Loading required python packages

In [1]:
```
import pandas as pd
import geopy.distance  #distance of coordinates
import numpy as np
import glob            #for specific pattern recognition
import matplotlib.pyplot as plt
```

Collecting all files

Here we are working with data from a cyclistic 12 months dataset. all are in same type (.csv) files.

In [2]:
```
all_files=glob.glob(r'C:\Users\mahad\Downloads\capstone_project\project_1\*.
```

In [3]:
```
all_files
```

Out[3]:
```
['C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202004-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202005-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202006-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202007-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202008-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202009-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202010-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202011-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202012-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202101-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202102-divvy-tripda
ta.csv',
 'C:\\Users\\mahad\\Downloads\\capstone_project\\project_1\\202103-divvy-tripda
ta.csv']
```

Concatenating all files into one

In [4]:
```
yearly= pd.concat((pd.read_csv(file)
                   for file in all_files),ignore_index=True)
```

inspecting files for inconsistancy, null value and data typtes.

In [5]:
```
1 yearly.describe()
```

Out[5]:

	start_lat	start_lng	end_lat	end_lng
count	3.489748e+06	3.489748e+06	3.485010e+06	3.485010e+06
mean	4.190417e+01	-8.764494e+01	4.190444e+01	-8.764522e+01
std	4.364222e-02	2.575969e-02	4.373705e-02	2.589123e-02
min	4.164000e+01	-8.787000e+01	4.154000e+01	-8.807000e+01
25%	4.188224e+01	-8.765888e+01	4.188266e+01	-8.765917e+01
50%	4.190000e+01	-8.764170e+01	4.190068e+01	-8.764275e+01
75%	4.193000e+01	-8.762773e+01	4.193120e+01	-8.762775e+01
max	4.208000e+01	-8.752000e+01	4.216000e+01	-8.744000e+01

In [6]:
```
1 yearly.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3489748 entries, 0 to 3489747
Data columns (total 13 columns):
 #   Column              Non-Null Count    Dtype  
---  ------              --------------    -----  
 0   ride_id             3489748 non-null  object 
 1   rideable_type       3489748 non-null  object 
 2   started_at          3489748 non-null  object 
 3   ended_at            3489748 non-null  object 
 4   start_station_name  3367573 non-null  object 
 5   start_station_id    3366947 non-null  object 
 6   end_station_name    3346506 non-null  object 
 7   end_station_id      3346045 non-null  object 
 8   start_lat           3489748 non-null  float64
 9   start_lng           3489748 non-null  float64
 10  end_lat             3485010 non-null  float64
 11  end_lng             3485010 non-null  float64
 12  member_casual       3489748 non-null  object 
dtypes: float64(4), object(9)
memory usage: 346.1+ MB
```

In [7]:
```
1 yearly.columns
```

Out[7]:
```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual'],
      dtype='object')
```

convert date and time *datetime* readable

In [ ]:
```
1
```

In [8]:
```
1 yearly['started_at'] =  pd.to_datetime(yearly['started_at'], format='%Y-%m-%
2
```

In [9]:
```
1 yearly['ended_at'] =  pd.to_datetime(yearly['ended_at'], format='%Y-%m-%d %H
```

In [10]:
```
1 yearly.dtypes
```

Out[10]:
```
ride_id                     object
rideable_type               object
started_at          datetime64[ns]
ended_at            datetime64[ns]
start_station_name          object
start_station_id            object
end_station_name            object
end_station_id              object
start_lat                  float64
start_lng                  float64
end_lat                    float64
end_lng                    float64
member_casual               object
dtype: object
```

Calculate the riding time

In [11]:
```
1 yearly['riding_time'] = (yearly['ended_at'] - yearly['started_at'])/pd.Timed
```

checking the result. we find some unusual vaules so we inspect them. there are some negetive values and some illogically big vaules which caused by corrupted data we need to fillter them. we took maximum value of 24 hour.

