In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

To read/get the CSV file The 'r' is important to avoid error in the below code utf is not important Save it in something u like

In [2]:
```
uber_23=pd.read_csv(r'C:\Users\Aakasha\Desktop\uber-pickups-in-new-york-city-20220814T121250Z-001\uber-pickups-in-new-york-city/u
```

In [3]:
```
uber_23
```

Out[3]:

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID
0	B02617	2015-05-17 09:47:00	B02617	141
1	B02617	2015-05-17 09:47:00	B02617	65
2	B02617	2015-05-17 09:47:00	B02617	100
3	B02617	2015-05-17 09:47:00	B02774	80
4	B02617	2015-05-17 09:47:00	B02617	90
...	...	...	...	...
14270474	B02765	2015-05-08 15:43:00	B02765	186
14270475	B02765	2015-05-08 15:43:00	B02765	263
14270476	B02765	2015-05-08 15:43:00	B02765	90
14270477	B02765	2015-05-08 15:44:00	B01899	45
14270478	B02765	2015-05-08 15:44:00	B02682	144

14270479 rows × 4 columns

1D is series that is Array 2D is Data that is Table 3D is Panel mostly used

To see only 1st 2 rows

In [4]:
```
uber_23.head(2)
```

Out[4]:

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID
0	B02617	2015-05-17 09:47:00	B02617	141
1	B02617	2015-05-17 09:47:00	B02617	65

To count the No.of duplicates

In [5]:
```
uber_23.duplicated().sum()
```

Out[5]:
```
898225
```

To remove the Duplicate Rows

In [6]:
```
uber_23.drop_duplicates(inplace=True)
```

Take a look at 'inplace' in Drop_duplicates

In [7]:
```
uber_23.shape
```

Out[7]:
```
(13372254, 4)
```

Which month has max uber pickups:

In [8]:
```
uber_23['Pickup_date']=pd.to_datetime(uber_23['Pickup_date'],format='%Y-%m-%d %H:%M:%S')
```

In [9]:
```
uber_23['Pickup_date'].dtype
```

Out[9]:
```
dtype('<M8[ns]')
```

dt=datetime/daytime

In [10]:
```
month_count=uber_23['Pickup_date'].dt.month
```

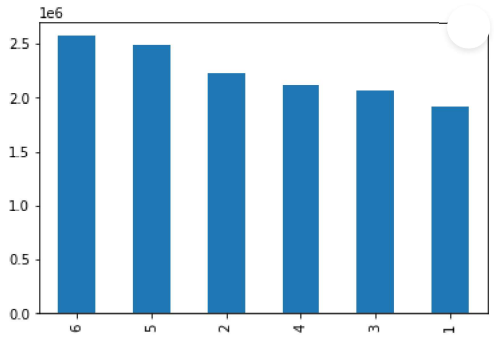
In [11]:
```
month_count.value_counts()
```

Out[11]:
```
6    2571771
5    2483980
2    2222189
4    2112705
3    2062639
1    1918970
Name: Pickup_date, dtype: int64
```

To Represent in graph

In [12]:
```
month_count.value_counts().plot(kind='bar')
```

Out[12]:
```
<AxesSubplot:>
```



Now, We have to find the No.of Trips in Month and Week Days

In [13]:
```
uber_23['month']=uber_23['Pickup_date'].dt.month
uber_23['week_day']=uber_23['Pickup_date'].dt.weekday
uber_23['day_name']=uber_23['Pickup_date'].dt.day_name
uber_23['hour']=uber_23['Pickup_date'].dt.hour
uber_23['minute']=uber_23['Pickup_date'].dt.minute
```

In [14]:
```
uber_23.head(5)
```

Out[14]:

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID	month	week_day	day_name	hour	minute
0	B02617	2015-05-17 09:47:00	B02617	141	5	6	<bound method PandasDelegate._add_delegate_acc...	9	47
1	B02617	2015-05-17 09:47:00	B02617	65	5	6	<bound method PandasDelegate._add_delegate_acc...	9	47
2	B02617	2015-05-17 09:47:00	B02617	100	5	6	<bound method PandasDelegate._add_delegate_acc...	9	47
3	B02617	2015-05-17 09:47:00	B02774	80	5	6	<bound method PandasDelegate._add_delegate_acc...	9	47
4	B02617	2015-05-17 09:47:00	B02617	90	5	6	<bound method PandasDelegate._add_delegate_acc...	9	47

