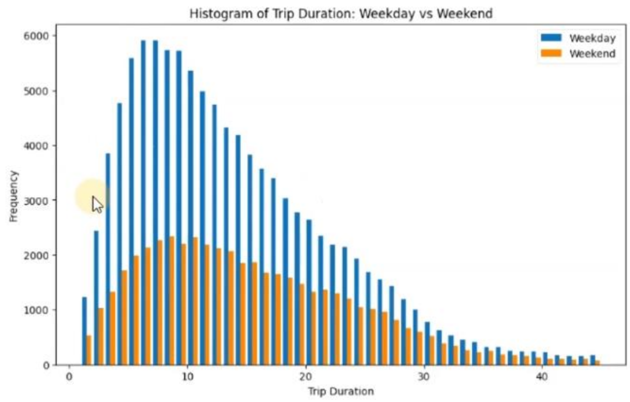
We use the box plot to compare the difference in trip duration

```
# Create the box plot
plt.figure(figsize=(10, 6))
df.boxplot(by='Day Type', column='Trip Duration (Mins)',grid=False, color = 'red')

# Set titles and labels

plt.xlabel('Day Type')
plt.ylabel('Trip Duration')

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

```
<Figure size 1000x600 with 0 Axes>
```



We further use the histogram to provide more information, it demonstrates that in weekends, people seem to travel longer.



If we compare the histogram, it gives similar idea, because the longer trips accounts for slightly more share in the weekend trips.

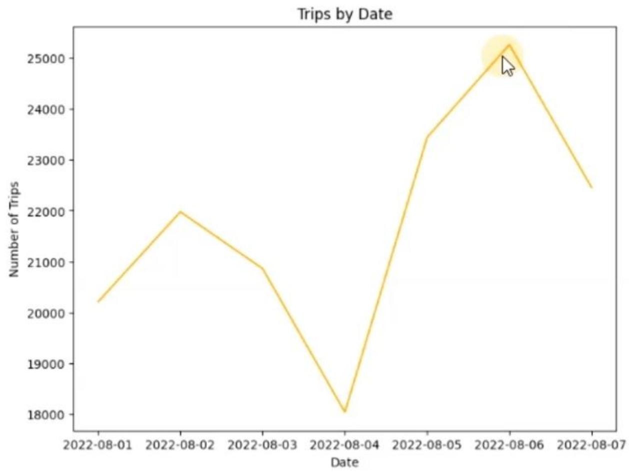
5. Understanding the trip pattern in the week

- **Group by the data and then plot the map**

  We will first group the data by the date itself.

```
# Extract just the date part from 'Start Time' presented in our dataframe df.
df['Date'] = df['Start Time'].dt.date

# Groupby the new 'Date' column and count the number of trips
trip_counts = df.groupby('Date').size()

df
```

We can further obtain a line graph; it shows that Saturday has the highest rides.



6. Disclosing the temporality of the data by hour of day

- **Group by the data and then plot the map**

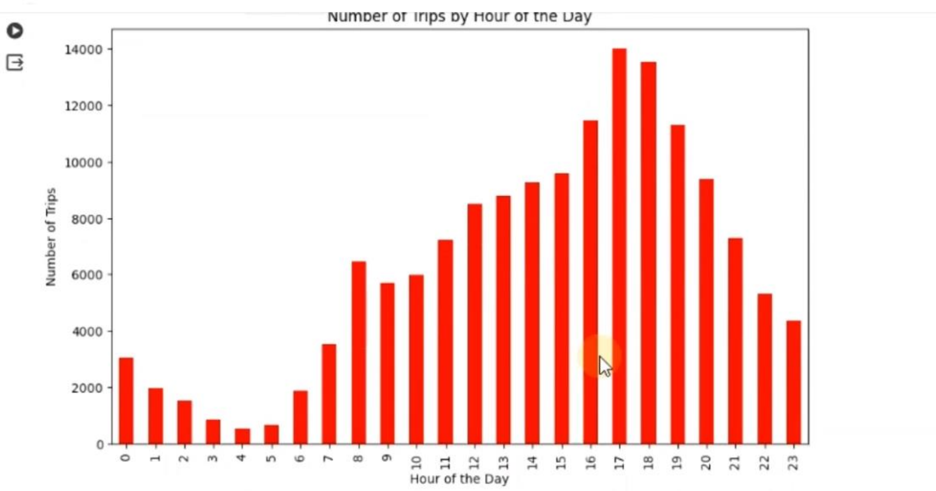
  We will first group the data by the hour.