In [12]:
```
1 print (yearly['riding_time'].max())
2 print (yearly['riding_time'].min())
```

```
58720.03333333333
-29049.966666666667
```

In [13]:
```
1 yearly.describe()
```

Out[13]:

	start_lat	start_lng	end_lat	end_lng	riding_time
count	3.489748e+06	3.489748e+06	3.485010e+06	3.485010e+06	3.489748e+06
mean	4.190417e+01	-8.764494e+01	4.190444e+01	-8.764522e+01	2.476664e+01
std	4.364222e-02	2.575969e-02	4.373705e-02	2.589123e-02	3.904216e+02
min	4.164000e+01	-8.787000e+01	4.154000e+01	-8.807000e+01	-2.904997e+04
25%	4.188224e+01	-8.765888e+01	4.188266e+01	-8.765917e+01	7.883333e+00
50%	4.190000e+01	-8.764170e+01	4.190068e+01	-8.764275e+01	1.451667e+01
75%	4.193000e+01	-8.762773e+01	4.193120e+01	-8.762775e+01	2.663333e+01
max	4.208000e+01	-8.752000e+01	4.216000e+01	-8.744000e+01	5.872003e+04

In [14]:
```
1 yearly = yearly[yearly['riding_time'].between(0,1440)]
```

In [15]:
```
1 print (yearly['riding_time'].max())
2 print (yearly['riding_time'].min())
```

```
1439.9
0.0
```

In [ ]:
```
1
```

Creating weekday for analysis

In [16]:
```
1 yearly['day_of_week'] = yearly['started_at'].dt.day_name()
```

In [17]:
```
1 yearly['day_of_week'].unique()
```

Out[17]:
```
array(['Sunday', 'Friday', 'Wednesday', 'Tuesday', 'Saturday', 'Thursday',
       'Monday'], dtype=object)
```

classifying riding hour

In [18]:
```
1 yearly['riding_hour'] = yearly['started_at'].dt.hour
```

In [19]:
```
1 yearly['riding_hour'].unique()
```

Out[19]:
```
array([17, 12, 10, 14, 15, 18, 13,  2, 16,  8, 20, 19, 11, 23,  6,  9,  7,
       22, 21,  1,  5,  0,  4,  3], dtype=int64)
```

In [20]:
```
conditions = [
    (yearly['riding_hour'].between(0,4)),
    (yearly['riding_hour'].between(4,12)),
    (yearly['riding_hour'].between(12,16)),
    (yearly['riding_hour'].between(16,24))
]

values = ['night_ride','morning_ride','afternoon_ride', 'evening_ride']

yearly['riding_hour'] = np.select(conditions, values)
```

In [21]:
```
yearly['riding_hour'].unique()
```

Out[21]:
```
array(['evening_ride', 'morning_ride', 'afternoon_ride', 'night_ride'],
      dtype=object)
```

In [22]:
```
yearly.columns
```

Out[22]:
```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual', 'riding_time', 'day_of_week', 'riding_hour'],
      dtype='object')
```

In [23]:
```
yearly
```

Out[23]:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_stat
0	A847FADBBC638E45	docked_bike	2020-04-26 17:45:14	2020-04-26 18:12:03	Eckhart Park	
1	5405B80E996FF60D	docked_bike	2020-04-17 17:08:54	2020-04-17 17:17:03	Drake Ave & Fullerton Ave	
2	5DD24A79A4E006F4	docked_bike	2020-04-01 17:54:13	2020-04-01 18:08:36	McClurg Ct & Erie St	
3	2A59BBDF5CDBA725	docked_bike	2020-04-07 12:50:19	2020-04-07 13:02:31	California Ave & Division St	
4	27AD306C119C6158	docked_bike	2020-04-18 10:22:59	2020-04-18 11:15:54	Rush St & Hubbard St	
...	...	...	...	...	...	

In [24]:
```
1 yearly.dtypes
```

Out[24]:
```
ride_id                      object
rideable_type                object
started_at           datetime64[ns]
ended_at             datetime64[ns]
start_station_name           object
start_station_id             object
end_station_name             object
end_station_id               object
start_lat                   float64
start_lng                   float64
end_lat                     float64
end_lng                     float64
member_casual                object
riding_time                 float64
day_of_week                  object
riding_hour                  object
dtype: object
```

actually we do not need user ID and Station names we can drop those columns.