In [15]:
```
uber_23.groupby(['month','week_day']).size()
```

Out[15]:
```
month  week_day
1      0           190606
       1           196574
       2           245650
       3           330319
       4           339285
       5           386049
       6           230487
2      0           274948
       1           287260
       2           286387
       3           335603
       4           373550
       5           368311
       6           296130
3      0           269931
       1           320634
       2           256767
       3           277026
       4           309631
       5           314785
       6           313865
4      0           238429
       1           250632
       2           338015
       3           372522
       4           315002
       5           324545
       6           273560
5      0           255501
       1           290004
       2           316045
       3           337607
       4           430134
       5           464298
       6           390391
6      0           375312
       1           405500
       2           328141
       3           357782
       4           371225
       5           399377
       6           334434
dtype: int64
```

To convert the upper one into DATA FRAME using as_index= False

To see the unique month

In [16]:
```
x=uber_23.groupby(['month','week_day'],as_index=False).size()
```

In [17]:
```
x.head()
```

Out[17]:

	month	week_day	size
0	1	0	190606
1	1	1	196574
2	1	2	245650
3	1	3	330319
4	1	4	339285

To change the month number to text:-

In [18]:
```
x['month'].unique()
```

Out[18]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

In [19]:
```
x.head(5)
```

Out[19]:

	month	week_day	size
0	1	0	190606
1	1	1	196574
2	1	2	245650
3	1	3	330319
4	1	4	339285

In [20]:
```
x_dict={1:'JAN',2:'FEB',3:'MAR',4:'APR',5:'MAY',6:'JUNE'}
```

In [21]:
```
x['month']=x['month'].map(x_dict)
```

In [22]:
```
x.head(4)
```

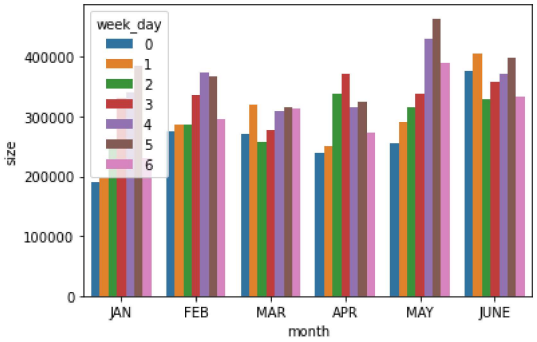
Out[22]:

	month	week_day	size
0	JAN	0	190606
1	JAN	1	196574
2	JAN	2	245650
3	JAN	3	330319

To plot this Data

In [23]:
```
sns.barplot(x='month',y='size',hue='week_day',data=x)
plt.figure(figsize=(12,8))
```

Out[23]:
```
<Figure size 864x576 with 0 Axes>
```




```
<Figure size 864x576 with 0 Axes>
```

To find the rush of the New York city: No.of rides per hour

In [24]:
```
y=uber_23.groupby(['week_day','hour'],as_index=False).size()
```

In [25]:
```
y.head(3)
```

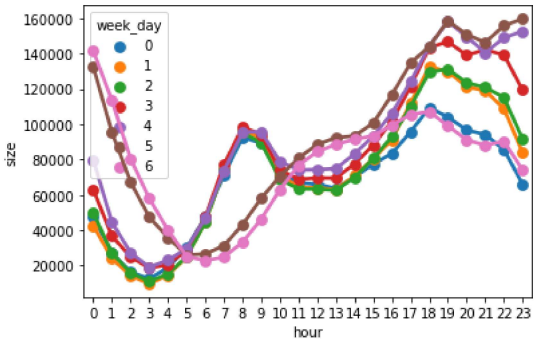
Out[25]:

	week_day	hour	size
0	0	0	47608
1	0	1	27093
2	0	2	16394

If we use Bar Graph, The chart is messy and full of disturbance So, we use POINT PLOT

In [26]:
```
sns.pointplot(x='hour',y='size',hue='week_day',data=y)
plt.figure(figsize=(12,8))
```

Out[26]: <Figure size 864x576 with 0 Axes>



<Figure size 864x576 with 0 Axes>

NOW, OBSERVE THE FIGURE CAREFULLY AND SAY, AT WHAT STAGES THE TRIPS ARE HIGH AND LOW DURING WHICH THE DAYS ACCORDING TO THE HOURS FROM 1-24

WHICH BASE NUMBER HAS THE MOST NUMBER OF ACTIVE VEHICLES.

