Vehicle Status Change Analysis

Import Libraries and Read Data

In [1]:
```
#Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import pytz
import seaborn as sns
import plotly.express as px
```

In [2]:
```
#Load .csv file into dataframe
vsc_df = pd.read_csv("VehicleStatusChange.csv")
```

Data Dictionary

VEHICLENUMBER: the unique ID for a vehicle. This is not a unique ID for the table, there are potentially many records for each vehicle.

LONG, LAT: Longitude and Latitude. These are generated at the time of the event.

RIDEID: If the event was a user ride, there is an associated ID for that ride. This will be NULL when the event was not associated with a ride.

BATTERYPCT: The battery reading on the vehicle at the time of the event.

TIMESTAMP: The time and date of the event. Note that these timestamps are in UTC time.

AREAID: the ID for the area the event occurred in. 81 is Atlanta.

EVENT_TYPE: the type of event being recorded.

VEHICLETYPE: the type of vehicle the event corresponds to. There are two types of vehicles in Atlanta, Scooters (stand-up) and Cosmos (sit-down). Both are electric vehicles.

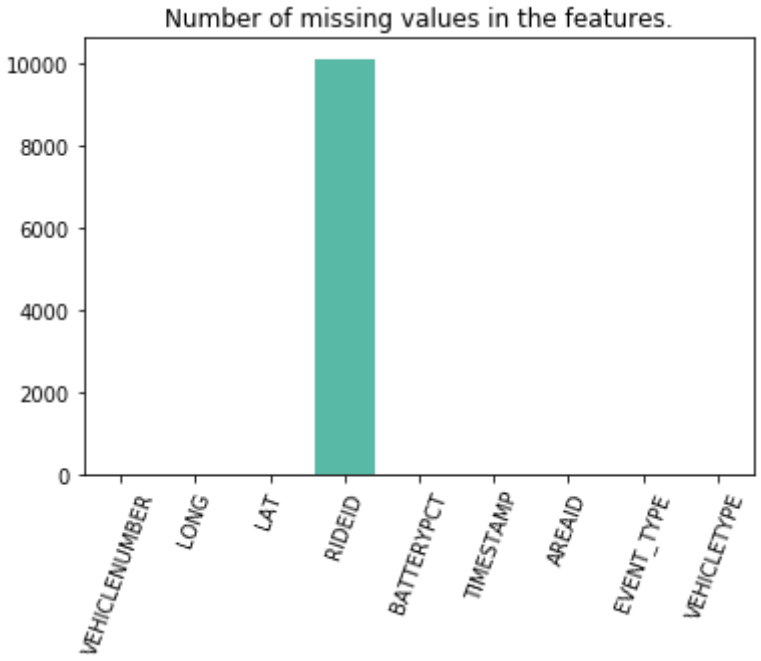
Examine Dataframe

In [49]:
```
#Information related to the dataset
vsc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50951 entries, 0 to 50950
Data columns (total 8 columns):
VEHICLENUMBER    50951 non-null int64
LONG             50948 non-null float64
LAT              50948 non-null float64
RIDEID           40839 non-null float64
BATTERYPCT       50951 non-null float64
TIMESTAMP        50951 non-null datetime64[ns]
EVENT_TYPE       50951 non-null object
VEHICLETYPE      50951 non-null object
dtypes: datetime64[ns](1), float64(4), int64(1), object(2)
memory usage: 3.1+ MB
```

In [4]:
```
#Check for Null Values
na_counts = vsc_df.isna().sum()
base_color = sns.color_palette()[0]
sns.barplot(na_counts.index.values, na_counts, color = "#48C9B0")
plt.xticks(rotation=70);
plt.title('Number of missing values in the features.')
```

Out[4]:
```
Text(0.5, 1.0, 'Number of missing values in the features.')
```



In [5]:
```
#Return first 5 rows
vsc_df.head()
```

Out[5]:

	VEHICLENUMBER	LONG	LAT	RIDEID	BATTERYPCT	TIMESTAMP	AREAID	
0	50107700	-84.368447	33.771819	2846652.0	0.91	8/31/20 23:59	81	U
1	90100018	-84.364719	33.755342	2846447.0	0.54	8/31/20 23:59	81	USE
2	50105350	-84.374735	33.787513	2846624.0	0.27	8/31/20 23:59	81	USE
3	50105350	-84.374777	33.787652	2846624.0	0.28	8/31/20 23:58	81	U
4	50109360	-84.361827	33.761190	2846611.0	0.47	8/31/20 23:56	81	U

In [6]:
```
#Number of Cosmo Rides
vsc_df[vsc_df['VEHICLETYPE'] == 'Cosmo'].count()
```

Out[6]:
```
VEHICLENUMBER    3505
LONG             3504
LAT              3504
RIDEID           2690
BATTERYPCT       3505
TIMESTAMP        3505
AREAID           3505
EVENT_TYPE       3505
VEHICLETYPE      3505
dtype: int64
```

In [7]:
```
#Number of Scooter Rides
vsc_df[vsc_df['VEHICLETYPE'] == 'Scooter'].count()
```

Out[7]:
```
VEHICLENUMBER    47446
LONG             47444
LAT              47444
RIDEID           38149
BATTERYPCT       47446
TIMESTAMP        47446
AREAID           47446
EVENT_TYPE       47446
VEHICLETYPE      47446
dtype: int64
```

In [8]:
```
#Convert Timestamp to datetime
vsc_df['TIMESTAMP'] = pd.to_datetime(vsc_df['TIMESTAMP'])
#Convert UTC to Eastern timezone for Atlanta
vsc_df['TIMESTAMP'] = vsc_df['TIMESTAMP'].dt.tz_localize('US/Eastern').d
t.tz_convert('UTC')
vsc_df['TIMESTAMP'] = vsc_df['TIMESTAMP'].apply(lambda x: datetime.repla
ce(x, tzinfo=None))
```

In [9]:
```
#Check for any anomalies in AREAID
area = vsc_df.AREAID.unique()
area
```

Out[9]:
```
array([81])
```

In [10]:
```
vsc_df = vsc_df.drop('AREAID', 1)
```

In [11]:
```
#vsc_df.to_excel('VSC.xlsx', index = False)
```

In [12]:
```
#Disseminate data into rides nd non-rides dataframe
riders_df = vsc_df[pd.notnull(vsc_df['RIDEID'])]
not_riders_df = vsc_df[pd.isnull(vsc_df['RIDEID'])]
riders_df.head()
```

Out[12]:

	VEHICLENUMBER	LONG	LAT	RIDEID	BATTERYPCT	TIMESTAMP	EVENT_T
0	50107700	-84.368447	33.771819	2846652.0	0.91	2020-09-01 03:59:00	USER_PICK
1	90100018	-84.364719	33.755342	2846447.0	0.54	2020-09-01 03:59:00	USER_DROP_
2	50105350	-84.374735	33.787513	2846624.0	0.27	2020-09-01 03:59:00	USER_DROP_
3	50105350	-84.374777	33.787652	2846624.0	0.28	2020-09-01 03:58:00	USER_PICK
4	50109360	-84.361827	33.761190	2846611.0	0.47	2020-09-01 03:56:00	USER_PICK

Rides Dataframe exploration and Analysis

In [13]:
```
#Check for datatypes
riders_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40839 entries, 0 to 50950
Data columns (total 8 columns):
VEHICLENUMBER    40839 non-null int64
LONG             40839 non-null float64
LAT              40839 non-null float64
RIDEID           40839 non-null float64
BATTERYPCT       40839 non-null float64
TIMESTAMP        40839 non-null datetime64[ns]
EVENT_TYPE       40839 non-null object
VEHICLETYPE      40839 non-null object
dtypes: datetime64[ns](1), float64(4), int64(1), object(2)
memory usage: 2.8+ MB
```

