EDA - MTA Wi-Fi Network Working Plan

October 9, 2021

1 Overview:

The Metropolitan Transportation Authority (MTA) is the entity responsible over New York state subway system, recently they require to install Wi-Fi network in the subway stations to minimize the inconveniences that might results due to the wait time before train arrival by keeping the daily commuters well connected. The MTA contracted us as a consultancy group to advise on a working plan for this project.

[1]:
```
#First we will upload the dataset in the ipython environment to start the MTA␣
 ↪data exploration
#Using pandas and sqlalchemy libraries

import pandas as pd
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import create_engine
```

[2]:
```
#Get the MTA data within the desired datetime range

def get_data(week_nums):
    url = "http://web.mta.info/developers/data/nyct/turnstile/turnstile_{}.txt"
    dfs = []
    for week_num in week_nums:
        file_url = url.format(week_num)
        dfs.append(pd.read_csv(file_url))
    return pd.concat(dfs)
```

[3]:
```
# The dataset used running data for the months of July, August and Septemeber␣
 ↪of 2021
data_weeks = [210703,210710,210717,210724,210731,
              210807,210814,210821,210828,
              210904,210911,210918,210925,
              211002]
get_data(data_weeks)
```

[3]:
```
         C/A  UNIT       SCP        STATION LINENAME DIVISION        DATE  \
0       A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
1       A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
2       A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
```

```
3         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
4         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
…          …     …        …               …       …        …
210206  TRAM2  R469  00-05-01  RIT-ROOSEVELT         R      RIT  10/01/2021
210207  TRAM2  R469  00-05-01  RIT-ROOSEVELT         R      RIT  10/01/2021
210208  TRAM2  R469  00-05-01  RIT-ROOSEVELT         R      RIT  10/01/2021
210209  TRAM2  R469  00-05-01  RIT-ROOSEVELT         R      RIT  10/01/2021
210210  TRAM2  R469  00-05-01  RIT-ROOSEVELT         R      RIT  10/01/2021

            TIME     DESC  ENTRIES  \
0       00:00:00  REGULAR  7592792
1       04:00:00  REGULAR  7592804
2       08:00:00  REGULAR  7592816
3       12:00:00  REGULAR  7592870
4       16:00:00  REGULAR  7592992
…            …        …       …
210206  05:00:00  REGULAR     5554
210207  09:00:00  REGULAR     5554
210208  13:00:00  REGULAR     5554
210209  17:00:00  REGULAR     5554
210210  21:00:00  REGULAR     5554

        EXITS
0                                                            2595706
1                                                            2595713
2                                                            2595729
3                                                            2595762
4                                                            2595791
…                                                              …
210206                                                           649
210207                                                           649
210208                                                           649
210209                                                           649
210210                                                           650

[2934629 rows x 11 columns]
```

[4]:
```
#Storing the dataset in pandas dataframe to be able to manipulate it
my_df = get_data(data_weeks)
engine = create_engine("sqlite:///MTA_db.db")
my_df.to_sql('MTA_table', engine, if_exists = 'replace', index=False)
```

[5]:
```
tables = engine.table_names()
print(tables)
```

```
['MTA_table']
```

```
/var/folders/4y/plz6nn617g3gccj1l9_90bhc0000gn/T/ipykernel_3542/1698560817.py:1:
```

```
SADeprecationWarning: The Engine.table_names() method is deprecated and will be
removed in a future release.  Please refer to Inspector.get_table_names().
(deprecated since: 1.4)
  tables = engine.table_names()
```

```
[6]: my_df=pd.read_sql('select * from MTA_table', engine)
     my_df.tail()
```

```
[6]:           C/A  UNIT       SCP        STATION LINENAME DIVISION       DATE  \
     2934624  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
     2934625  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
     2934626  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
     2934627  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
     2934628  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021

                  TIME     DESC  ENTRIES  \
     2934624  05:00:00  REGULAR     5554
     2934625  09:00:00  REGULAR     5554
     2934626  13:00:00  REGULAR     5554
     2934627  17:00:00  REGULAR     5554
     2934628  21:00:00  REGULAR     5554

              EXITS
     2934624                                               649
     2934625                                               649
     2934626                                               649
     2934627                                               649
     2934628                                               650
```

```
[7]: #Removing the leading and trailing whitespaces using strip() function
     my_df.columns.str.strip()
```

```
[7]: Index(['C/A', 'UNIT', 'SCP', 'STATION', 'LINENAME', 'DIVISION', 'DATE', 'TIME',
            'DESC', 'ENTRIES', 'EXITS'],
           dtype='object')
```

```
[8]: #The dataframe below represents a snippet of the data (the first 4 rows)
     my_df.head()
```

```
[8]:    C/A  UNIT       SCP STATION LINENAME DIVISION       DATE      TIME  \
     0  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  00:00:00
     1  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  04:00:00
     2  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  08:00:00
     3  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  12:00:00
     4  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  16:00:00

           DESC  ENTRIES  \
     0  REGULAR  7592792
```

```
1  REGULAR  7592804
2  REGULAR  7592816
3  REGULAR  7592870
4  REGULAR  7592992

   EXITS
0                                          2595706
1                                          2595713
2                                          2595729
3                                          2595762
4                                          2595791
```