In [27]:
```
uber_foil=pd.read_csv(r'C:\Users\Aakasha\Desktop\uber-pickups-in-new-york-city-20220626T112836Z-001\uber-pickups-in-new-york-city
```

In [28]:
```
uber_foil.head(3).sort_values('active_vehicles',ascending=False)
```

Out[28]:

	dispatching_base_number	date	active_vehicles	trips
2	B02764	1/1/2015	3427	29421
1	B02765	1/1/2015	225	1765
0	B02512	1/1/2015	190	1132

WE NEED TO FIND THE DISTRIBUTION OF THE BASE NUMBER THROUGH GRAPHS

In [29]:
```
!pip install chart_studio
!pip install plotly
```

```
Requirement already satisfied: chart_studio in c:\python folder\anaconda aakash\lib\site-packages (1.1.0)
Requirement already satisfied: six in c:\python folder\anaconda aakash\lib\site-packages (from chart_studio) (1.15.0)
Requirement already satisfied: plotly in c:\python folder\anaconda aakash\lib\site-packages (from chart_studio) (5.10.0)
Requirement already satisfied: requests in c:\python folder\anaconda aakash\lib\site-packages (from chart_studio) (2.25.1)
Requirement already satisfied: retrying>=1.3.3 in c:\python folder\anaconda aakash\lib\site-packages (from chart_studio) (1.3.
3)
Requirement already satisfied: tenacity>=6.2.0 in c:\python folder\anaconda aakash\lib\site-packages (from plotly->chart_studi
o) (8.0.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\python folder\anaconda aakash\lib\site-packages (from requests->chart_st
udio) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\python folder\anaconda aakash\lib\site-packages (from requests->char
t_studio) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\python folder\anaconda aakash\lib\site-packages (from requests->chart_studio)
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\python folder\anaconda aakash\lib\site-packages (from requests->chart_s
tudio) (2020.12.5)
Requirement already satisfied: plotly in c:\python folder\anaconda aakash\lib\site-packages (5.10.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\python folder\anaconda aakash\lib\site-packages (from plotly) (8.0.1)
```

In [30]:
```
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly.express as px
from plotly.offline import download_plotlyjs, plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)
```

In [31]:
```
px.box(x='dispatching_base_number',y='active_vehicles',data_frame=uber_foil)
```

VIOLIN CHART IS ALSO USEFUL FOR THE DISTRIBUTION

In [32]:
```
px.violin(x='dispatching_base_number',y='active_vehicles',data_frame=uber_foil)
```

NOW,

WE HAVE TO COLLECT THE DATA & MAKE IT READY FOR THE DATA ANALYSIS

In [33]:
```
import os
```

In [34]:
```
os.listdir(r'C:\Users\Aakasha\Desktop\uber-pickups-in-new-york-city-20220626T112836Z-001\uber-pickups-in-new-york-city')
```

Out[34]:
```
['other-American_B01362.csv',
 'other-Carmel_B00256.csv',
 'other-Dial7_B00887.csv',
 'other-Diplo_B01196.csv',
 'other-Federal_02216.csv',
 'other-FHV-services_jan-aug-2015.csv',
 'other-Firstclass_B01536.csv',
 'other-Highclass_B01717.csv',
 'other-Lyft_B02510.csv',
 'other-Prestige_B01338.csv',
 'other-Skyline_B00111.csv',
 'Uber-Jan-Feb-FOIL.csv',
 'uber-raw-data-apr14.csv',
 'uber-raw-data-aug14.csv',
 'uber-raw-data-janjune-15.csv',
 'uber-raw-data-jul14.csv',
 'uber-raw-data-jun14.csv',
 'uber-raw-data-may14.csv',
 'uber-raw-data-sep14.csv']
```