```
# Extract just the hour part from 'Start Time'
df['Hour'] = df['Start Time'].dt.hour

# Group by the new 'Hour' column and count the number of trips
trip_counts_by_hour = df.groupby('Hour').size()
```

```
[ ] df
```

Then we plot the figure

```
# Create the plot
plt.figure(figsize=(10, 6))
trip_counts_by_hour.plot(kind='bar', color='skyblue')

# Set titles and labels
plt.title('Number of Trips by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Trips')

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



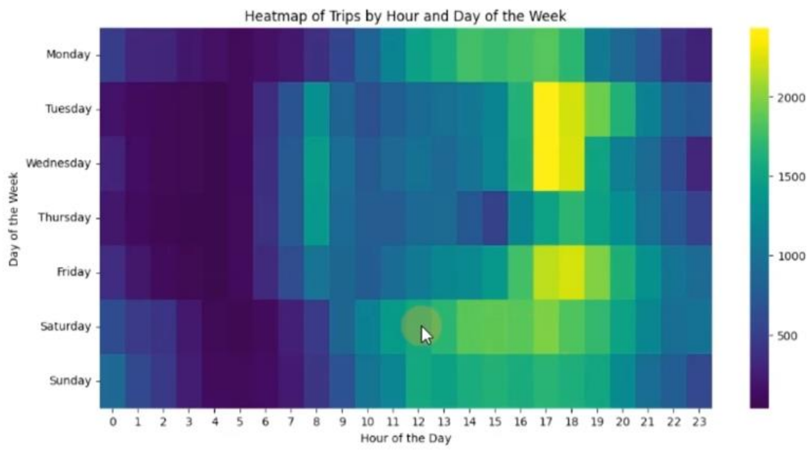
The busiest hour is 5pm, and this trend aligns with the rush hour. This suggests lots of rides are from work to home.

-

- **Creating a heatmap by hour and day**

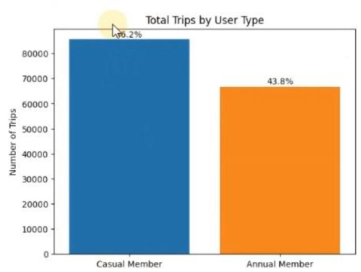
  We will first create a pivot table and then fill the value using corresponding count in these cells.

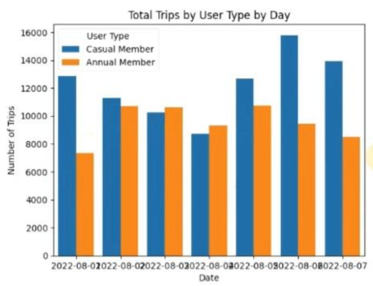
```
df['DayOfWeek'] = df['Start Time'].dt.day_name()

# Create a pivot table
pivot_table = df.pivot_table(index='DayOfWeek', columns='Hour', aggfunc='size', fill_value=0)

# Reorder the days of the week
reorderlist = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
pivot_table = pivot_table.reindex(reorderlist)

# Create the heatmap
plt.figure(figsize=(12, 6))
sns.heatmap(pivot_table, cmap='viridis', annot=False)

# Set titles and labels
plt.title('Heatmap of Trips by Hour and Day of the Week')
plt.xlabel('Hour of the Day')
plt.ylabel('Day of the Week')

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



7. Comparing the trips by different user types

- **Compare the total trips**





there are more rides by casual member during weekends.

8. Spatial Analysis – Data Preparation

- **Aggregating data to the station level**

We need to groupby the data based on unique Start Station ID.

```
[31] # Group the trips by bikeshare station ID and count the number of trips for each station
     Departuretrips = df2.groupby('Start Station Id')['Start Station Id'].count()

[32] Departuretrips

     Start Station Id
     7000.0    783
     7001.0    471
     7002.0    539
     7003.0    296
     7004.0    202
              ...
     7708.0     67
     7709.0     65
     7710.0    155
     7711.0     71
     7712.0    248
     Name: Start Station Id, Length: 625, dtype: int64
```

- **Merging the trip data to another dataset contains the geo information of stations**

We will load another dataframe of the station information

```
# we will load another csv file which contains stations geolocations.
# the Station ID is the unique value that we can rely on to join our trip data to the station.