- We can create a new column to combine both the date and time

```
[9]: my_df['DATETIME'] = pd.to_datetime(my_df.DATE + ' ' + my_df.TIME)
```

```
[10]: my_df.head()
```

```
[10]:     C/A  UNIT       SCP STATION LINENAME DIVISION        DATE      TIME  \
0  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  00:00:00
1  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  04:00:00
2  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  08:00:00
3  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  12:00:00
4  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  16:00:00

      DESC  ENTRIES  \
0  REGULAR  7592792
1  REGULAR  7592804
2  REGULAR  7592816
3  REGULAR  7592870
4  REGULAR  7592992

   EXITS                                                                  \
0                                          2595706
1                                          2595713
2                                          2595729
3                                          2595762
4                                          2595791

             DATETIME
0 2021-06-26 00:00:00
1 2021-06-26 04:00:00
2 2021-06-26 08:00:00
3 2021-06-26 12:00:00
4 2021-06-26 16:00:00
```

```
[11]: #Check for NA vlaues in the dataset
      #The following two runs shows that there are no observations of type 'NA'
```

```
my_df.isna()
```

```
[11]:            C/A   UNIT    SCP  STATION  LINENAME  DIVISION   DATE   TIME  \
      0        False  False  False    False     False     False  False  False
      1        False  False  False    False     False     False  False  False
      2        False  False  False    False     False     False  False  False
      3        False  False  False    False     False     False  False  False
      4        False  False  False    False     False     False  False  False
      …          …     …     …      …       …     …      …
      2934624  False  False  False    False     False     False  False  False
      2934625  False  False  False    False     False     False  False  False
      2934626  False  False  False    False     False     False  False  False
      2934627  False  False  False    False     False     False  False  False
      2934628  False  False  False    False     False     False  False  False

                DESC  ENTRIES  \
      0        False    False
      1        False    False
      2        False    False
      3        False    False
      4        False    False
      …          …     …
      2934624  False    False
      2934625  False    False
      2934626  False    False
      2934627  False    False
      2934628  False    False

               EXITS                                                  \
      0                                              False
      1                                              False
      2                                              False
      3                                              False
      4                                              False
      …                                              …
      2934624                                        False
      2934625                                        False
      2934626                                        False
      2934627                                        False
      2934628                                        False

               DATETIME
      0           False
      1           False
      2           False
      3           False
      4           False
```

```
…           …
2934624     False
2934625     False
2934626     False
2934627     False
2934628     False

[2934629 rows x 12 columns]
```

[12]:
```
my_df.isna().sum()
```

[12]:
```
C/A                                                                   0
UNIT                                                                  0
SCP                                                                   0
STATION                                                               0
LINENAME                                                              0
DIVISION                                                              0
DATE                                                                  0
TIME                                                                  0
DESC                                                                  0
ENTRIES                                                               0
EXITS                                                                 0
DATETIME                                                              0
dtype: int64
```

[13]:
```
#sorting values in an ascending order to check for duplicates per turnstile per␣
 ↪date
#The following 3 runs check for any duplicate observations within the time␣
 ↪range used for this analysis

my_df.sort_values(['C/A','UNIT','SCP','STATION','DATETIME'], ascending = True)
```

[13]:
```
           C/A  UNIT       SCP        STATION LINENAME DIVISION       DATE  \
0         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
1         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
2         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
3         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
4         A002  R051  02-00-00          59 ST  NQR456W      BMT  06/26/2021
…          …    …         …            …        …        …          …
2934624  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
2934625  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
2934626  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
2934627  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021
2934628  TRAM2  R469  00-05-01  RIT-ROOSEVELT        R      RIT  10/01/2021

             TIME     DESC  ENTRIES  \
0        00:00:00  REGULAR  7592792
```

```
1        04:00:00  REGULAR  7592804
2        08:00:00  REGULAR  7592816
3        12:00:00  REGULAR  7592870
4        16:00:00  REGULAR  7592992
…             …        …       …
2934624  05:00:00  REGULAR     5554
2934625  09:00:00  REGULAR     5554
2934626  13:00:00  REGULAR     5554
2934627  17:00:00  REGULAR     5554
2934628  21:00:00  REGULAR     5554

         EXITS                                                     \
0                                              2595706
1                                              2595713
2                                              2595729
3                                              2595762
4                                              2595791
…                                                  …
2934624                                            649
2934625                                            649
2934626                                            649
2934627                                            649
2934628                                            650

                    DATETIME
0       2021-06-26 00:00:00
1       2021-06-26 04:00:00
2       2021-06-26 08:00:00
3       2021-06-26 12:00:00
4       2021-06-26 16:00:00
…                     …
2934624 2021-10-01 05:00:00
2934625 2021-10-01 09:00:00
2934626 2021-10-01 13:00:00
2934627 2021-10-01 17:00:00
2934628 2021-10-01 21:00:00

[2934629 rows x 12 columns]
```