In [25]:
```
1 trip_data=yearly.drop(['ride_id','start_station_name','start_station_id','en
2                        'end_station_id'], axis=1)
```

In [26]:
```
1 #yearly.to_csv('tripdata.csv',index= False)
```

In [27]:
```
1 trip_data.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3476314 entries, 0 to 3489747
Data columns (total 11 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   rideable_type  3476314 non-null  object
 1   started_at     3476314 non-null  datetime64[ns]
 2   ended_at       3476314 non-null  datetime64[ns]
 3   start_lat      3476314 non-null  float64
 4   start_lng      3476314 non-null  float64
 5   end_lat        3472286 non-null  float64
 6   end_lng        3472286 non-null  float64
 7   member_casual  3476314 non-null  object
 8   riding_time    3476314 non-null  float64
 9   day_of_week    3476314 non-null  object
 10  riding_hour    3476314 non-null  object
dtypes: datetime64[ns](2), float64(5), object(4)
memory usage: 318.3+ MB
```

the final data look like this

In [28]:
```
1 trip_data
```

Out[28]:

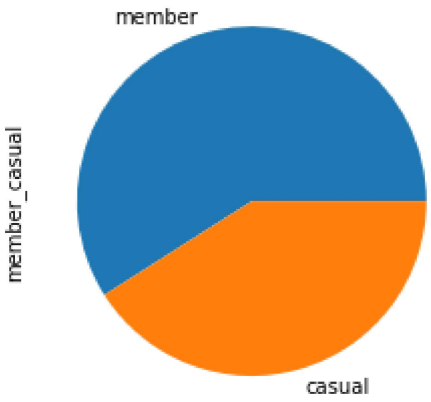
	rideable_type	started_at	ended_at	start_lat	start_lng	end_lat	end_lng	mem
0	docked_bike	2020-04-26 17:45:14	2020-04-26 18:12:03	41.896400	-87.661000	41.932200	-87.658600	
1	docked_bike	2020-04-17 17:08:54	2020-04-17 17:17:03	41.924400	-87.715400	41.930600	-87.723800	
2	docked_bike	2020-04-01 17:54:13	2020-04-01 18:08:36	41.894500	-87.617900	41.867900	-87.623000	
3	docked_bike	2020-04-07 12:50:19	2020-04-07 13:02:31	41.903000	-87.697500	41.899200	-87.672200	
4	docked_bike	2020-04-18 10:22:59	2020-04-18 11:15:54	41.890200	-87.626200	41.969500	-87.654700	

In [ ]:
```
1
```

In [29]:
```
1 print(trip_data['member_casual'].value_counts())
2 trip_data['member_casual'].value_counts().plot(kind='pie')
```

```
member    2051734
casual    1424580
Name: member_casual, dtype: int64
```

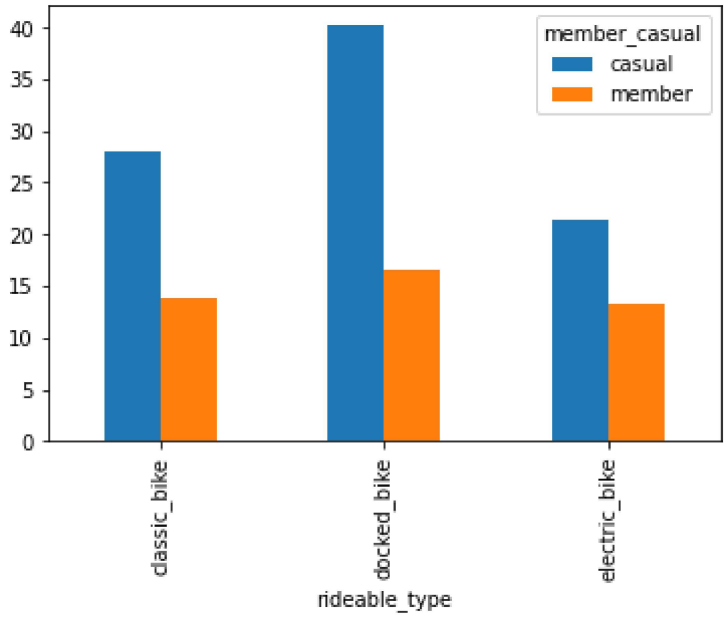
Out[29]: <AxesSubplot:ylabel='member_casual'>



In [30]:
```
print(trip_data.pivot_table(index='rideable_type',values='riding_time',colum
trip_data.pivot_table(index='rideable_type',values='riding_time',columns='me
```

```
member_casual     casual     member
rideable_type
classic_bike   27.956468  13.889381
docked_bike    40.186438  16.493655
electric_bike  21.385278  13.309021
```