NOW, WE HAVE CONSIDER ONLY THE FILES OF UBER_RAW-DATA- [APR-SEP]. THAT IS THE LAST SEVEN EXCEPT THE JANJUNE-15.CSV FILE

In [35]:
```
files=os.listdir(r'C:\Users\Aakasha\Desktop\uber-pickups-in-new-york-city-20220626T112836Z-001\uber-pickups-in-new-york-city')[-7
```

In [36]:
```
files
```

Out[36]:
```
['uber-raw-data-apr14.csv',
 'uber-raw-data-aug14.csv',
 'uber-raw-data-janjune-15.csv',
 'uber-raw-data-jul14.csv',
 'uber-raw-data-jun14.csv',
 'uber-raw-data-may14.csv',
 'uber-raw-data-sep14.csv']
```

NOW, WE HAVE REMOVE THE UBER-.....JANJUNE-15.CSV FILE

In [37]:
```
files.remove('uber-raw-data-janjune-15.csv')
```

In [38]:
```
files
```

Out[38]:
```
['uber-raw-data-apr14.csv',
 'uber-raw-data-aug14.csv',
 'uber-raw-data-jul14.csv',
 'uber-raw-data-jun14.csv',
 'uber-raw-data-may14.csv',
 'uber-raw-data-sep14.csv']
```

NOW, WE GOT THE REQUIRED FILES

In [39]:
```
path=r'C:\Users\Aakasha\Desktop\uber-pickups-in-new-york-city-20220626T112836Z-001\uber-pickups-in-new-york-city'
final=pd.DataFrame()
for file in files:
    current_df=pd.read_csv(path+'/'+file,encoding='utf-8')
    final=pd.concat([current_df,final])
```

In [40]:
```
final.shape
```

Out[40]:
```
(4534327, 4)
```

In [41]:
```
final.head(2)
```

Out[41]:

	Date/Time	Lat	Lon	Base
0	9/1/2014 0:01:00	40.2201	-74.0021	B02512
1	9/1/2014 0:01:00	40.7500	-74.0027	B02512

NOW, WE HAVE ANALYSE i.e., CLEAN THE DATA

1.DUPLICATE VALUES

2.MISSING VALUE

3.WRONG DATA TYPE

TO KNOW NO.OF DUPLICATES:-

In [42]:
```
final.duplicated().sum()
```

Out[42]: 82581

TO DELETE/CLEAR THE DUPLICATES:-

IF, INPLACE= TRUE, THEN DATASET CHANGE OCCURS.

IF, INPLACE= FALSE, NO CHANGE OCCURS IN THE DATASET

In [43]:
```
final.drop_duplicates(inplace=True)
```

In [44]:
```
final.shape
```

Out[44]: (4451746, 4)

WE CAN OBSERVE THE DECREASE IN THE NO.OF ROWS

NOW,

PROB:-

AT WHAT LOCATIONS OF NEW YORK CITY WE ARE GETTING RUSH

In [45]:
```
final.head()
```

Out[45]:

	Date/Time	Lat	Lon	Base
0	9/1/2014 0:01:00	40.2201	-74.0021	B02512
1	9/1/2014 0:01:00	40.7500	-74.0027	B02512
2	9/1/2014 0:03:00	40.7559	-73.9864	B02512
3	9/1/2014 0:06:00	40.7450	-73.9889	B02512
4	9/1/2014 0:11:00	40.8145	-73.9444	B02512

In [46]:
```
rush_uber=final.groupby(['Lat','Lon'],as_index=False).size()
```

In [47]:
```
rush_uber
```

Out[47]:

	Lat	Lon	size
0	39.6569	-74.2258	1
1	39.6686	-74.1607	1
2	39.7214	-74.2446	1
3	39.8416	-74.1512	1
4	39.9055	-74.0791	1
...	...	...	...
574553	41.3730	-72.9237	1
574554	41.3737	-73.7988	1
574555	41.5016	-72.8987	1
574556	41.5276	-72.7734	1
574557	42.1166	-72.0666	1

574558 rows × 3 columns

HOW TO SEE THE WORLD MAP!!