[14]:
```
my_df.duplicated().sum()
```

[14]:
```
0
```

[15]:
```
#Locating the duplicate rows if any
my_df.loc[my_df.duplicated(), :]
```

```
[15]: Empty DataFrame
      Columns: [C/A, UNIT, SCP, STATION, LINENAME, DIVISION, DATE, TIME, DESC,
      ENTRIES, EXITS                                                           ,
      DATETIME]
      Index: []
```

```
[16]: #Get the count of entries for each Turnstile and check for duplicates
      #The results shows that there are two entries for the same turnstile in the␣
       ↪same DATETIME series
      my_df.groupby(["C/A", "UNIT", "SCP", "STATION", "DATETIME"]).ENTRIES.count().
       ↪reset_index().sort_values("ENTRIES", ascending=False)
```

```
[16]:           C/A  UNIT       SCP        STATION            DATETIME  ENTRIES
      2911468   S101  R070  00-00-02     ST. GEORGE 2021-09-15 08:00:00        2
      2911391   S101  R070  00-00-02     ST. GEORGE 2021-09-02 20:00:00        2
      2911401   S101  R070  00-00-02     ST. GEORGE 2021-09-04 12:00:00        2
      2911400   S101  R070  00-00-02     ST. GEORGE 2021-09-04 08:00:00        2
      2911398   S101  R070  00-00-02     ST. GEORGE 2021-09-04 00:00:00        2
      …          …    …        …             …                  …      …
      978170    N128  R200  00-00-03      EUCLID AV 2021-07-31 04:00:00        1
      978171    N128  R200  00-00-03      EUCLID AV 2021-07-31 08:00:00        1
      978172    N128  R200  00-00-03      EUCLID AV 2021-07-31 12:00:00        1
      978173    N128  R200  00-00-03      EUCLID AV 2021-07-31 16:00:00        1
      2934481  TRAM2  R469  00-05-01  RIT-ROOSEVELT 2021-10-01 21:00:00        1

      [2934482 rows x 6 columns]
```

```
[17]: import datetime

      test = ((my_df['C/A'] == 'R516') &
              (my_df['UNIT'] == 'R291') &
              (my_df['SCP'] == '00-00-02') &
              (my_df['STATION'] == '33 ST-RAWSON ST') &
              (my_df['DATETIME'].dt.date == datetime.datetime(2021, 8 , 25).date()))
```

```
[18]: my_df[test].tail()
```

```
[18]:          C/A  UNIT       SCP          STATION LINENAME DIVISION        DATE  \
      1864708  R516  R291  00-00-02  33 ST-RAWSON ST        7      IRT  08/25/2021
      1864709  R516  R291  00-00-02  33 ST-RAWSON ST        7      IRT  08/25/2021
      1864710  R516  R291  00-00-02  33 ST-RAWSON ST        7      IRT  08/25/2021
      1864711  R516  R291  00-00-02  33 ST-RAWSON ST        7      IRT  08/25/2021
      1864712  R516  R291  00-00-02  33 ST-RAWSON ST        7      IRT  08/25/2021

                   TIME         DESC   ENTRIES  \
      1864708  12:00:00  RECOVR AUD   1535620
      1864709  16:00:00     REGULAR  10991727
```

```
1864710  16:00:00  RECOVR AUD   1535734
1864711  20:00:00     REGULAR  10991870
1864712  20:00:00  RECOVR AUD   1535925

         EXITS                                                      \
1864708                                            1649485
1864709                                            8495220
1864710                                            1649537
1864711                                            8495226
1864712                                            1649602

                   DATETIME
1864708 2021-08-25 12:00:00
1864709 2021-08-25 16:00:00
1864710 2021-08-25 16:00:00
1864711 2021-08-25 20:00:00
1864712 2021-08-25 20:00:00
```

- The results above shows that there are more than 1 row for the same turnstile per entries, the difference between the two entries are the DESC column which has a Reguler audit and a Recovered Audit.
- For the sake of simplicity, the Recovered Audit columns will be dropped.

[19]:
```
my_df.drop(my_df[my_df['DESC'] == 'RECOVR AUD'].index , axis=0, inplace=True)
```

[20]:
```
#checking the drop was done correctly
my_df[test].tail()
```

```
/var/folders/4y/plz6nn617g3gccj1l9_90bhc0000gn/T/ipykernel_3542/2925128706.py:2:
UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  my_df[test].tail()
```

[20]:
```
          C/A  UNIT       SCP         STATION LINENAME DIVISION        DATE  \
1864703  R516  R291  00-00-02  33 ST-RAWSON ST        7       IRT  08/25/2021
1864705  R516  R291  00-00-02  33 ST-RAWSON ST        7       IRT  08/25/2021
1864707  R516  R291  00-00-02  33 ST-RAWSON ST        7       IRT  08/25/2021
1864709  R516  R291  00-00-02  33 ST-RAWSON ST        7       IRT  08/25/2021
1864711  R516  R291  00-00-02  33 ST-RAWSON ST        7       IRT  08/25/2021

             TIME     DESC   ENTRIES  \
1864703  04:00:00  REGULAR  10991599
1864705  08:00:00  REGULAR  10991607
1864707  12:00:00  REGULAR  10991639
1864709  16:00:00  REGULAR  10991727
1864711  20:00:00  REGULAR  10991870

         EXITS                                                      \
1864703                                            8495017
```

```
1864705                                             8495122
1864707                                             8495203
1864709                                             8495220
1864711                                             8495226

                   DATETIME
1864703 2021-08-25 04:00:00
1864705 2021-08-25 08:00:00
1864707 2021-08-25 12:00:00
1864709 2021-08-25 16:00:00
1864711 2021-08-25 20:00:00
```

[21]:
```
#checking for duplicate observations again
my_df.groupby(["C/A", "UNIT", "SCP", "STATION", "DATETIME"]).ENTRIES.count().
 ↪reset_index().sort_values("ENTRIES", ascending=False)
```

[21]:
```
           C/A  UNIT      SCP         STATION            DATETIME  ENTRIES
0         A002  R051  02-00-00           59 ST 2021-06-26 00:00:00        1
1946622   R138  R293  00-06-00  34 ST-PENN STA 2021-09-17 14:00:00        1
1946602   R138  R293  00-06-00  34 ST-PENN STA 2021-09-14 06:00:00        1
1946603   R138  R293  00-06-00  34 ST-PENN STA 2021-09-14 10:00:00        1
1946604   R138  R293  00-06-00  34 ST-PENN STA 2021-09-14 14:00:00        1
…          …     …       …              …                 …        …
973306    N128  R200  00-00-03       EUCLID AV 2021-09-13 00:00:00        1
973307    N128  R200  00-00-03       EUCLID AV 2021-09-13 04:00:00        1
973308    N128  R200  00-00-03       EUCLID AV 2021-09-13 08:00:00        1
973309    N128  R200  00-00-03       EUCLID AV 2021-09-13 12:00:00        1
2919911  TRAM2  R469  00-05-01   RIT-ROOSEVELT 2021-10-01 21:00:00        1

[2919912 rows x 6 columns]
```

- The following steps will focus on understanding the data and discover any anomalies

[22]:
```
my_df.DESC.value_counts()
```

[22]:
```
REGULAR    2919912
Name: DESC, dtype: int64
```

[23]:
```
my_df.shape
```

[23]:
```
(2919912, 12)
```

[24]:
```
my_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919912 entries, 0 to 2934628
Data columns (total 12 columns):
 #   Column                                                              Dtype
---  ------                                                              -----
```

```
 0   C/A
object
 1   UNIT
object
 2   SCP
object
 3   STATION
object
 4   LINENAME
object
 5   DIVISION
object
 6   DATE
object
 7   TIME
object
 8   DESC
object
 9   ENTRIES                                                                int64
 10  EXITS                                                                  int64
 11  DATETIME
datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(9)
memory usage: 289.6+ MB
```