#Load the dataset
df_station = pd.read_csv("Bikeshare Station_cleaned.csv", encoding='latin1')

#preview the first 5 rows
df_station.head()
```

	station_id	name	lat	lon
0	7000	Fort York Blvd / Capreol Ct	43.639832	-79.395954
1	7001	Wellesley Station Green P	43.664964	-79.383550
2	7002	St. George St / Bloor St W	43.667333	-79.399429
3	7003	Madison Ave / Bloor St W	43.667158	-79.402761
4	7004	University Ave / Elm St	43.656518	-79.389099

After renaming the column name, we will match the 2 dataframes based on the same column 'Station ID'.

```
# Merge the station_info DataFrame
station_trips = pd.merge( Departuretrips_df,df_station, on='Station ID', how = 'inner')

# DataFrame
station_trips
```

	Station ID	DepartureTrips	name	lat	lon
0	7000.0	783	Fort York Blvd / Capreol Ct	43.639832	-79.395954
1	7001.0	471	Wellesley Station Green P	43.664964	-79.383550
2	7002.0	539	St. George St / Bloor St W	43.667333	-79.399429
3	7003.0	296	Madison Ave / Bloor St W	43.667158	-79.402761
4	7004.0	202	University Ave / Elm St	43.656518	-79.389099
...	...	...	...	...	...
599	7708.0	67	101 Cedarvale Ave	43.686868	-79.311094
600	7709.0	65	Beltline Trail / Yonge St	43.696230	-79.395043
601	7710.0	155	11 Spadina Rd	43.667725	-79.404137
602	7711.0	71	Havelock St / Dewson St	43.655479	-79.430246
603	7712.0	248	Queen St W / Shaw St	43.644246	-79.416104

-

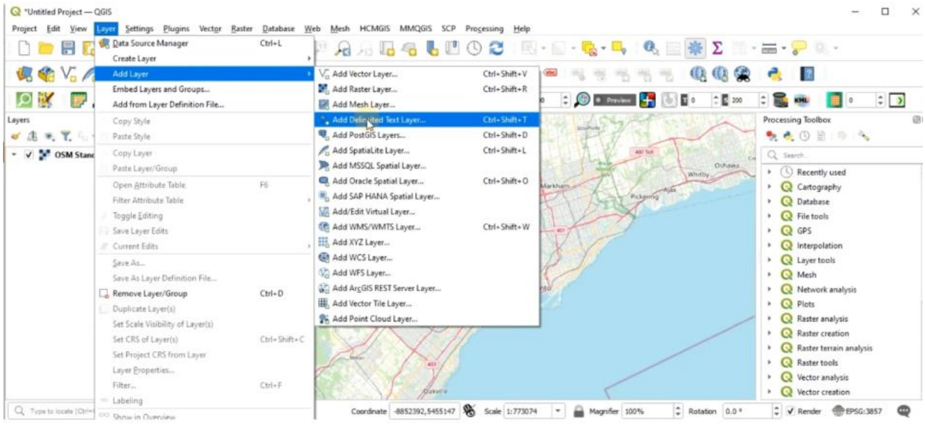
- **Export the csv and move into QGIS**

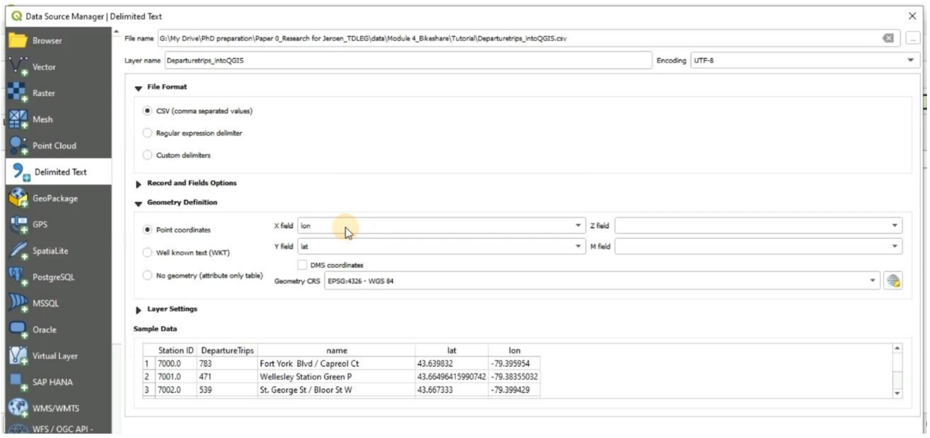
```
# export the dataframe to csv file so we can open the data and visualise in QGIS
station_trips.to_csv('Departuretrips_intoQGIS.csv', index = False)
```

9. Spatial Analysis - Mapping

- **Add points by delimited text file csv.**

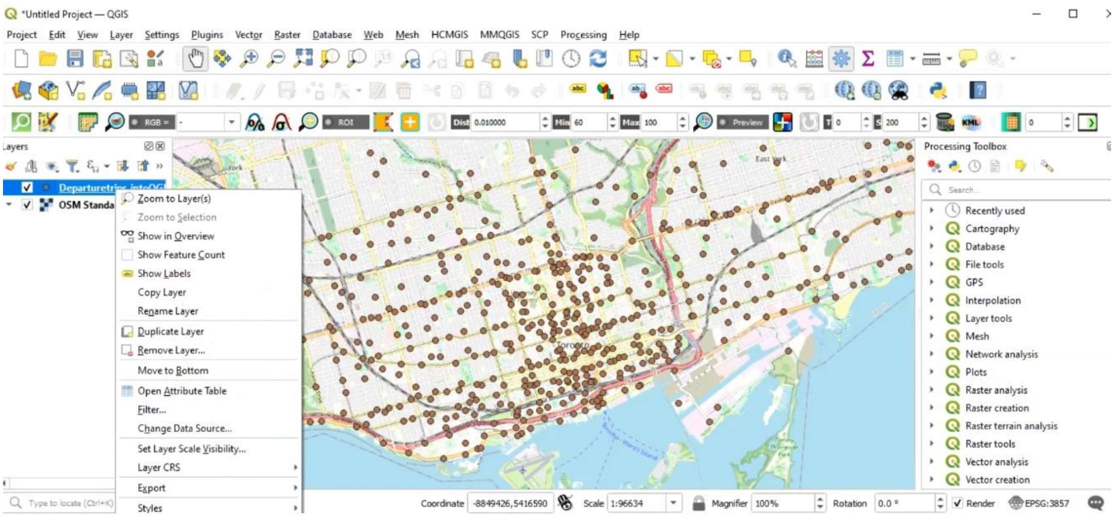
  Add a layer - add delimited text and load the data by choosing the point coordinate.