In [14]:
```
#Check for null values
riders_df.isnull().sum()
```

Out[14]:
```
VEHICLENUMBER    0
LONG             0
LAT              0
RIDEID           0
BATTERYPCT       0
TIMESTAMP        0
EVENT_TYPE       0
VEHICLETYPE      0
dtype: int64
```

In [15]:
```
#Check for types in Event
arr1 = riders_df.EVENT_TYPE.unique()
arr1
```

Out[15]:
```
array(['USER_PICK_UP', 'USER_DROP_OFF'], dtype=object)
```

In [16]:
```
#Check for types in Vehicle
arr2 = riders_df.VEHICLETYPE.unique()
arr2
```

Out[16]:
```
array(['Scooter', 'Cosmo'], dtype=object)
```

In [17]:
```
#Convert Ride Id's to int from float
riders_df = riders_df.astype({"RIDEID":'int'})
```

In [18]:
```
#Map/Join PickUp with Drop Off and rename the columns
pickup_df = riders_df.loc[riders_df['EVENT_TYPE'] == 'USER_PICK_UP']
dropoff_df = riders_df.loc[riders_df['EVENT_TYPE'] == 'USER_DROP_OFF']
merge_df = pickup_df.merge(dropoff_df, on='RIDEID', how='inner')
merge_df = merge_df.drop(['EVENT_TYPE_x','VEHICLENUMBER_y','EVENT_TYPE_
y','VEHICLETYPE_y'], 1)
merge_df.rename(columns={'VEHICLENUMBER_x': 'VehicleID', 'LONG_x': 'Pick
UpLong','LAT_x': 'PickUpLat', 'LONG_y': 'DropOffLong','LAT_y': 'DropOffL
at','BATTERYPCT_x': 'StartBatPct','BATTERYPCT_y': 'EndBatPct','TIMESTAMP
_x': 'StartTime','TIMESTAMP_y': 'EndTime','VEHICLETYPE_x': 'VehicleType'
}, inplace=True)
merge_df = merge_df[["RIDEID","VehicleID","StartTime","EndTime","StartBa
tPct","EndBatPct","PickUpLong","PickUpLat","DropOffLong","DropOffLat","V
ehicleType"]]
merge_df.head()
```

Out[18]:

	RIDEID	VehicleID	StartTime	EndTime	StartBatPct	EndBatPct	PickUpLong	PickUpLat	Dro
0	2846624	50105350	2020-09-01 03:58:00	2020-09-01 03:59:00	0.28	0.27	-84.374777	33.787652	-8
1	2846550	20100209	2020-09-01 03:51:00	2020-09-01 03:53:00	1.00	1.00	-84.405345	33.747487	-8
2	2846525	20100209	2020-09-01 03:49:00	2020-09-01 03:51:00	1.00	1.00	-84.405411	33.747375	-8
3	2846521	50107698	2020-09-01 03:49:00	2020-09-01 03:52:00	0.83	0.83	-84.383259	33.772271	-8
4	2846517	20100040	2020-09-01 03:49:00	2020-09-01 03:51:00	0.97	0.94	-84.358072	33.763010	-8

In [19]:
```
#Compute Trip Duration based on Pickup and Drop Off timestamp
merge_df['Trip Duration'] = merge_df['EndTime'] - merge_df['StartTime']
```

In [20]:
```
#Check for distribution of timestamp
print(merge_df['Trip Duration'].describe())
```

```
count                      20329
mean      0 days 00:31:16.294948
std       0 days 01:36:25.778776
min              0 days 00:00:00
25%              0 days 00:05:00
50%              0 days 00:12:00
75%              0 days 00:30:00
max              3 days 04:02:00
Name: Trip Duration, dtype: object
```

In [21]:
```
#Add columns with date, hour and day of week from timestmp
merge_df['day'] = merge_df['StartTime'].dt.day
merge_df['hour'] = merge_df['StartTime'].dt.hour
merge_df['weekday'] = merge_df['StartTime'].dt.dayofweek
```

In [22]:
```
#Trip durations exceeding 24 hours
merge_df[merge_df['Trip Duration']> "24:00:00"]
```

Out[22]:

	RIDEID	VehicleID	StartTime	EndTime	StartBatPct	EndBatPct	PickUpLong	PickUpLat
4469	2807863	90100096	2020-08-28 04:57:00	2020-08-29 05:27:00	0.80	0.18	-84.424683	33.716826
4477	2807756	90100027	2020-08-28 04:46:00	2020-08-29 05:27:00	0.68	0.42	-84.373180	33.766224
17742	2646003	50107424	2020-08-05 02:55:00	2020-08-06 07:29:00	0.76	0.42	-84.371297	33.788358
17860	2645411	20100253	2020-08-05 01:14:00	2020-08-06 07:29:00	0.39	0.83	-84.388024	33.761258
17902	2645236	50107015	2020-08-05 00:31:00	2020-08-06 07:29:00	0.56	0.00	-84.392020	33.759960
18227	2642288	20100046	2020-08-04 05:00:00	2020-08-05 06:45:00	0.38	0.00	-84.349531	33.764875
18264	2642027	50105390	2020-08-04 04:33:00	2020-08-05 22:29:00	0.81	0.48	-84.406150	33.745563
18286	2641816	50104531	2020-08-04 04:11:00	2020-08-06 07:29:00	0.65	0.59	-84.390866	33.751543
18296	2641735	50105308	2020-08-04 04:04:00	2020-08-05 09:12:00	0.91	1.00	-84.412179	33.768445
18309	2641646	20100320	2020-08-04 03:55:00	2020-08-05 09:02:00	0.90	1.00	-84.413598	33.755470
18379	2641242	50106374	2020-08-04 02:58:00	2020-08-06 07:29:00	0.32	0.17	-84.393450	33.805129
18433	2640981	20100261	2020-08-04 02:12:00	2020-08-05 09:33:00	0.88	1.00	-84.387063	33.760694
18478	2640691	50107657	2020-08-04 01:14:00	2020-08-05 09:12:00	0.81	1.00	-84.353457	33.757741
18508	2640424	50106452	2020-08-04 00:24:00	2020-08-05 04:58:00	1.00	0.34	-84.386006	33.780715
18611	2639602	50106226	2020-08-03 20:50:00	2020-08-06 07:29:00	0.88	0.24	-84.392028	33.763253
18886	2636556	20100249	2020-08-03 03:27:00	2020-08-06 07:29:00	0.81	1.00	-84.366209	33.768338
19686	2629643	50106374	2020-08-02 03:02:00	2020-08-03 03:29:00	0.58	0.33	-84.387521	33.788989

	RIDEID	VehicleID	StartTime	EndTime	StartBatPct	EndBatPct	PickUpLong	PickUpLat
19751	2629338	50108711	2020-08-02 02:22:00	2020-08-03 03:28:00	0.89	0.37	-84.412412	33.763801
19781	2629140	50105061	2020-08-02 01:55:00	2020-08-03 03:28:00	0.91	0.00	-84.390980	33.755790