In [48]:
```
!pip install folium
```

```
Requirement already satisfied: folium in c:\python folder\anaconda aakash\lib\site-packages (0.12.1.post1)
Requirement already satisfied: jinja2>=2.9 in c:\python folder\anaconda aakash\lib\site-packages (from folium) (2.11.3)
Requirement already satisfied: requests in c:\python folder\anaconda aakash\lib\site-packages (from folium) (2.25.1)
Requirement already satisfied: branca>=0.3.0 in c:\python folder\anaconda aakash\lib\site-packages (from folium) (0.5.0)
Requirement already satisfied: numpy in c:\python folder\anaconda aakash\lib\site-packages (from folium) (1.20.1)
Requirement already satisfied: MarkupSafe>=0.23 in c:\python folder\anaconda aakash\lib\site-packages (from jinja2>=2.9->foliu
m) (1.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\python folder\anaconda aakash\lib\site-packages (from requests->foli
um) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\python folder\anaconda aakash\lib\site-packages (from requests->folium) (2.1
0)
Requirement already satisfied: certifi>=2017.4.17 in c:\python folder\anaconda aakash\lib\site-packages (from requests->folium)
(2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\python folder\anaconda aakash\lib\site-packages (from requests->folium)
(4.0.0)
```

In [49]:
```
import folium
```

In [50]:
```
basemap=folium.Map()
```

In [51]:
```
basemap
```

Out[51]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [52]:
```
from folium.plugins import HeatMap
```

In [53]:
```
HeatMap(rush_uber).add_to(basemap)
```

Out[53]: <folium.plugins.heat_map.HeatMap at 0x1c82663d880>

In [54]:
```
basemap
```

Out[54]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [55]:
```
final['Date/Time']=pd.to_datetime(final['Date/Time'],format='%m/%d/%Y %H:%M:%S')
```

In [56]:
```
final
```

Out[56]:

	Date/Time	Lat	Lon	Base
0	2014-09-01 00:01:00	40.2201	-74.0021	B02512
1	2014-09-01 00:01:00	40.7500	-74.0027	B02512
2	2014-09-01 00:03:00	40.7559	-73.9864	B02512
3	2014-09-01 00:06:00	40.7450	-73.9889	B02512
4	2014-09-01 00:11:00	40.8145	-73.9444	B02512
...	...	...	...	...
564511	2014-04-30 23:22:00	40.7640	-73.9744	B02764
564512	2014-04-30 23:26:00	40.7629	-73.9672	B02764
564513	2014-04-30 23:31:00	40.7443	-73.9889	B02764
564514	2014-04-30 23:32:00	40.6756	-73.9405	B02764
564515	2014-04-30 23:48:00	40.6880	-73.9608	B02764

4451746 rows × 4 columns

In [57]:
```
final['Date/Time'].dtype
```

Out[57]:
```
dtype('<M8[ns]')
```

In [58]:
```
final
```

Out[58]:

	Date/Time	Lat	Lon	Base
0	2014-09-01 00:01:00	40.2201	-74.0021	B02512
1	2014-09-01 00:01:00	40.7500	-74.0027	B02512
2	2014-09-01 00:03:00	40.7559	-73.9864	B02512
3	2014-09-01 00:06:00	40.7450	-73.9889	B02512
4	2014-09-01 00:11:00	40.8145	-73.9444	B02512
...	...	...	...	...
564511	2014-04-30 23:22:00	40.7640	-73.9744	B02764
564512	2014-04-30 23:26:00	40.7629	-73.9672	B02764
564513	2014-04-30 23:31:00	40.7443	-73.9889	B02764
564514	2014-04-30 23:32:00	40.6756	-73.9405	B02764
564515	2014-04-30 23:48:00	40.6880	-73.9608	B02764

4451746 rows × 4 columns

PROB:-

EXAMINE THE RUSH ON HOUR AND WEEKDAY(PAIR WISE ANALYSIS)

WE CAN DO IT BY MATRIX METHOD, D D D H | | H | | H | |

WE NEED TO EXTRACT THE MONTH, WEEKDAY

In [59]:
```
final['Date/Time']=pd.to_datetime(final['Date/Time'],format='%m/%d/%Y %H:%M:%S')
```

