Analysis of Tax_trip in NYC 2018

Questions are answered:

1, Is there a correlation between trip distance and fare amounts?

2, What are the busiest hours, days and months for tax trips?

3, What is the busiest location by hourly?

4, What are the most common pickup and dropoff locations?

5, Over all of the locations, how the passengers distribute.

In [1]:
```
import statsmodels.api as sm
import os
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv(r"C:\Users\jianx\OneDrive\Documents\SQL courses\taxi data\taxi_trip_data.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	rate_code	store_and_fwd_flag	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	imp_s
0	2	3/29/2018 13:37	3/29/2018 14:17	1	18.15	3	N	1	70.0	0.0	0.0	16.16	10.50	
1	2	3/29/2018 13:37	3/29/2018 14:15	1	4.59	1	N	1	25.0	0.0	0.5	5.16	0.00	
2	2	3/29/2018 13:26	3/29/2018 13:28	1	0.30	1	N	1	3.0	0.0	0.5	0.76	0.00	
3	2	3/29/2018 13:07	3/29/2018 14:03	2	16.97	1	N	1	49.5	0.0	0.5	5.61	5.76	
4	2	3/29/2018 14:19	3/29/2018 15:19	5	14.45	1	N	1	45.5	0.0	0.5	10.41	5.76	

In [5]:
```
df.shape
```

Out[5]:
```
(1048575, 17)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 17 columns):
 #   Column               Non-Null Count    Dtype  
---  ------               --------------    -----  
 0   vendor_id            1048575 non-null  int64  
 1   pickup_datetime      1048575 non-null  object 
 2   dropoff_datetime     1048575 non-null  object 
 3   passenger_count      1048575 non-null  int64  
 4   trip_distance        1048575 non-null  float64
 5   rate_code            1048575 non-null  int64  
 6   store_and_fwd_flag   1048575 non-null  object 
 7   payment_type         1048575 non-null  int64  
 8   fare_amount          1048575 non-null  float64
 9   extra                1048575 non-null  float64
 10  mta_tax              1048575 non-null  float64
 11  tip_amount           1048575 non-null  float64
 12  tolls_amount         1048575 non-null  float64
 13  imp_surcharge        1048575 non-null  float64
 14  total_amount         1048575 non-null  float64
 15  pickup_location_id   1048575 non-null  int64  
 16  dropoff_location_id  1048575 non-null  int64  
dtypes: float64(8), int64(6), object(3)
memory usage: 136.0+ MB
```

In [7]:
```
df=df.drop_duplicates()
```

In [8]:
```
df
```

Out[8]:

	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	rate_code	store_and_fwd_flag	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount
0	2	3/29/2018 13:37	3/29/2018 14:17	1	18.15	3	N	1	70.0	0.0	0.0	16.16	10.50
1	2	3/29/2018 13:37	3/29/2018 14:15	1	4.59	1	N	1	25.0	0.0	0.5	5.16	0.00
2	2	3/29/2018 13:26	3/29/2018 13:28	1	0.30	1	N	1	3.0	0.0	0.5	0.76	0.00
3	2	3/29/2018 13:07	3/29/2018 14:03	2	16.97	1	N	1	49.5	0.0	0.5	5.61	5.76
4	2	3/29/2018 14:19	3/29/2018 15:19	5	14.45	1	N	1	45.5	0.0	0.5	10.41	5.76
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048570	2	5/11/2018 21:06	5/11/2018 21:40	1	13.55	1	N	1	41.0	0.5	0.5	9.61	5.76
1048571	2	5/11/2018 21:59	5/11/2018 22:25	1	14.79	1	N	1	40.5	0.5	0.5	9.51	5.76
1048572	2	5/11/2018 23:03	5/11/2018 23:32	1	5.77	1	N	1	24.0	0.5	0.5	6.32	0.00
1048573	2	5/11/2018 23:06	5/11/2018 23:33	5	9.30	1	N	1	30.0	0.5	0.5	7.41	5.76
1048574	2	5/11/2018 23:02	5/11/2018 23:31	1	6.04	1	N	1	23.0	0.5	0.5	4.86	0.00

1038957 rows × 17 columns

In [9]:
```
df.vendor_id.unique()
```

Out[9]:
```
array([2, 1, 4], dtype=int64)
```

In [10]:
```
df.trip_distance.mean()
```

Out[10]: 8.846010951367573

In [11]:
```
df.trip_distance.min()
```

Out[11]: 0.0

In [12]:
```
df.trip_distance.max()
```

Out[12]: 7655.76

In [13]:
```
df.fare_amount.max()
```

Out[13]: 19269.65

Q1: Is there a correlation between trip distance and fare amounts?

Conclusion: the trip distance has a moderate realation with fare amounts.(the correlation is at 0.525 and R-square is at 0.276)

In [14]:
```
correlation = df['trip_distance'].corr(df['fare_amount'])
print(f"Correlation between trip_distance and fare_amount: {correlation}")
```