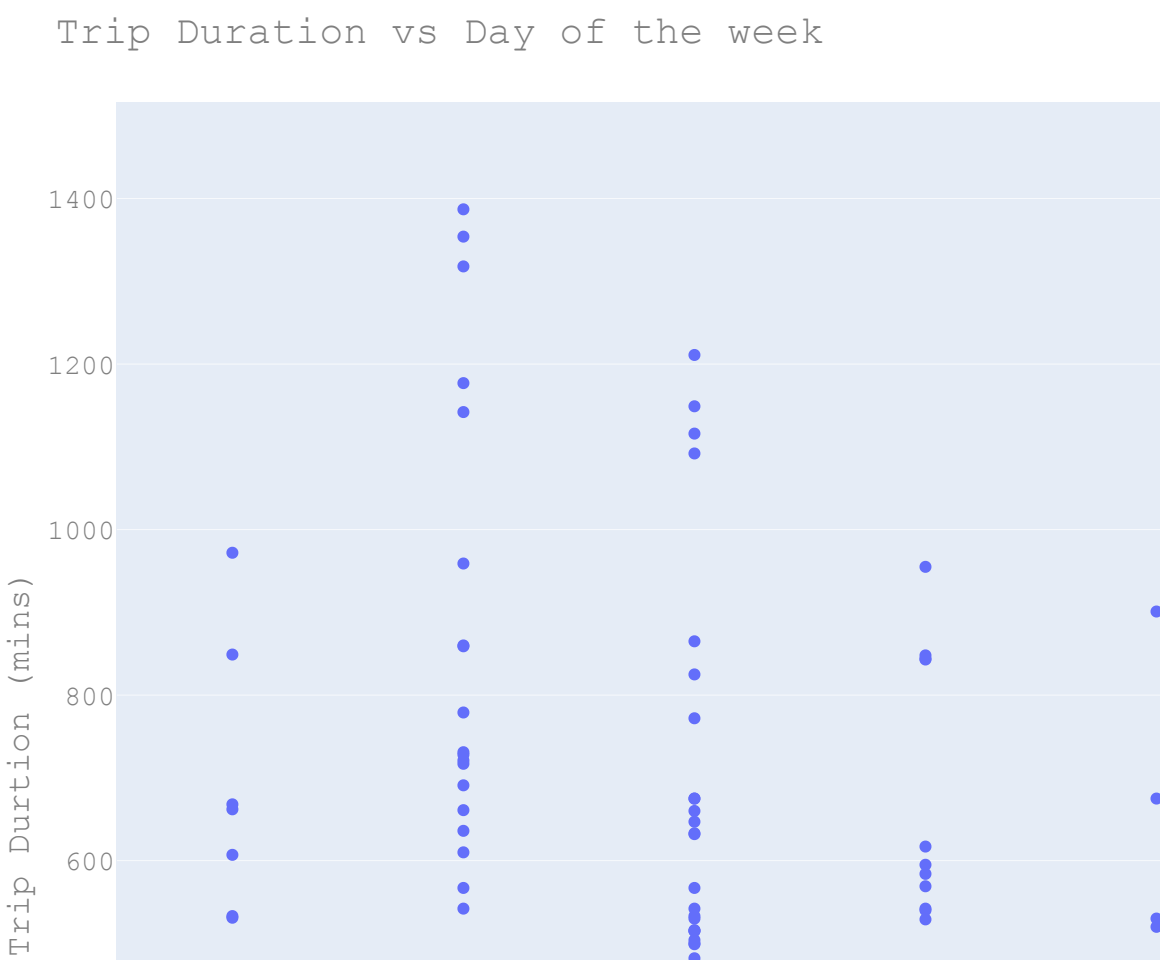
Since the trip durations of 19 e-vehicles are greater than 1 day which is an anamoly as the battery won't last that long.

In [23]:
```
#Add trip durtion in mins from timestamp
merge_df['Trip Duration mins'] = merge_df['Trip Duration'].apply(lambda
x: x/np.timedelta64(1,'m'))
```

In [24]:
```
#Final dataframe with removed anomolies
riders_result = merge_df[merge_df['Trip Duration mins'] < 1440]
```

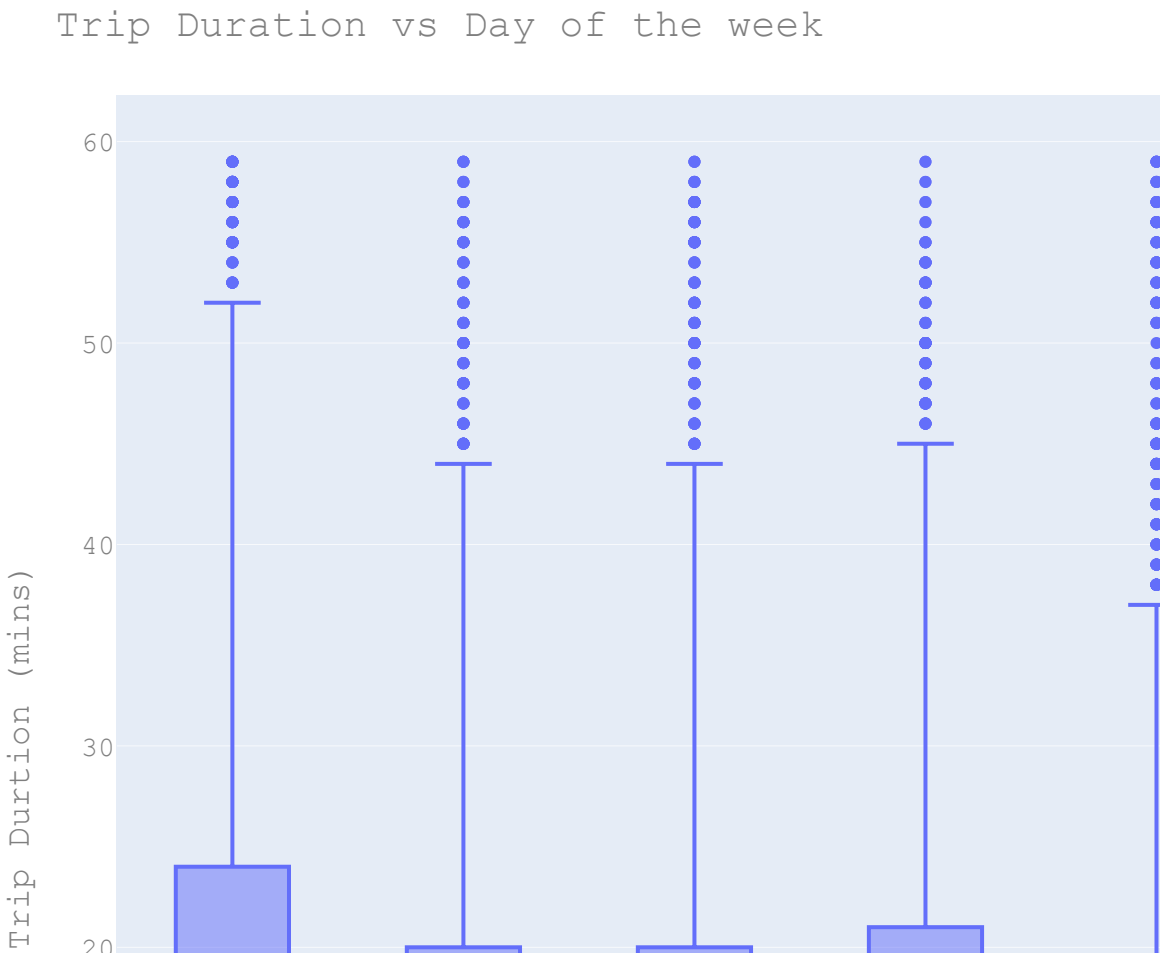
In [25]:

```
#Box plot of Trip duraation time (Most of the trips aare within 100 min
s)
fig = px.box(riders_result, x='weekday', y='Trip Duration mins')
fig.update_layout(
    height=800,
    title_text='Trip Duration vs Day of the week',
    xaxis_title='Day of Week',
    yaxis_title='Trip Durtion (mins)',
    font=dict(
        family="Courier New, monospace",
        size=14,
        color="#7f7f7f"),
    xaxis = dict(
        tickmode = 'array',
        tickvals = [0, 1, 2, 3, 4, 5, 6],
        ticktext = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']
    ))
fig.show()
```