[25]:
```
#looking at the descriptive values for the commulative Entries and Exists as a␣
 ↪ballpark number doesn't make sense,
#so we need to get the unique daily Entries to understand the problem statment␣
 ↪better. P.S. the Exits won't be
#looked at in this analysis as our focus is to get a sense of the traffic per␣
 ↪station per day and it can be deduced
#fairly from the daily Entries only.
my_df.describe()
```

[25]:
```
            ENTRIES  \
count  2.919912e+06
mean   4.149555e+07
std    2.181855e+08
min    0.000000e+00
25%    2.203418e+05
50%    1.397294e+06
75%    5.997214e+06
max    2.147407e+09

       EXITS
count                                      2.919912e+06
mean                                       3.322832e+07
```

```
std                                           1.918885e+08
min                                           0.000000e+00
25%                                           1.033300e+05
50%                                           8.541220e+05
75%                                           3.974544e+06
max                                           2.133797e+09
```

- The following few steps is to create a new column for unique daily entries and to check for any anomalies

[28]:
```
#creating a turnstile_id for simplicity of coding
my_df['Turnstile_ID'] = my_df['C/A'].astype(str) + '_' + my_df.UNIT.astype(str)␣
 ↪+ '_' + \
my_df.SCP.astype(str) + '_' + my_df.STATION.astype(str)
```

[30]:
```
my_df.head()
```

[30]:
```
    C/A  UNIT       SCP STATION LINENAME DIVISION        DATE      TIME  \
0  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  00:00:00
1  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  04:00:00
2  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  08:00:00
3  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  12:00:00
4  A002  R051  02-00-00   59 ST  NQR456W      BMT  06/26/2021  16:00:00

      DESC  ENTRIES  \
0  REGULAR  7592792
1  REGULAR  7592804
2  REGULAR  7592816
3  REGULAR  7592870
4  REGULAR  7592992

   EXITS                                                                  \
0                                           2595706
1                                           2595713
2                                           2595729
3                                           2595762
4                                           2595791

             DATETIME             Turnstile_ID
0 2021-06-26 00:00:00  A002_R051_02-00-00_59 ST
1 2021-06-26 04:00:00  A002_R051_02-00-00_59 ST
2 2021-06-26 08:00:00  A002_R051_02-00-00_59 ST
3 2021-06-26 12:00:00  A002_R051_02-00-00_59 ST
4 2021-06-26 16:00:00  A002_R051_02-00-00_59 ST
```

[31]:
```
#For the purpose of this analysis, looking at the traffic per station is enough␣
 ↪to take decisions where to start
```

```
#the work to install the Wi-Fi Netwrok as opposed to looking at it at a granual␣
↪level per Turnstile
station_df = my_df.groupby(['STATION','DATE'])['ENTRIES'].max().reset_index()

station_df
```

[31]:
```
          STATION        DATE    ENTRIES
0            1 AV  06/26/2021  370891078
1            1 AV  06/27/2021  370891152
2            1 AV  06/28/2021  370891228
3            1 AV  06/29/2021  370891317
4            1 AV  06/30/2021  370891419
…             …          …         …
37087   ZEREGA AV  09/27/2021    1308054
37088   ZEREGA AV  09/28/2021    1308726
37089   ZEREGA AV  09/29/2021    1309387
37090   ZEREGA AV  09/30/2021    1310068
37091   ZEREGA AV  10/01/2021    1310763

[37092 rows x 3 columns]
```

[37]:
```
station_df.sort_values(['STATION','DATE'], ascending=True)

station_df
```

[37]:
```
          STATION        DATE    ENTRIES
0            1 AV  06/26/2021  370891078
1            1 AV  06/27/2021  370891152
2            1 AV  06/28/2021  370891228
3            1 AV  06/29/2021  370891317
4            1 AV  06/30/2021  370891419
…             …          …         …
37087   ZEREGA AV  09/27/2021    1308054
37088   ZEREGA AV  09/28/2021    1308726
37089   ZEREGA AV  09/29/2021    1309387
37090   ZEREGA AV  09/30/2021    1310068
37091   ZEREGA AV  10/01/2021    1310763

[37092 rows x 3 columns]
```

[38]:
```
station_df[['PREV_DATE','PREV_ENTRIES']] = station_df.
↪groupby(['STATION'])['DATE','ENTRIES'] \
.apply(lambda x: x.shift(periods = 1, axis = 0, fill_value = 0))

station_df
```

```
/var/folders/4y/plz6nn617g3gccj1l9_90bhc0000gn/T/ipykernel_3542/518037491.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  station_df[['PREV_DATE','PREV_ENTRIES']] =
station_df.groupby(['STATION'])['DATE','ENTRIES'] \
```

[38]:
```
          STATION        DATE    ENTRIES   PREV_DATE  PREV_ENTRIES
0            1 AV  06/26/2021  370891078           0             0
1            1 AV  06/27/2021  370891152  06/26/2021     370891078
2            1 AV  06/28/2021  370891228  06/27/2021     370891152
3            1 AV  06/29/2021  370891317  06/28/2021     370891228
4            1 AV  06/30/2021  370891419  06/29/2021     370891317
…               …           …          …           …           …
37087   ZEREGA AV  09/27/2021    1308054  09/26/2021       1307428
37088   ZEREGA AV  09/28/2021    1308726  09/27/2021       1308054
37089   ZEREGA AV  09/29/2021    1309387  09/28/2021       1308726
37090   ZEREGA AV  09/30/2021    1310068  09/29/2021       1309387
37091   ZEREGA AV  10/01/2021    1310763  09/30/2021       1310068

[37092 rows x 5 columns]
```

[39]:
```python
#Now let add one more column in the dataframe to get the unique daily entries
station_df['DAILY_ENTRIES'] = station_df.ENTRIES - station_df.PREV_ENTRIES

station_df
```