In [60]:
```
final['weekday']=final['Date/Time'].dt.day
final['hour']=final['Date/Time'].dt.hour
```

In [61]:
```
final.head()
```

Out[61]:

	Date/Time	Lat	Lon	Base	weekday	hour
0	2014-09-01 00:01:00	40.2201	-74.0021	B02512	1	0
1	2014-09-01 00:01:00	40.7500	-74.0027	B02512	1	0
2	2014-09-01 00:03:00	40.7559	-73.9864	B02512	1	0
3	2014-09-01 00:06:00	40.7450	-73.9889	B02512	1	0
4	2014-09-01 00:11:00	40.8145	-73.9444	B02512	1	0

In [62]:
```
final.groupby(['weekday','hour']).size()
```

Out[62]:
```
weekday  hour
1        0       3178
         1       1944
         2       1256
         3       1308
         4       1429
                 ...
31       19      4898
         20      4819
         21      5064
         22      5164
         23      3961
Length: 744, dtype: int64
```

UNSTACK() WILL CHANGE THE DATATYPE FROM SERIES TO DATAFRAMES

In [63]:
```
pivot=final.groupby(['weekday','hour']).size().unstack()
```

In [64]:
```
pivot
```

Out[64]:

hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
weekday																					
1	3178	1944	1256	1308	1429	2126	3664	5380	5292	4617	...	6933	7910	8633	9511	8604	8001	7315	7803	6268	4050
2	2435	1569	1087	1414	1876	2812	4920	6544	6310	4712	...	6904	8449	10109	11100	11123	9474	8759	8357	6998	5160
3	3354	2142	1407	1467	1550	2387	4241	5663	5386	4657	...	7226	8850	10314	10491	11239	9599	9026	8531	7142	4686
4	2897	1688	1199	1424	1696	2581	4592	6029	5704	4744	...	7158	8515	9492	10357	10259	9097	8358	8649	7706	5130
5	2733	1541	1030	1253	1617	2900	4814	6261	6469	5530	...	6955	8312	9609	10699	10170	9430	9354	9610	8853	6518
6	4537	2864	1864	1555	1551	2162	3642	4766	4942	4401	...	7235	8612	9444	9929	9263	8405	8117	8567	7852	5946
7	3645	2296	1507	1597	1763	2422	4102	5575	5376	4639	...	7276	8474	10393	11013	10573	9472	8691	8525	7194	4801
8	2830	1646	1123	1483	1889	3224	5431	7361	7357	5703	...	7240	8775	9851	10673	9687	8796	8604	8367	6795	4256
9	2657	1724	1222	1480	1871	3168	5802	7592	7519	5895	...	7877	9220	10270	11910	11449	9804	8909	8665	7499	5203
10	3296	2126	1464	1434	1591	2594	4664	6046	6158	5072	...	7612	9578	11045	11875	10934	9613	9687	9240	7766	5496
11	3036	1665	1095	1424	1842	2520	4954	6876	6871	5396	...	7503	8920	10125	10898	10361	9327	8824	8730	7771	5360
12	3227	2147	1393	1362	1757	2710	4576	6250	6231	5177	...	7743	9390	10734	11713	12216	10393	9965	10310	9992	7945
13	5408	3509	2262	1832	1705	2327	4196	5685	6060	5631	...	8200	9264	10534	11826	11450	9921	8705	8423	7363	5936
14	3748	2349	1605	1656	1756	2629	4257	5781	5520	4824	...	6963	8192	9511	10115	9553	9146	9182	8589	6891	4460
15	2497	1515	1087	1381	1862	2980	5050	6837	6729	5201	...	7633	8505	10285	11959	11728	11032	10509	9105	7153	4480
16	2547	1585	1119	1395	1818	2966	5558	7517	7495	5958	...	7597	9290	10804	11773	10855	10924	10142	10374	8094	5380
17	3155	2048	1500	1488	1897	2741	4562	6315	5882	4934	...	7472	8997	10323	11236	11089	9919	9935	9823	8362	5699
18	3390	2135	1332	1626	1892	2959	4688	6618	6451	5377	...	7534	9040	10274	10692	10338	9551	9310	9285	8015	5492
19	3217	2188	1604	1675	1810	2639	4733	6159	6014	5006	...	7374	8898	9893	10741	10429	9701	10051	10049	9090	6666
20	4475	3190	2100	1858	1618	2143	3584	4900	5083	4765	...	7462	8630	9448	10046	9272	8592	8614	8703	7787	5907
21	4294	3194	1972	1727	1926	2615	4185	5727	5529	4707	...	7064	8127	9483	9817	9291	8317	8107	8245	7362	5231
22	2787	1637	1175	1468	1934	3151	5204	6872	6850	5198	...	7337	9148	10574	10962	9884	8980	8772	8430	6784	4530
23	2546	1580	1136	1429	1957	3132	5204	6890	6436	5177	...	7575	9309	9980	10341	10823	11347	11447	10347	8637	5577
24	3200	2055	1438	1493	1798	2754	4484	6013	5913	5146	...	7083	8706	10366	10786	9772	9080	9213	8831	7480	4456
25	2405	1499	1072	1439	1943	2973	5356	7627	7078	5994	...	7298	8732	9922	10504	10673	9048	8751	9508	8522	6605
26	3810	3065	2046	1806	1730	2337	3776	5172	5071	4808	...	7269	8815	9885	10697	10867	10122	9820	10441	9486	7593
27	5196	3635	2352	2055	1723	2336	3539	4937	5053	4771	...	7519	8803	9793	9838	9228	8267	7908	8507	7720	6046
28	4123	2646	1843	1802	1883	2793	4290	5715	5671	5206	...	7341	8584	9671	9975	9132	8255	8309	7949	6411	4461
29	2678	1827	1409	1678	1948	3056	5213	6852	6695	5481	...	7630	9249	10105	11113	10411	9301	9270	9114	6992	4323
30	2401	1510	1112	1403	1841	3216	5757	7596	7611	6064	...	8396	10243	11554	12126	12561	11024	10836	10042	8275	4723
31	2174	1394	1087	919	773	997	1561	2169	2410	2525	...	4104	5099	5386	5308	5350	4898	4819	5064	5164	3961