Trip Duration vs Day of the week
Trip Durtion (mins)
1400
1200
1000
800
600

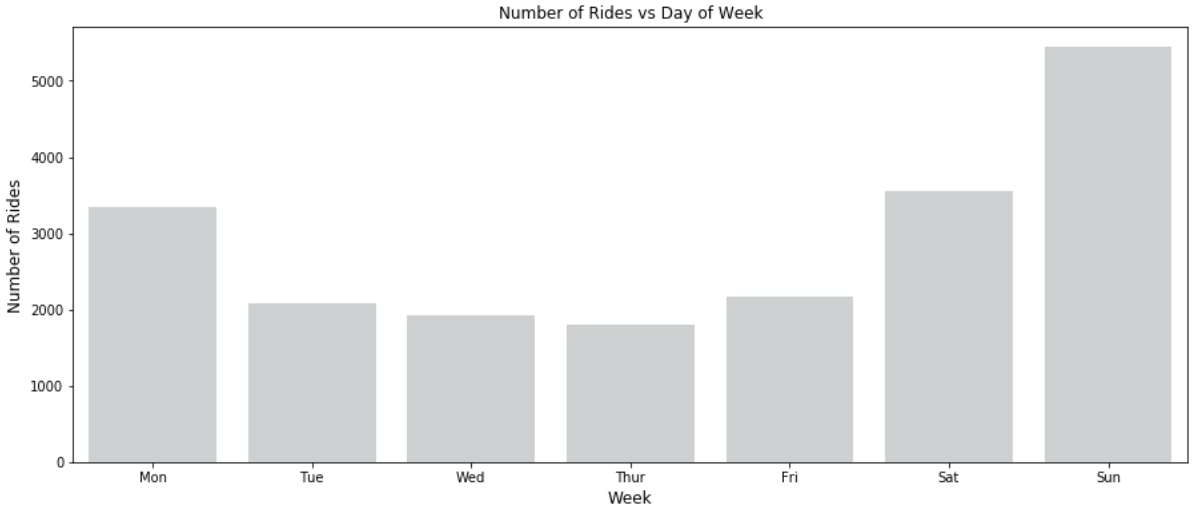
In [26]:

```
#Box plot of trip duration vs daay of the week thresholding at 60 mins
df1 = merge_df[merge_df['Trip Duration mins'] < 60]
fig = px.box(df1, x='weekday', y='Trip Duration mins')
fig.update_layout(
    height=800,
    title_text='Trip Duration vs Day of the week',
    xaxis_title='Day of Week',
    yaxis_title='Trip Durtion (mins)',
    font=dict(
        family="Courier New, monospace",
        size=14,
        color="#7f7f7f"),
    xaxis = dict(
        tickmode = 'array',
        tickvals = [0, 1, 2, 3, 4, 5, 6],
        ticktext = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']
    ))
fig.show()
```

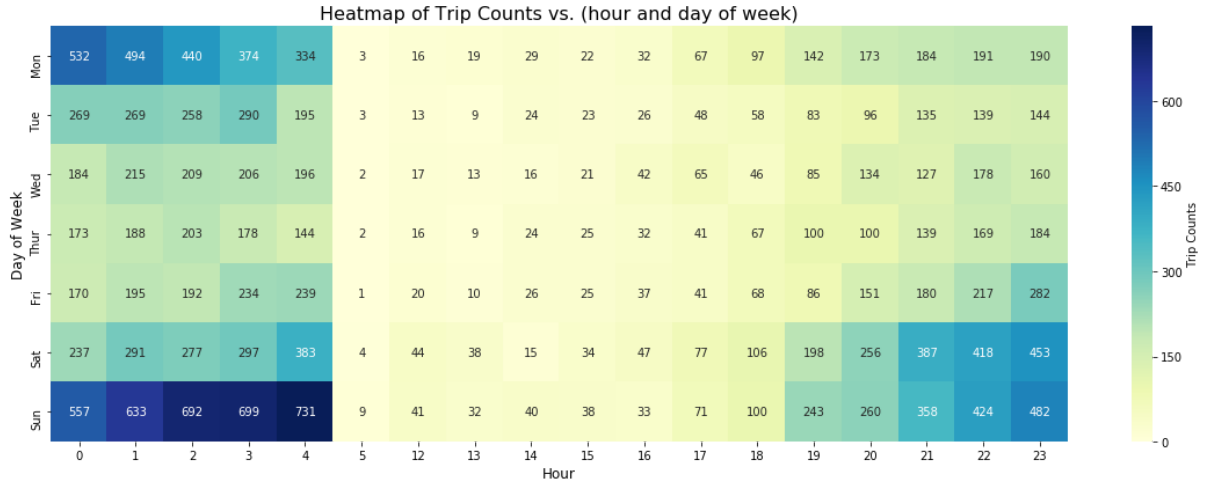

Trip Duration vs Day of the week
Trip Durtion (mins)
60
50
40
30
20

In [27]:
```
#Countplot of Number of Rides vs Day of Week
fig, ax = plt.subplots(figsize=(15,6))
sns.countplot(data=riders_result, x='weekday', color="#CCD1D1")
ax.set_title('Number of Rides vs Day of Week', fontsize=16)
ax.set_xlabel('Week', fontsize=12)
ax.set_ylabel('Number of Rides', fontsize=12)
plt.xticks(np.arange(7), ('Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Su
n'))
plt.title('Number of Rides vs Day of Week')
```

Out[27]:
```
Text(0.5, 1.0, 'Number of Rides vs Day of Week')
```



In [28]:
```
# heatmap of trip count vs. (hour and day of week)
fig, ax = plt.subplots(figsize=(16, 6))
tmp = riders_result[['weekday','hour', 'Trip Duration mins']].groupby([
'hour','weekday']).count().reset_index()
pivots = tmp.pivot('weekday','hour','Trip Duration mins')
sns.heatmap(pivots, cbar_kws={'label': 'Trip Counts'},fmt='d', annot=Tru
e, cmap="YlGnBu",annot_kws={"size": 10})
ax.set_title('Heatmap of Trip Counts vs. (hour and day of week)', fontsi
ze=16)
ax.set_xlabel('Hour', fontsize=12)
ax.set_ylabel('Day of Week', fontsize=12)
ax.set_yticklabels(('Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun'), f
ontsize=10)
plt.tight_layout()
plt.show()
```