[39]:
```
          STATION        DATE    ENTRIES   PREV_DATE  PREV_ENTRIES  \
0            1 AV  06/26/2021  370891078           0             0
1            1 AV  06/27/2021  370891152  06/26/2021     370891078
2            1 AV  06/28/2021  370891228  06/27/2021     370891152
3            1 AV  06/29/2021  370891317  06/28/2021     370891228
4            1 AV  06/30/2021  370891419  06/29/2021     370891317
…               …           …          …           …           …
37087   ZEREGA AV  09/27/2021    1308054  09/26/2021       1307428
37088   ZEREGA AV  09/28/2021    1308726  09/27/2021       1308054
37089   ZEREGA AV  09/29/2021    1309387  09/28/2021       1308726
37090   ZEREGA AV  09/30/2021    1310068  09/29/2021       1309387
37091   ZEREGA AV  10/01/2021    1310763  09/30/2021       1310068

        DAILY_ENTRIES
0           370891078
1                  74
2                  76
3                  89
4                 102
…                   …
37087             626
37088             672
37089             661
37090             681
```

```
37091            695

[37092 rows x 6 columns]
```

[40]:
```
#Now let check the descriptive figures for the unique daily entries
station_df['DAILY_ENTRIES'].describe()
```

[40]:
```
count    3.709200e+04
mean     3.526291e+06
std      7.681926e+07
min     -1.942719e+09
25%      1.220000e+02
50%      3.820000e+02
75%      6.820000e+02
max      2.147407e+09
Name: DAILY_ENTRIES, dtype: float64
```

- The unique daily entries shows negative minimum value, which is not correct given that the daily entries should be cummulative

[41]:
```
station_df.dtypes
```

[41]:
```
STATION           object
DATE              object
ENTRIES            int64
PREV_DATE         object
PREV_ENTRIES       int64
DAILY_ENTRIES      int64
dtype: object
```

[42]:
```
station_df.PREV_ENTRIES = station_df.PREV_ENTRIES.astype(int)
station_df.DAILY_ENTRIES = station_df.DAILY_ENTRIES.astype(int)
```

[44]:
```
#lets take a subset of the dataframe to check how are the rows of negative␣
↪value looking like

station_df[station_df.DAILY_ENTRIES < 0]
```

[44]:
```
              STATION        DATE     ENTRIES   PREV_DATE  PREV_ENTRIES  \
295            104 ST  06/27/2021  1681050247  06/26/2021    1681050263
296            104 ST  06/28/2021  1681050222  06/27/2021    1681050247
297            104 ST  06/29/2021  1681050128  06/28/2021    1681050222
298            104 ST  06/30/2021  1681050016  06/29/2021    1681050128
299            104 ST  07/01/2021  1681049915  06/30/2021    1681050016
…                 …           …         …           …           …
34644  TIMES SQ-42 ST  10/01/2021  1891091412  09/30/2021    1891091983
35080        UTICA AV  08/10/2021    17025492  08/09/2021     102825308
35117        UTICA AV  09/16/2021    17089093  09/15/2021     102825371
```

```
36462     WINTHROP ST  08/20/2021     1961869  08/19/2021       19163917
36466     WINTHROP ST  08/24/2021     7185669  08/23/2021       19167678

       DAILY_ENTRIES
295              -16
296              -25
297              -94
298             -112
299             -101
…                 …
34644           -571
35080      -85799816
35117      -85736278
36462      -17202048
36466      -11982009

[2762 rows x 6 columns]
```

- I am observing two abnormal patterns here:
  1. The first one is the negative daily entries (which is most likly to be erronous entries where passengers reversed the turnstile wheel or somthing of that sort).
  2. The second one is the unexpected big difference between the ENTRIES and the PREV_ENTRIES, where ENTRIES is a lot more smaller than the PREV_ENTRIES and here I will assume if something like this happens then the commulative counter has been reset.
- In the next few steps, we will be working on fixing these anomalies.

[54]:
```
station_df.dtypes
```

[54]:
```
STATION          object
DATE             object
ENTRIES           int64
PREV_DATE        object
PREV_ENTRIES      int64
DAILY_ENTRIES     int64
dtype: object
```

[57]:
```
station_df['DATE'] = pd.to_datetime(station_df['DATE'])

#df['Date'] = pd.to_datetime(df['Date'])
#df['Month'] = df['Date'].dt.month
```

[58]:
```
station_df['PREV_DATE'] = pd.to_datetime(station_df['PREV_DATE'])
```

[59]:
```
station_df.dtypes
```

[59]:
```
STATION                  object
DATE             datetime64[ns]
```

```
ENTRIES                   int64
PREV_DATE        datetime64[ns]
PREV_ENTRIES              int64
DAILY_ENTRIES             int64
dtype: object
```

[66]:
```
station_df[(station_df['STATION'] == 'UTICA AV') & (station_df['DATE'] >=␣
↪'2021-08-01') \
          & (station_df['DATE'] <= '2021-08-31')]

#created a subset of the UTICA AV station for the month of August, where I saw␣
↪one of the anomalies above;
#to try and understand the problem at hand
```