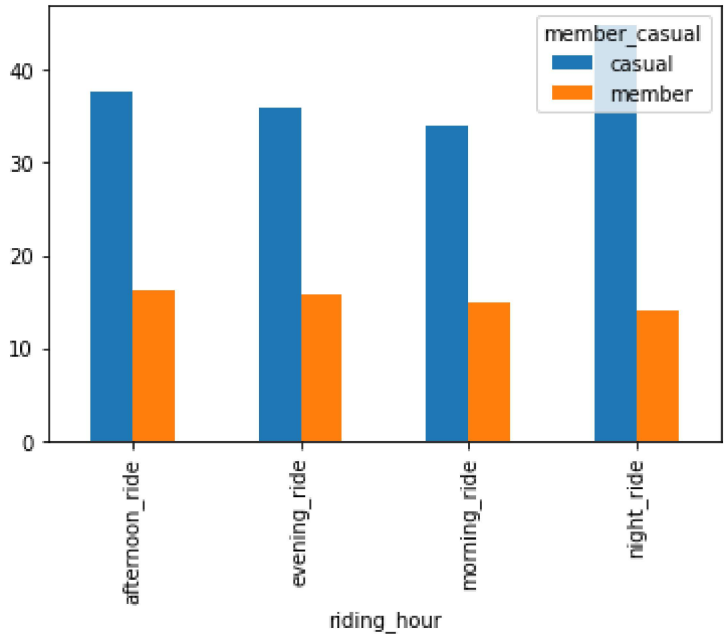
Out[30]: <AxesSubplot:xlabel='rideable_type'>



In [31]:
```
print(trip_data.pivot_table(index='riding_hour',values='riding_time',columns
trip_data.pivot_table(index='riding_hour',values='riding_time',columns='memb
```

```
member_casual      casual     member
riding_hour
afternoon_ride  37.691010  16.234165
evening_ride    35.934929  15.849243
morning_ride    34.078059  14.881799
night_ride      44.841748  14.057713
```

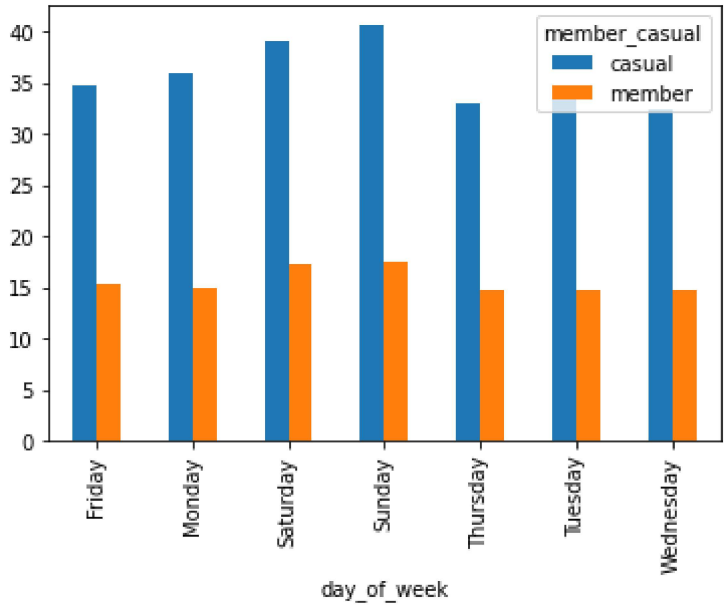
Out[31]: <AxesSubplot:xlabel='riding_hour'>



In [32]:
```
print(trip_data.pivot_table(index='day_of_week',values='riding_time',columns
trip_data.pivot_table(index='day_of_week',values='riding_time',columns='memb
```

member_casual	casual	member
day_of_week		
Friday	34.776969	15.384440
Monday	35.890543	14.830327
Saturday	39.040948	17.266623
Sunday	40.618858	17.396474
Thursday	32.991772	14.809095
Tuesday	33.549535	14.715884
Wednesday	32.411688	14.806827