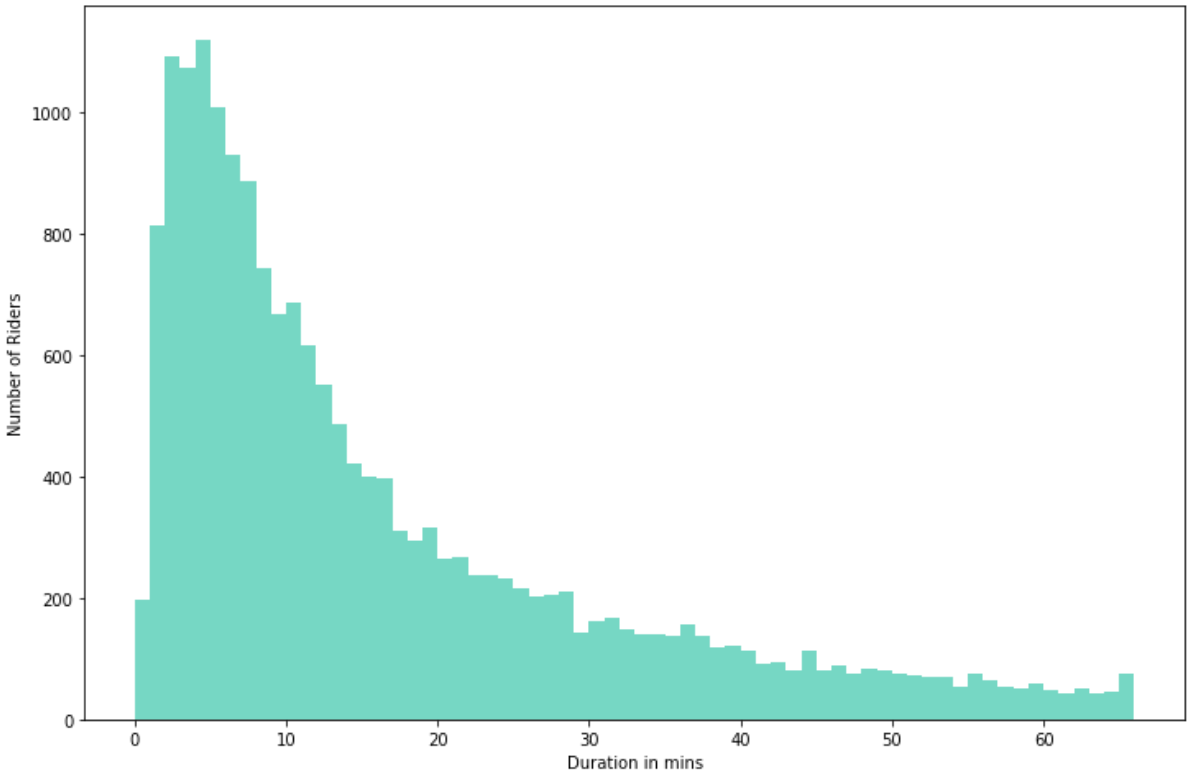


In [29]:
```
#description of trip duration in mins
riders_result['Trip Duration mins'].describe()
```

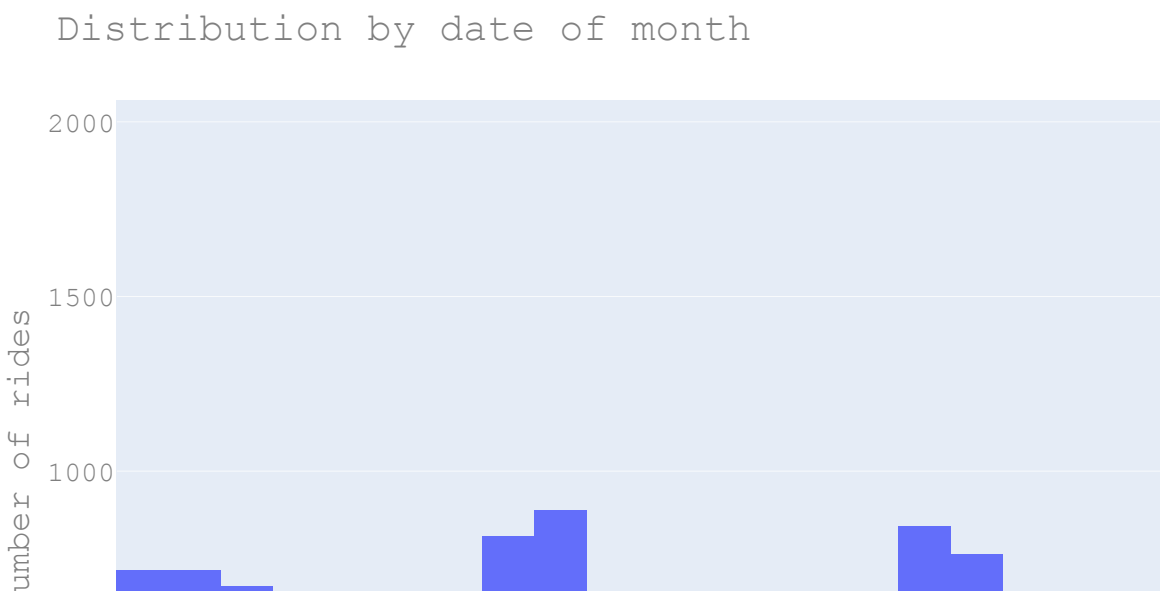
Out[29]:
```
count    20310.000000
mean        29.320384
std         67.690911
min          0.000000
25%          5.000000
50%         12.000000
75%         30.000000
max       1436.000000
Name: Trip Duration mins, dtype: float64
```

Plot it according to the standard deviation with the help of numpy and distribute it by binsizes.

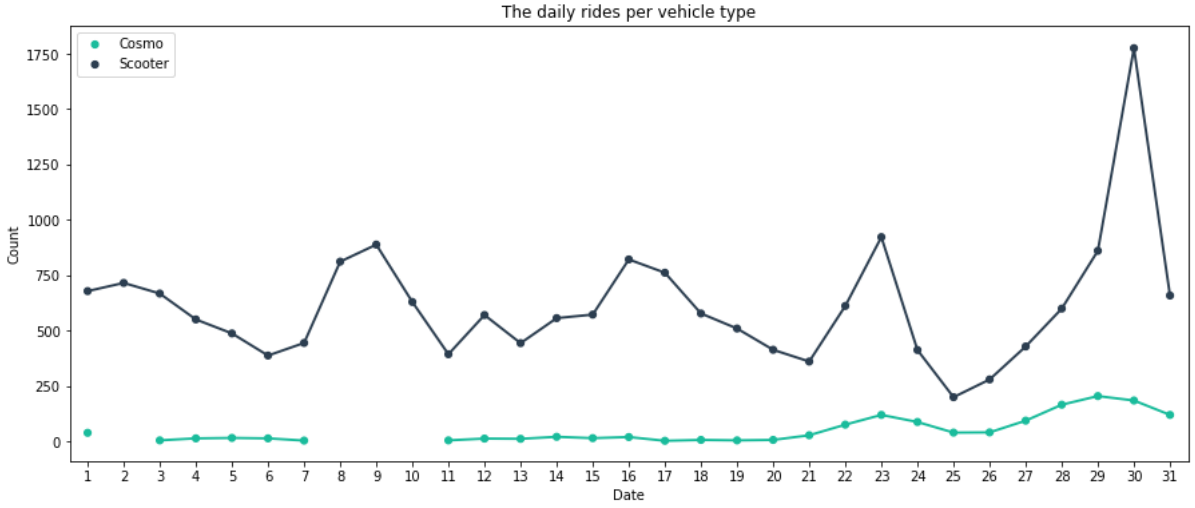
In [30]:
```
#Creating bins to plot the distribution of trip duration
binsize = 1
bins = np.arange(0, 67, binsize)
plt.figure(figsize=[12, 8])
plt.hist(data = riders_result, x = 'Trip Duration mins', bins = bins, co
lor="#76D7C4")
plt.xlabel('Duration in mins')
plt.ylabel('Number of Riders')
plt.show()
```



In [31]:
```
#Distribution by date of month
df = px.data.tips()
fig = px.histogram(df, x=riders_result['day'])
fig.update_layout(
    height=500,
    title_text='Distribution by date of month',
    xaxis_title='Date',
    yaxis_title='Number of rides',
    font=dict(
        family="Courier New, monospace",
        size=14,
        color="#7f7f7f"))
fig.show()
```