[66]:
```
        STATION        DATE    ENTRIES  PREV_DATE  PREV_ENTRIES  DAILY_ENTRIES
35071  UTICA AV  2021-08-01  102822537 2021-07-31     102822333            204
35072  UTICA AV  2021-08-02  102822960 2021-08-01     102822537            423
35073  UTICA AV  2021-08-03  102823496 2021-08-02     102822960            536
35074  UTICA AV  2021-08-04  102823961 2021-08-03     102823496            465
35075  UTICA AV  2021-08-05  102824368 2021-08-04     102823961            407
35076  UTICA AV  2021-08-06  102824770 2021-08-05     102824368            402
35077  UTICA AV  2021-08-07  102824993 2021-08-06     102824770            223
35078  UTICA AV  2021-08-08  102825159 2021-08-07     102824993            166
35079  UTICA AV  2021-08-09  102825308 2021-08-08     102825159            149
35080  UTICA AV  2021-08-10   17025492 2021-08-09     102825308      -85799816
35081  UTICA AV  2021-08-11   17027428 2021-08-10      17025492           1936
35082  UTICA AV  2021-08-12   17029328 2021-08-11      17027428           1900
35083  UTICA AV  2021-08-13   17031197 2021-08-12      17029328           1869
35084  UTICA AV  2021-08-14   17032568 2021-08-13      17031197           1371
35085  UTICA AV  2021-08-15   17033846 2021-08-14      17032568           1278
35086  UTICA AV  2021-08-16   17035644 2021-08-15      17033846           1798
35087  UTICA AV  2021-08-17   17037569 2021-08-16      17035644           1925
35088  UTICA AV  2021-08-18   17039469 2021-08-17      17037569           1900
35089  UTICA AV  2021-08-19   17041353 2021-08-18      17039469           1884
35090  UTICA AV  2021-08-20   17043176 2021-08-19      17041353           1823
35091  UTICA AV  2021-08-21   17044567 2021-08-20      17043176           1391
35092  UTICA AV  2021-08-22   17045401 2021-08-21      17044567            834
35093  UTICA AV  2021-08-23   17046994 2021-08-22      17045401           1593
35094  UTICA AV  2021-08-24   17048945 2021-08-23      17046994           1951
35095  UTICA AV  2021-08-25   17050832 2021-08-24      17048945           1887
35096  UTICA AV  2021-08-26   17052835 2021-08-25      17050832           2003
35097  UTICA AV  2021-08-27   17054769 2021-08-26      17052835           1934
35098  UTICA AV  2021-08-28   17056223 2021-08-27      17054769           1454
35099  UTICA AV  2021-08-29   17057564 2021-08-28      17056223           1341
35100  UTICA AV  2021-08-30   17059443 2021-08-29      17057564           1879
35101  UTICA AV  2021-08-31   17061365 2021-08-30      17059443           1922
```

[70]:
```
#ok lets check the maximum daily entries in the datframe then assign a␣
 ↪threshold accordingly to reset
#the counter to zero if we encounter a value thats even bigger than out␣
 ↪threshold

mask = station_df.groupby(['STATION','DATE'])['DAILY_ENTRIES'].max().
 ↪reset_index(). \
sort_values('DAILY_ENTRIES')

mask
```

[70]:
```
                STATION        DATE  DAILY_ENTRIES
812              121 ST  2021-07-24    -1942719354
816              121 ST  2021-07-28    -1942718733
17094   CANARSIE-ROCKAW  2021-08-27    -1526315613
17123   CANARSIE-ROCKAW  2021-09-25    -1526304935
21091   FLATBUSH AV-B.C  2021-08-10    -1374037426
…                   …           …              …
13700     BAYCHESTER AV  2021-06-26     2064066499
18893          COURT SQ  2021-06-26     2066596158
17619       CHAMBERS ST  2021-06-26     2116123427
1274              14 ST  2021-06-26     2128673609
23494          HEWES ST  2021-06-26     2147407029

[37092 rows x 3 columns]
```