- **Visualise the trip pattern based on the trip volume**

  Now we have data loaded indicating the corresponding locations

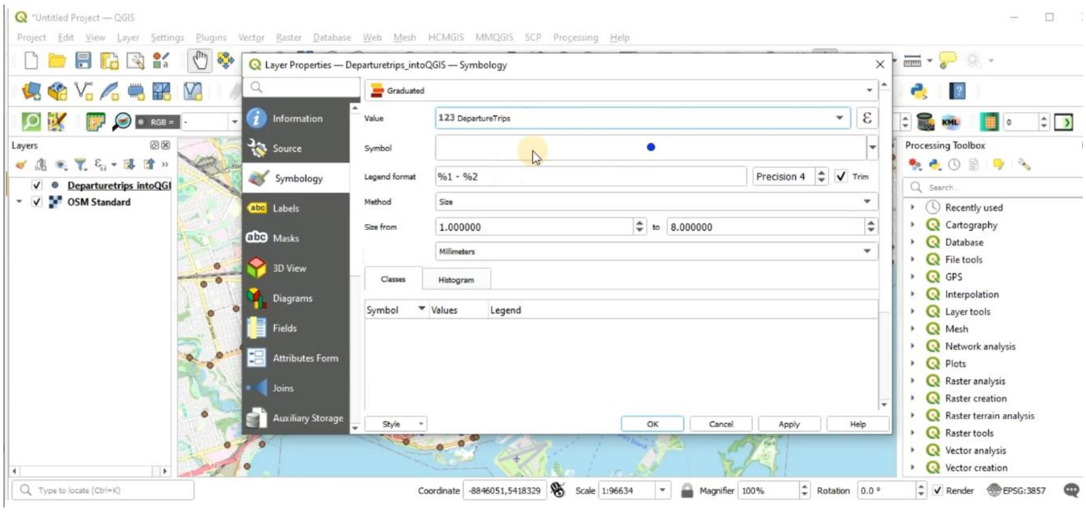


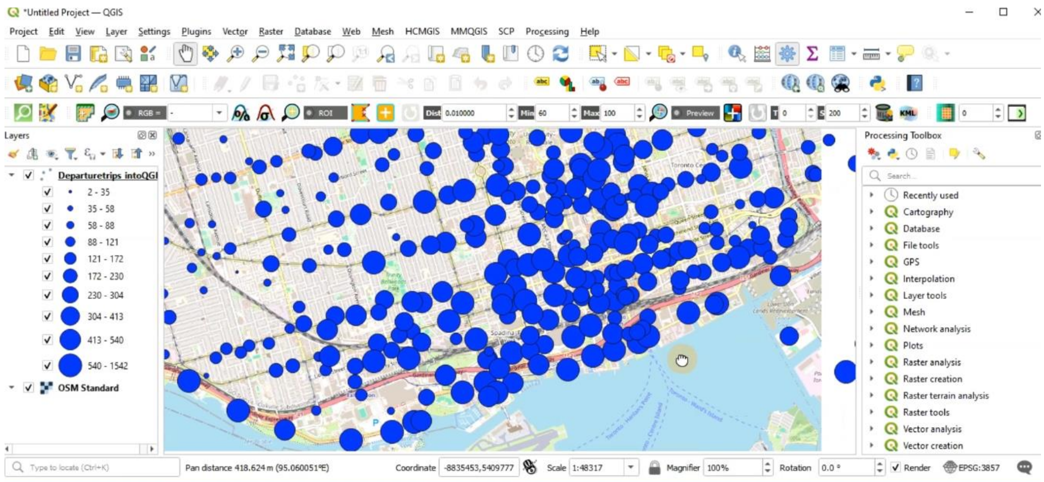
We will need to visualise the data based on the trip volume, which can determine the size of the dots.

We can choose gradual symbology

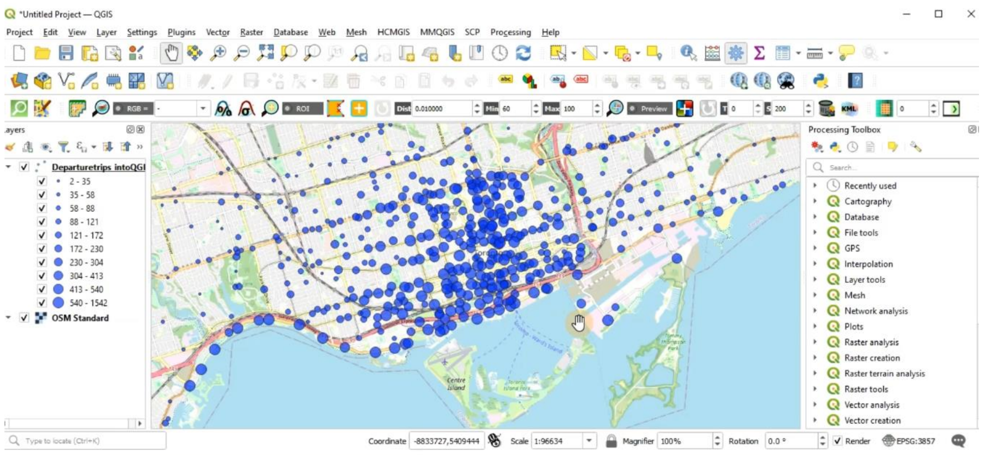
Value is based on the column ‘Departure Trips’

Method: size



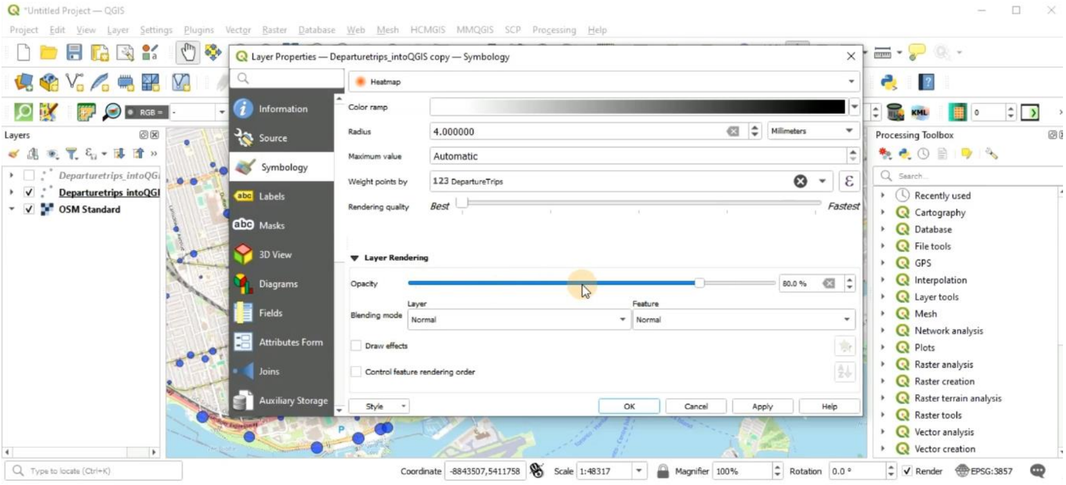


Adjust the opacity of the symbology.

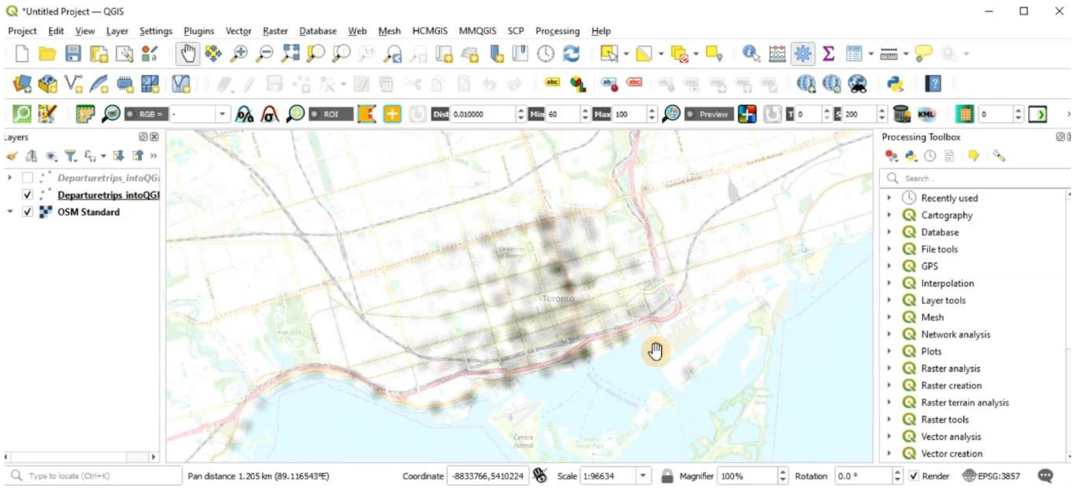


10. Spatial Analysis – Creating a Heatmap

- **In the symbology panel, defining a reasonable radius, and weight points by 'DepartureTrips'.**
- **Adjust the opacity**



-



-

11. Final Notes

- Bikeshare data from different providers may contain different information.

– For instance, New York trip data contains the geographical information of stations directly.

ride_id	rideable_t	started_at	ended_at	start_stati	start_stati	end_statio	end_statio	start_lat	start_lng	end_lat	end_lng	member_casual
E2E964A1(	classic_bik	########	########	6 St & Grai	HB302	Madison S	HB503	40.744397	-74.0345	40.749943	-74.0359	member
0660F2E48	classic_bik	########	########	6 St & Grai	HB302	6 St & Grai	HB302	40.744397	-74.0345	40.744397	-74.0345	member
940FC7C6	classic_bik	########	########	Heights Ele	JC059	Heights Ele	JC059	40.74872	-74.0405	40.748715	-74.0404	member

- Dockless Bikeshare Data in China

– may not be open access (Shenzhen provides bikeshare data in 2021, some other bikeshare companies provide maybe 1-2 weeks data); might need to get the data directly from bikeshare companies.
– be careful about its unique projection system, needs an additional conversion step and decode the geohash information.
– Dockless bikeshare trip patterns could consider clustering algorithms to understand its heat spots.