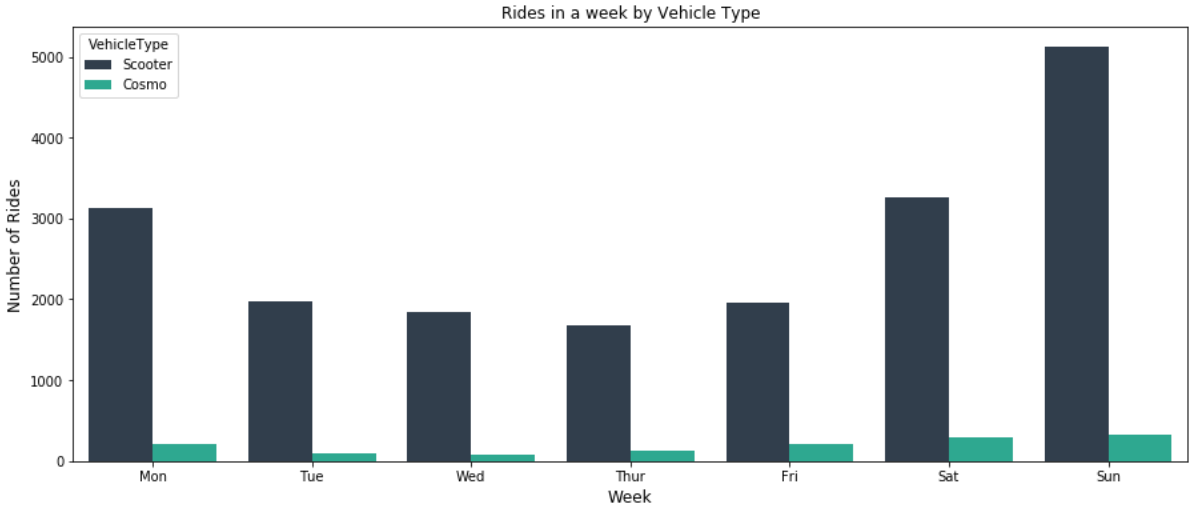


In [32]:
```
#Daily rides vs Scooter/Cosmos
user_type_count = riders_result.groupby(["day", "VehicleType"]).size().reset_index()
plt.figure(figsize=(15,6))
color = {'Scooter':'#2C3E50', 'Cosmo':'#1ABC9C'}
axis = sns.pointplot(x="day", y=0, hue="VehicleType", palette=color, scale=.7, data=user_type_count)
plt.title('The daily rides per vehicle type')
plt.xlabel('Date')
plt.ylabel('Count')
plt.legend();
```



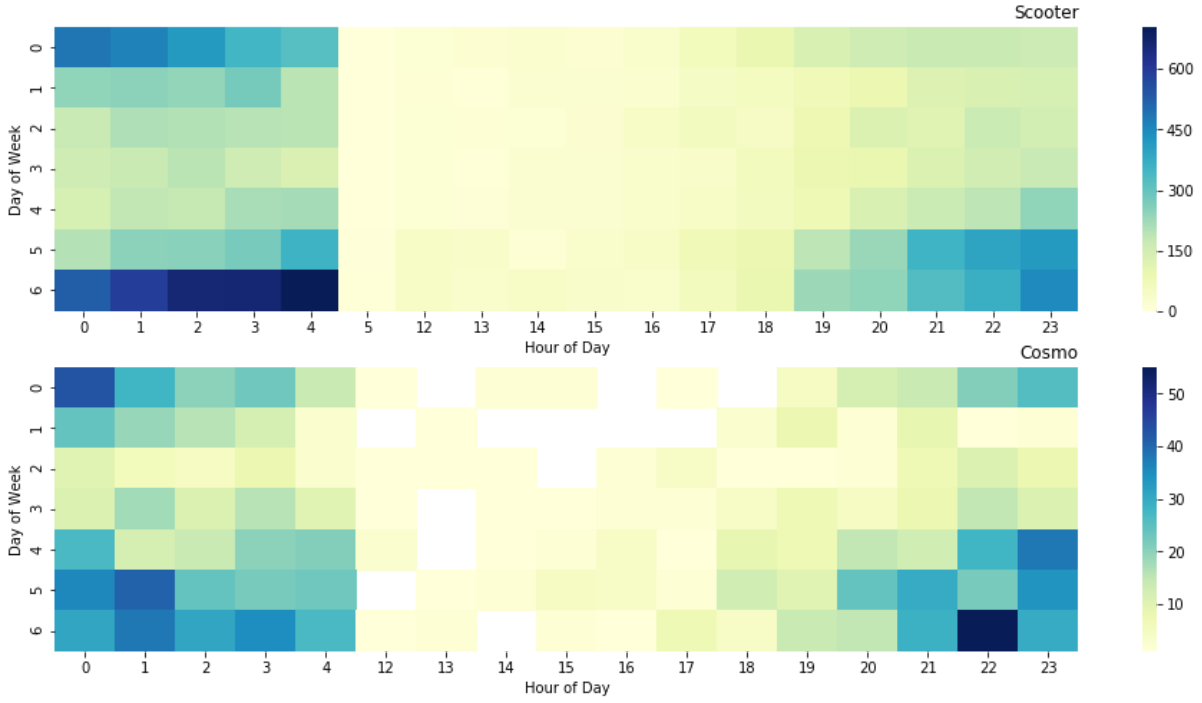
In [33]:
```
#Day of week rides by Scooter/Cosmos
fig, ax = plt.subplots(figsize=(15,6))
sns.countplot(data=riders_result, x='weekday',palette={'Scooter':'#2C3E50', 'Cosmo':'#1ABC9C'}, hue='VehicleType')
ax.set_title('Day of Week vs Vehicle Type', fontsize=20)
ax.set_xlabel('Week', fontsize=12)
ax.set_ylabel('Number of Rides', fontsize=12)
plt.xticks(np.arange(7), ('Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun'))
plt.title('Rides in a week by Vehicle Type')
```