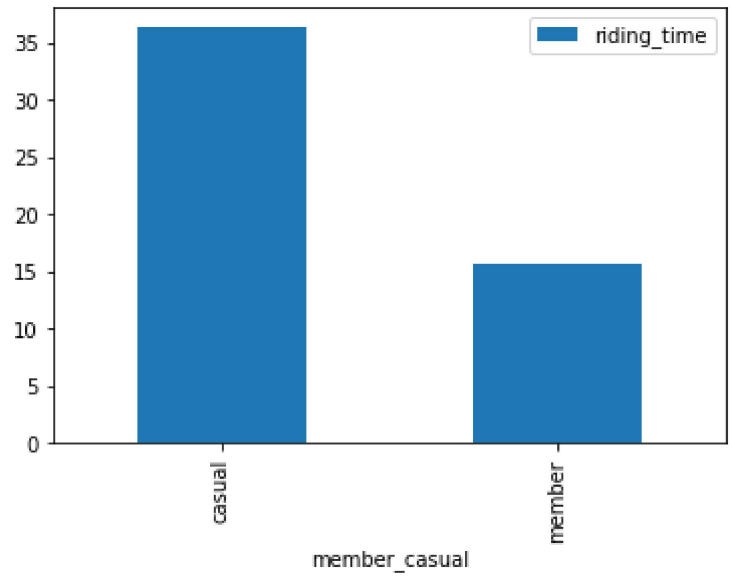
Out[32]: <AxesSubplot:xlabel='day_of_week'>



In [33]:
```
print(trip_data.pivot_table(index='member_casual',values='riding_time'))
trip_data.pivot_table(index='member_casual',values='riding_time').plot(kind=
```

```
               riding_time
member_casual
casual           36.373883
member           15.605983
```

Out[33]: <AxesSubplot:xlabel='member_casual'>



In [34]:
```
#yearly.pivot_table(index='member_casual',columns='started_at',aggfunc='coun
```

In [35]:
```
#date=trip_data.groupby(by=[trip_data[started_at].month, trip_data[started_a
#trip_data.groupby(pd.Grouper(freq='M'))
#g = trip_data.groupby('started_at')

#res = g['Values'].sum()
```

In [36]:
```
#g
```

In [37]:
```
#total_trip=yearly.groupby(pd.Grouper(key='started_at', axis=0, freq='M')).c
```

In [38]:
```
#total_trip
```

In [39]:
```
#total_trip.pivot_table(index='started_at',values='member_casual',aggfunc='c
```

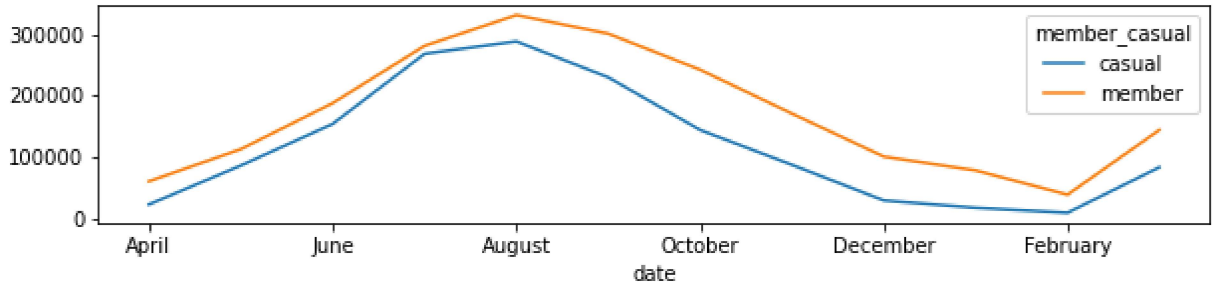
In [40]:
```
#trip_data['date']=trip_data['started_at'].dt.to_period('M')
yearly['date']=trip_data['started_at'].dt.month_name()

#sales_df['Month'] = sales_df['Date'].dt.month_name(locale='English')
```

In [46]:
```
yearly['date'].unique()
```

Out[46]:
```
array(['April', 'May', 'June', 'July', 'August', 'September', 'October',
       'November', 'December', 'January', 'February', 'March'],
      dtype=object)
```

In [61]:
```
print(yearly.pivot_table(index='date',sort=False,values= 'ride_id',columns='
yearly.pivot_table(index='date',sort=False,values= 'ride_id',columns='member
plt.rcParams["figure.figsize"] = (10, 5)
```

member_casual	casual	member
date		
April	23507	61095
May	86666	113236
June	154216	187963
July	268021	281003
August	288183	330914
September	229800	300715
October	144368	242169
November	87820	170920
December	29956	101130
January	18090	78705
February	10073	39432
March	83880	144452



In [43]:
```
#trip_data.groupby('date')['member_casual'].count()
```

In [ ]:
```

```

In [ ]:
```

```