31 rows × 24 columns

NOW, WE ACTUALLY NEED THE MAX OF THE UBER RUSH ON (HOUR,WEEKDAY)

In [65]:
```
pivot.style.background_gradient()
```

Out[65]:

hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
weekday																						
1	3178	1944	1256	1308	1429	2126	3664	5380	5292	4617	4607	4729	4930	5794	6933	7910	8633	9511	8604	8001	7315	7803
2	2435	1569	1087	1414	1876	2812	4920	6544	6310	4712	4797	4975	5188	5695	6904	8449	10109	11100	11123	9474	8759	8357
3	3354	2142	1407	1467	1550	2387	4241	5663	5386	4657	4788	5065	5384	6093	7226	8850	10314	10491	11239	9599	9026	8531
4	2897	1688	1199	1424	1696	2581	4592	6029	5704	4744	4743	4975	5193	6175	7158	8515	9492	10357	10259	9097	8358	8649
5	2733	1541	1030	1253	1617	2900	4814	6261	6469	5530	5141	5011	5047	5690	6955	8312	9609	10699	10170	9430	9354	9610
6	4537	2864	1864	1555	1551	2162	3642	4766	4942	4401	4801	5174	5426	6258	7235	8612	9444	9929	9263	8405	8117	8567
7	3645	2296	1507	1597	1763	2422	4102	5575	5376	4639	4905	5166	5364	6214	7276	8474	10393	11013	10573	9472	8691	8525
8	2830	1646	1123	1483	1889	3224	5431	7361	7357	5703	5288	5350	5483	6318	7240	8775	9851	10673	9687	8796	8604	8367
9	2657	1724	1222	1480	1871	3168	5802	7592	7519	5895	5406	5443	5496	6419	7877	9220	10270	11910	11449	9804	8909	8665
10	3296	2126	1464	1434	1591	2594	4664	6046	6158	5072	4976	5415	5506	6527	7612	9578	11045	11875	10934	9613	9687	9240

In [ ]:
```

```