Out[33]:
```
Text(0.5, 1.0, 'Rides in a week by Vehicle Type')
```

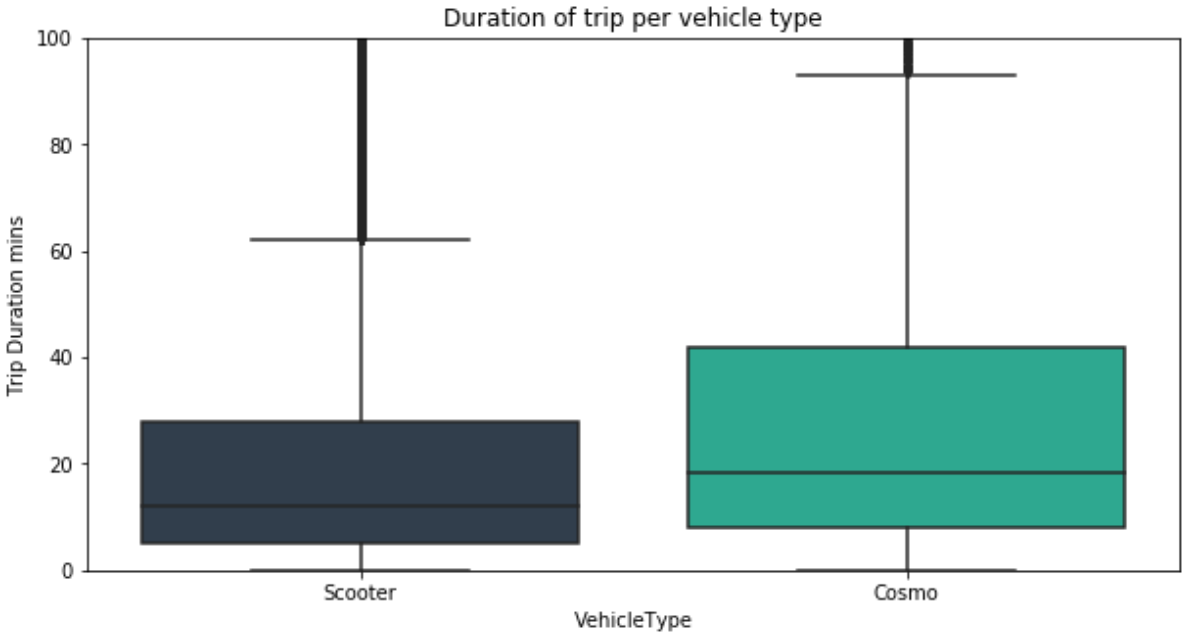


In [34]:
```
plt.suptitle('Hourly Usage during Weekdays for Customers and Subscriber
s')
#fig, ax = plt.subplots(figsize=(5,5))

plt.figure(figsize = (16,8))
plt.subplot(2, 1, 1)
customers = riders_result.query('VehicleType == "Scooter"')
ct_counts = customers.groupby(['weekday', 'hour']).size()
ct_counts = ct_counts.reset_index(name='count')
ct_counts = ct_counts.pivot(index='weekday', columns='hour', values='cou
nt')
sns.heatmap(ct_counts, cmap='YlGnBu');
plt.title('Scooter', loc='right');
plt.xlabel('Hour of Day');
plt.ylabel('Day of Week');

plt.subplot(2, 1, 2)
subscribers = riders_result.query('VehicleType == "Cosmo"')
st_counts = subscribers.groupby(['weekday', 'hour']).size()
st_counts = st_counts.reset_index(name='count')
st_counts = st_counts.pivot(index='weekday', columns='hour', values='cou
nt')
sns.heatmap(st_counts, cmap='YlGnBu');
plt.title('Cosmo', loc='right');
plt.xlabel('Hour of Day');
plt.ylabel('Day of Week');
```

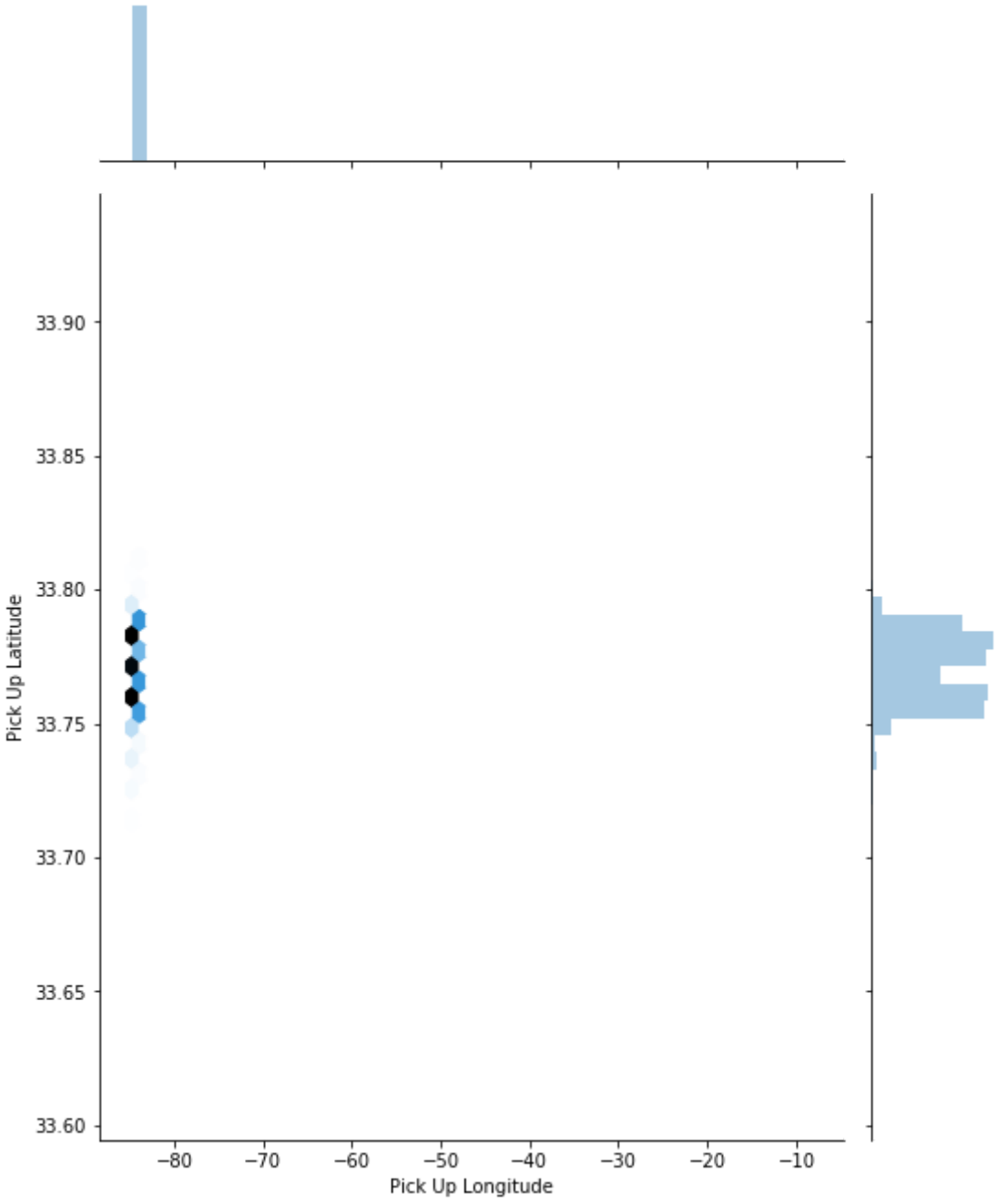
```
<Figure size 432x288 with 0 Axes>
```