[72]:
```
def get_daily_counts(row, max_counter):
    counter = row["ENTRIES"] - row["PREV_ENTRIES"]
    if counter < 0:
        counter = -counter
    if counter > max_counter:
        print(row["ENTRIES"], row["PREV_ENTRIES"])
        counter = min(row["ENTRIES"], row["PREV_ENTRIES"])
    if counter > max_counter:
        return 0
    return counter

station_df["DAILY_ENTRIES"] = station_df.apply(get_daily_counts, axis=1,␣
 ↪max_counter=1000000)
```

```
370891078 0
26181733 0
16478308 0
1681050263 0
14759168 0
50331896 0
185712452 0
1160300472 0
```

```
1946272192 0
3588552 1946307906
1946307906 3588742
3589173 1946307906
1946307918 3589617
1862989314 0
1828826573 0
855499479 1828837016
1828838087 855499479
18110479 0
991789060 0
2128673609 0
71309890 0
135439844 0
168165376 0
117444360 168165376
168165376 117444360
117444362 168165376
168165376 117444362
118374824 0
9411864 0
4283666 0
1711662138 0
9408595 0
1827866929 0
53804509 0
7603219 0
934214846 0
688206350 0
4690122 0
218418454 0
8006802 0
1946899327 0
102509856 0
1610418813 0
1475181684 0
1781739185 0
1441542729 0
654494788 0
67422098 0
8195885 0
135364871 0
3572194 0
16852032 0
4439476 0
906519329 0
1953908903 0
2000315733 0
```

```
16256532 0
87066800 0
8975569 0
5311973 0
11770280 0
1174478619 0
8207442 0
2046134325 0
1309228109 0
12706329 0
120082116 0
117440512 0
10982525 117440512
117440512 10983176
1253846890 0
5228851 0
1569885666 0
17863225 0
101226828 0
4549946 0
12596977 0
21726349 0
1834129654 0
2034035190 0
5981788 0
6205367 0
14087658 0
1925733891 0
15436315 0
223959537 0
14590447 0
119790018 0
6578939 119823341
119824435 6578939
136917584 0
335854320 0
569526656 0
9182669 0
101334682 0
7470563 0
7868122 0
885595278 0
1040921584 0
117520314 1040931990
1398458792 0
1208909703 0
1843771080 0
12510966 0
```

```
9889659 0
100663299 0
11591493 0
17689146 0
9917687 0
13159155 0
9983299 0
689473925 0
117442922 0
8986389 0
297605156 0
4453577 0
22569890 0
569491707 0
54982665 0
69046754 0
117440858 0
83907836 0
8526845 0
8410935 0
909043036 0
37621254 909043641
36570482 0
4694987 0
12129706 0
23570953 0
1189019447 23689857
646572814 0
23169825 646581434
646583743 23169825
3172708 0
5969503 0
19738774 0
2442028 0
67109632 0
51149054 67109640
5986209 0
36569106 0
11576096 0
654313478 0
1932400 0
117440724 0
4131859 0
9535212 0
100663296 0
1179269015 0
8934393 0
1325333385 0
```

```
3222240 0
1928603510 0
8173196 0
875171205 0
1560439089 0
154194177 0
2064066499 0
117440512 0
2437507 0
1354766 0
1374029 0
117440965 0
12533549 117440985
117440987 12533759
5604268 0
3376650 0
1928243 0
25799003 0
167790157 25859746
5294004 0
87148383 0
11691679 0
5796298 0
67436813 0
13218335 0
101218115 0
5470076 0
7089988 0
1289818231 0
12499188 0
16379306 0
1110507 0
6034949 0
17913288 0
22037061 0
990381610 0
8836724 0
389169819 0
3277594 0
6633028 0
607993803 0
4430067 0
1722625870 0
1560335770 0
34020206 1560335819
1560335842 34029604
34030907 1560335842
2043327134 0
```

```
9719309 0
14314945 0
3030185 0
52178133 0
2116123427 0
19880920 0
856392879 0
118726667 0
14008883 0
36784604 0
5700270 0
151870279 0
605276324 0
6856435 0
1058665572 0
2066596158 0
2359738 0
6437562 0
118013964 0
184682778 0
4967829 0
1999537633 0
14203498 0
4259110 0
13563384 0
2859918 0
5786360 0
4740203 5822112
5822112 4740203
13957612 0
9793658 0
101260262 0
1559807093 0
1124135763 0
6429367 0
1624510434 0
8995214 0
2786935 0
1383097184 0
9072927 1383110353
17396292 0
23396253 0
1762319069 0
6785569 0
20111054 0
306735962 0
11535457 0
169817662 0
```

```
14961728 0
6398050 0
1662537517 0
6305654 0
15326183 0
11251770 0
20791742 0
55426280 0
1208844885 0
1254630543 0
6071491 0
1359769875 0
74180831 0
8648727 0
2147407029 0
1915867960 0
11677312 0
2573274 0
7802703 0
5948531 7822138
7822138 5948531
19699486 0
589937978 0
1191103372 0
8937110 0
202177627 0
1762020917 0
1291015312 0
1205603810 0
6511520 0
170597358 0
8149817 0
50331648 0
13968508 0
7979566 14045092
14051313 7979671
5929078 0
15999749 0
12572418 16034831
419446441 0
705347143 0
352637709 705357551
14745284 0
5642286 0
4829046 0
9116301 0
1962579921 0
1476923212 0
```

```
1757901102 0
7387927 0
6940353 0
637578981 0
9575583 0
5320021 0
117441150 0
150995161 0
823494354 0
4982284 0
138584132 0
2906407 0
536917974 0
6492294 0
16810602 0
5885626 0
16622705 0
5315206 0
1664738223 0
8659115 0
100663296 0
5608386 0
8700273 0
4035595 0
172379101 0
13405112 0
5666621 0
17983049 0
18257223 0
7820161 0
337349109 0
7503107 0
13074257 0
5898605 0
7276654 0
6731112 0
168542433 0
16349442 0
10423515 0
150997136 0
436905968 0
4327382 0
6263114 0
14435140 0
67268813 0
1569344992 0
51680645 0
19152891 0
```

```
10341292 0
117777645 0
1019339 0
3772203 0
6460441 0
135988116 0
100664160 0
7744566 0
67620014 0
2633107 0
13793912 0
8339749 0
285343137 0
7944205 0
123946410 0
15367482 0
1006930535 0
1129213713 0
17094812 0
12035961 0
84424938 0
71543154 0
2591316 0
9236439 0
1720235844 0
1891101745 1720311340
369309424 0
11882758 0
102808272 0
17025492 102825308
102825371 17077888
17089093 102825371
621873222 0
7452282 0
7011137 0
14933507 0
22552962 0
4411627 0
7659338 0
168727845 0
555471868 0
12890876 0
135540928 0
5311949 0
8403983 0
19095274 0
1961869 19163917
19167678 1962037
```

```
7185669 19167678
19171640 7185669
12559328 0
8962516 0
1580169729 0
730966278 0
17018286 0
1266947 0
```

[73]:
```
station_df['DAILY_ENTRIES'].describe()
```

[73]:
```
count     37092.000000
mean        773.418392
std       13451.616270
min           0.000000
25%         174.000000
50%         417.000000
75%         705.000000
max      952501.000000
Name: DAILY_ENTRIES, dtype: float64
```

[75]:
```
station_df[station_df.DAILY_ENTRIES < 0]
```

[75]:
```
Empty DataFrame
Columns: [STATION, DATE, ENTRIES, PREV_DATE, PREV_ENTRIES, DAILY_ENTRIES]
Index: []
```

- OK! Now that we have cleaned up the dataframem let's answer the following: • What are the busiest train stations across New York City?

[85]:
```
station_df
```

[85]:
```
         STATION        DATE    ENTRIES  PREV_DATE  PREV_ENTRIES  DAILY_ENTRIES
0           1 AV  2021-06-26  370891078 1970-01-01             0              0
1           1 AV  2021-06-27  370891152 2021-06-26     370891078             74
2           1 AV  2021-06-28  370891228 2021-06-27     370891152             76
3           1 AV  2021-06-29  370891317 2021-06-28     370891228             89
4           1 AV  2021-06-30  370891419 2021-06-29     370891317            102
…            …         …          …        …             …           …
37087  ZEREGA AV  2021-09-27    1308054 2021-09-26       1307428            626
37088  ZEREGA AV  2021-09-28    1308726 2021-09-27       1308054            672
37089  ZEREGA AV  2021-09-29    1309387 2021-09-28       1308726            661
37090  ZEREGA AV  2021-09-30    1310068 2021-09-29       1309387            681
37091  ZEREGA AV  2021-10-01    1310763 2021-09-30       1310068            695

[37092 rows x 6 columns]
```