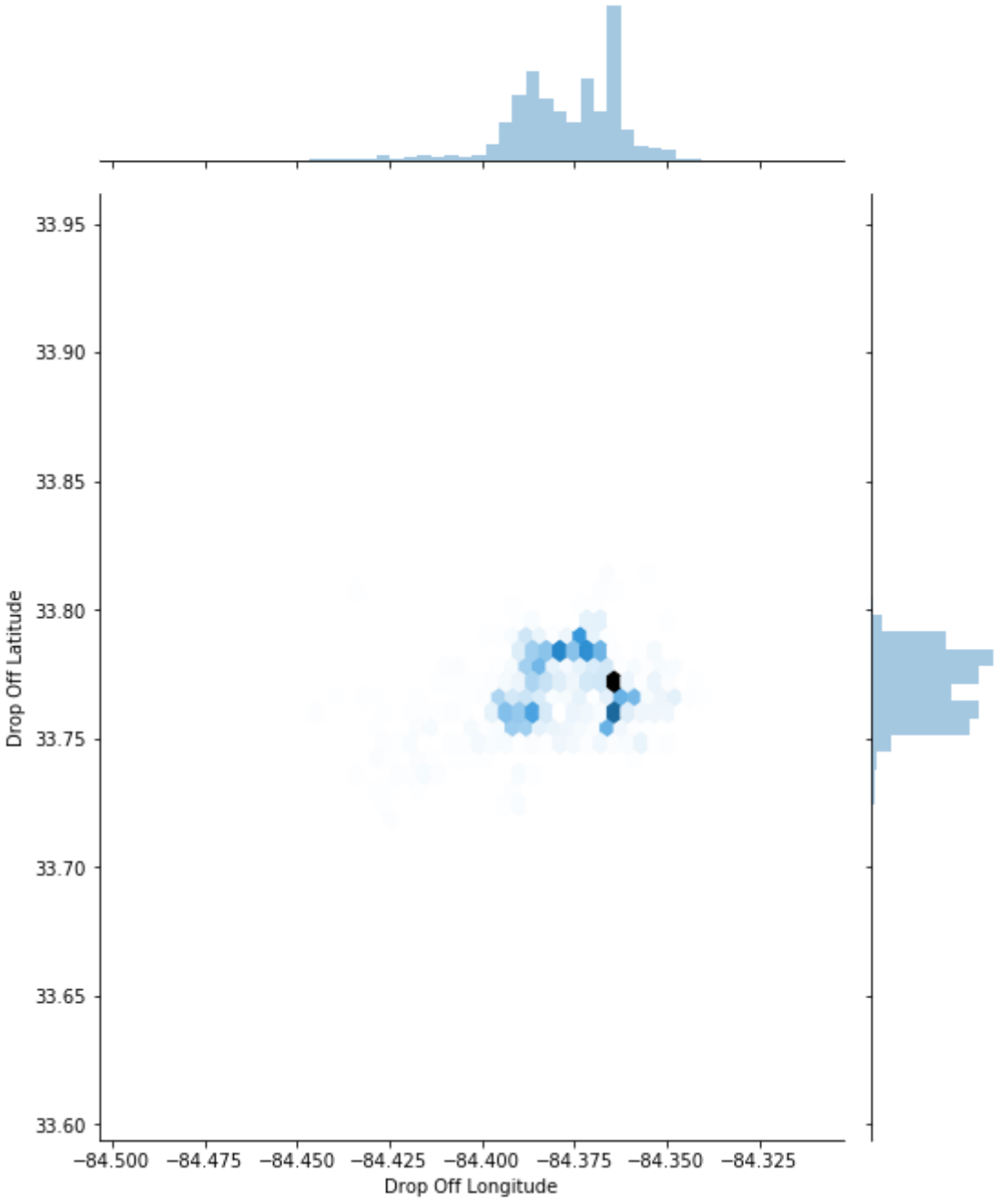
In [35]:
```
#Boxplot to check trip duration by Scooter/Cosmos
plt.figure(figsize = [10, 5])
ax1 = sns.boxplot(data = riders_result, x = 'VehicleType', y = 'Trip Dur
ation mins', palette = {'Scooter':'#2C3E50', 'Cosmo':'#1ABC9C'})
plt.ylim(0,100)
plt.title('Duration of trip per vehicle type');
```



In [36]:
```
g = sns.jointplot('PickUpLong', 'PickUpLat', data=riders_result, kind='hex')
g.set_axis_labels('Pick Up Longitude', 'Pick Up Latitude')
g.fig.set_figwidth(8)
g.fig.set_figheight(10)
plt.show()
```



In [37]:
```
g = sns.jointplot('DropOffLong', 'DropOffLat', data=riders_result, kind=
'hex')
g.set_axis_labels('Drop Off Longitude', 'Drop Off Latitude')
g.fig.set_figwidth(8)
g.fig.set_figheight(10)
plt.show()
```



Non - Rides data exploration

In [38]:
```
not_riders_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10112 entries, 43 to 50885
Data columns (total 8 columns):
VEHICLENUMBER    10112 non-null int64
LONG             10109 non-null float64
LAT              10109 non-null float64
RIDEID           0 non-null float64
BATTERYPCT       10112 non-null float64
TIMESTAMP        10112 non-null datetime64[ns]
EVENT_TYPE       10112 non-null object
VEHICLETYPE      10112 non-null object
dtypes: datetime64[ns](1), float64(4), int64(1), object(2)
memory usage: 711.0+ KB
```

In [39]:
```
not_riders_df.isnull().sum()
```

Out[39]:
```
VEHICLENUMBER        0
LONG                 3
LAT                  3
RIDEID           10112
BATTERYPCT           0
TIMESTAMP            0
EVENT_TYPE           0
VEHICLETYPE          0
dtype: int64
```

In [40]:
```
arr3 = not_riders_df.EVENT_TYPE.unique()
arr3
```

Out[40]:
```
array(['REBALANCE_DROP_OFF', 'REBALANCE_PICK_UP', 'MAINTENANCE',
       'LOW_BATTERY', 'MAINTENANCE_PICK_UP', 'MAINTENANCE_DROP_OFF',
       'SERVICE_START', 'START_SWAP_BATTERY', 'SERVICE_END'], dtype=obj
ect)
```

In [41]:
```
arr4 = not_riders_df.VEHICLETYPE.unique()
arr4
```

Out[41]:
```
array(['Scooter', 'Cosmo'], dtype=object)
```

In [42]:
```
not_riders_df = not_riders_df.drop('RIDEID', axis=1)
not_riders_df.head()
```

Out[42]:

	VEHICLENUMBER	LONG	LAT	BATTERYPCT	TIMESTAMP	EVENT_TYPE
43	50107387	-84.364909	33.757648	0.68	2020-09-01 03:45:00	REBALANCE_DROP_OFF
44	50107164	-84.364909	33.757648	0.87	2020-09-01 03:45:00	REBALANCE_DROP_OFF
45	50109489	-84.364873	33.757535	1.00	2020-09-01 03:45:00	REBALANCE_PICK_UP
46	20100280	-84.364873	33.757535	0.52	2020-09-01 03:45:00	REBALANCE_PICK_UP
49	50109349	-84.364933	33.757442	0.76	2020-09-01 03:44:00	REBALANCE_PICK_UP

In [43]:
```
len(not_riders_df[not_riders_df.duplicated() == True])
```

Out[43]:
```
191
```

In [44]:
```
duplicateRowsDF = not_riders_df[not_riders_df.duplicated(keep=False)]
duplicateRowsDF.EVENT_TYPE.unique()
```

Out[44]:
```
array(['REBALANCE_PICK_UP', 'REBALANCE_DROP_OFF', 'MAINTENANCE_DROP_OF
F'],
      dtype=object)
```

In [45]:
```
not_riders_df = not_riders_df.drop_duplicates(subset=['VEHICLENUMBER',
'LONG', 'LAT', 'BATTERYPCT', 'TIMESTAMP','EVENT_TYPE', 'VEHICLETYPE'], k
eep='first')
```

In [46]:
```
not_riders_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9921 entries, 43 to 50885
Data columns (total 7 columns):
VEHICLENUMBER    9921 non-null int64
LONG             9918 non-null float64
LAT              9918 non-null float64
BATTERYPCT       9921 non-null float64
TIMESTAMP        9921 non-null datetime64[ns]
EVENT_TYPE       9921 non-null object
VEHICLETYPE      9921 non-null object
dtypes: datetime64[ns](1), float64(3), int64(1), object(2)
memory usage: 620.1+ KB
```

In [47]:
```
not_riders_df['day'] = not_riders_df['TIMESTAMP'].dt.day
not_riders_df['hour'] = not_riders_df['TIMESTAMP'].dt.hour
not_riders_df['weekday'] = not_riders_df['TIMESTAMP'].dt.dayofweek
```

In [48]:
```
#Data for Tableau analysis
#pickup_df.to_excel('pickup.xlsx', index = False)
#dropoff_df.to_excel('dropoff.xlsx', index = False)
#riders_result.to_excel('riders.xlsx', index = False)
#not_riders_df.to_excel('not_riders.xlsx', index = False)
```