[113]:
```
station_df['DATE'] = pd.to_datetime(station_df['DATE'], errors='coerce')
station_df.dtypes
```

[113]:
```
STATION                   object
DATE              datetime64[ns]
ENTRIES                    int64
PREV_DATE         datetime64[ns]
PREV_ENTRIES               int64
DAILY_ENTRIES              int64
dtype: object
```

[114]:
```
station_df['weekNumber'] = station_df['DATE'].dt.week
```

```
/var/folders/4y/plz6nn617g3gccj1l9_90bhc0000gn/T/ipykernel_3542/3323928275.py:1:
FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated.
Please use Series.dt.isocalendar().week instead.
  station_df['weekNumber'] = station_df['DATE'].dt.week
```

[115]:
```
station_df
```

[115]:
```
         STATION       DATE    ENTRIES  PREV_DATE  PREV_ENTRIES  \
0           1 AV 2021-06-26  370891078 1970-01-01             0
1           1 AV 2021-06-27  370891152 2021-06-26     370891078
2           1 AV 2021-06-28  370891228 2021-06-27     370891152
3           1 AV 2021-06-29  370891317 2021-06-28     370891228
4           1 AV 2021-06-30  370891419 2021-06-29     370891317
…            …          …          …          …            …
37087  ZEREGA AV 2021-09-27    1308054 2021-09-26       1307428
37088  ZEREGA AV 2021-09-28    1308726 2021-09-27       1308054
37089  ZEREGA AV 2021-09-29    1309387 2021-09-28       1308726
37090  ZEREGA AV 2021-09-30    1310068 2021-09-29       1309387
37091  ZEREGA AV 2021-10-01    1310763 2021-09-30       1310068

       DAILY_ENTRIES  weekNumber
0                  0          25
1                 74          25
2                 76          26
3                 89          26
4                102          26
…                …          …
37087            626          39
37088            672          39
37089            661          39
37090            681          39
37091            695          39

[37092 rows x 7 columns]
```

```
[130]: busiest_station = station_df.groupby(['weekNumber','STATION'])['DAILY_ENTRIES'].
        ↪sum().reset_index().\
       sort_values('DAILY_ENTRIES', ascending = False)
       #df.groupby(month('date')).agg({'Revenue': 'sum'})
       #b.groupby(by=[b.index.month, b.index.year])
```

```
[131]: busiest_station
```

```
[131]:       weekNumber           STATION  DAILY_ENTRIES
       258           25    JOURNAL SQUARE         953564
       181           25       CHAUNCEY ST         932938
       235           25      GROVE STREET         822180
       256           25   JFK JAMAICA CT1         751924
       184           25        CITY / BUS         738037
       …           …               …            …
       1470          28   ROCKAWAY PARK B              0
       5446          39      BEACH 105 ST              0
       4952          38            170 ST              0
       405           26            170 ST              0
       3293          33          KINGS HWY              0

       [5684 rows x 3 columns]
```

```
[138]: test1 = busiest_station.groupby(['STATION'])['DAILY_ENTRIES'].mean().
        ↪reset_index().\
       sort_values('DAILY_ENTRIES', ascending = False).head(10)
```

```
[139]: test1
```

```
[139]:              STATION  DAILY_ENTRIES
       256  JFK JAMAICA CT1   74714.466667
       258   JOURNAL SQUARE   72213.733333
       181      CHAUNCEY ST   65311.933333
       235     GROVE STREET   62278.066667
       184       CITY / BUS   54328.333333
       351        THIRTY ST   53497.200000
       298      NEWARK BM BW   46346.066667
       352  THIRTY THIRD ST   46237.466667
       354    TOMPKINSVILLE   45850.533333
       315     PATH NEW WTC   40581.733333
```

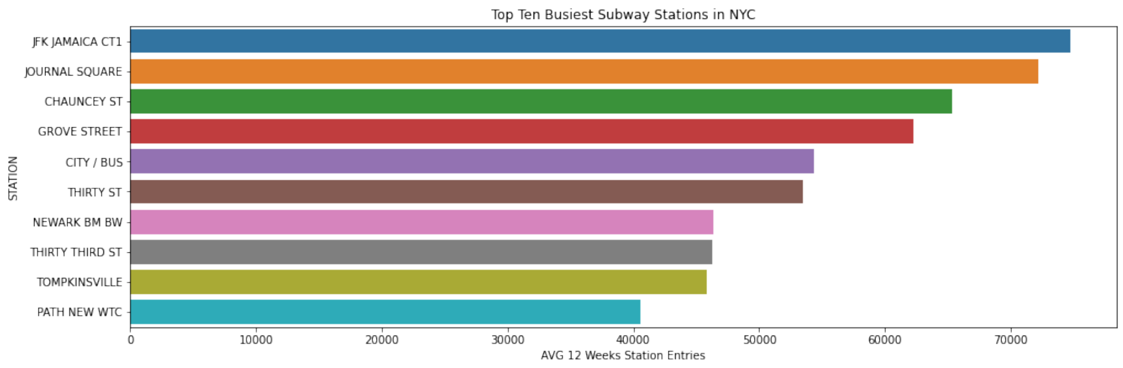
```
[157]: %matplotlib inline
       import matplotlib.pyplot as plt
       import numpy as np
       import seaborn as sns

       fig_dims = (16, 5)
```

```
fig, ax = plt.subplots(figsize=fig_dims)

my_plot = sns.barplot(x ='DAILY_ENTRIES', y='STATION',ax=ax, data=test1)
my_plot.set(xlabel="AVG 12 Weeks Station Entries",title='Top Ten Busiest Subway␣
↪Stations in NYC')
```

```
[157]: [Text(0.5, 0, 'AVG 12 Weeks Station Entries'),
        Text(0.5, 1.0, 'Top Ten Busiest Subway Stations in NYC')]
```



```
[ ]:
```