```
Correlation between trip_distance and fare_amount: 0.5257613791641136
```

In [15]:
```
# Create a linear regression model
X = df['trip_distance']
y = df['fare_amount']
X = sm.add_constant(X)  # Add a constant for the intercept term
model = sm.OLS(y, X).fit()

# Get the R-squared value
r_squared = model.rsquared

print(f"R-squared value: {r_squared}")
```

```
R-squared value: 0.2764250278205518
```

Q2: What are the busiest hours, days and months for tax trips?

The busiest hours for pickup passengers from 21:00 to 22:00PM

The passenger visiting seems to be week-cycle pattern

The highest number of passengers visting is on March

In [17]:
```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [18]:
```
df['dropoff_datetime']=pd.to_datetime(df['dropoff_datetime'])
```

In [19]:
```
df['pickup_hour'] =df['pickup_datetime'].dt.hour
```

In [20]:
```
df['dropoff_hour']=df['dropoff_datetime'].dt.hour
```

In [22]:
```
df.head()
```

Out[22]:

	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	trip_distance	rate_code	store_and_fwd_flag	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	imp_s
0	2	2018-03-29 13:37:00	2018-03-29 14:17:00	1	18.15	3	N	1	70.0	0.0	0.0	16.16	10.50	
1	2	2018-03-29 13:37:00	2018-03-29 14:15:00	1	4.59	1	N	1	25.0	0.0	0.5	5.16	0.00	
2	2	2018-03-29 13:26:00	2018-03-29 13:28:00	1	0.30	1	N	1	3.0	0.0	0.5	0.76	0.00	
3	2	2018-03-29 13:07:00	2018-03-29 14:03:00	2	16.97	1	N	1	49.5	0.0	0.5	5.61	5.76	
4	2	2018-03-29 14:19:00	2018-03-29 15:19:00	5	14.45	1	N	1	45.5	0.0	0.5	10.41	5.76	

In [23]:
```
df.info()
```

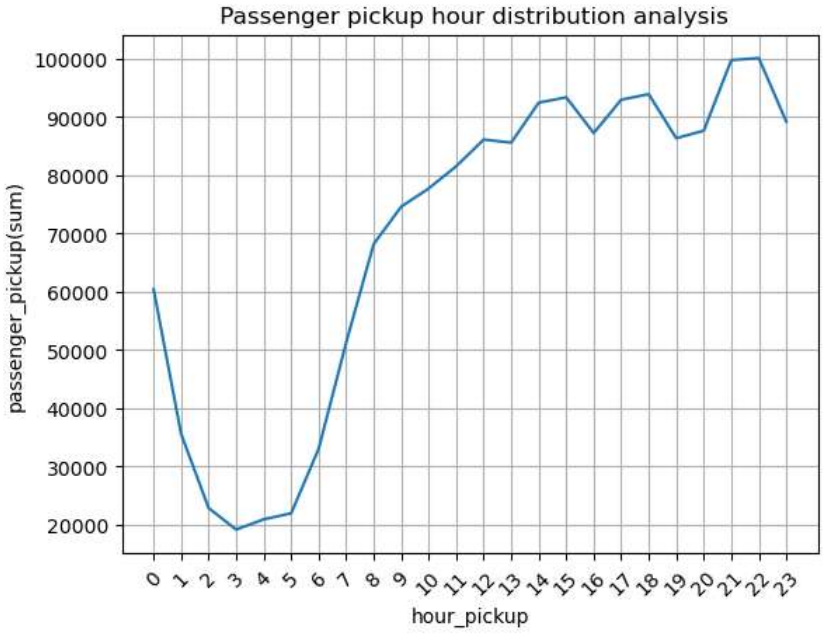
```
<class 'pandas.core.frame.DataFrame'>
Index: 1038957 entries, 0 to 1048574
Data columns (total 19 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   vendor_id            1038957 non-null  int64
 1   pickup_datetime      1038957 non-null  datetime64[ns]
 2   dropoff_datetime     1038957 non-null  datetime64[ns]
 3   passenger_count      1038957 non-null  int64
 4   trip_distance        1038957 non-null  float64
 5   rate_code            1038957 non-null  int64
 6   store_and_fwd_flag   1038957 non-null  object
 7   payment_type         1038957 non-null  int64
 8   fare_amount          1038957 non-null  float64
 9   extra                1038957 non-null  float64
 10  mta_tax              1038957 non-null  float64
 11  tip_amount           1038957 non-null  float64
 12  tolls_amount         1038957 non-null  float64
 13  imp_surcharge        1038957 non-null  float64
 14  total_amount         1038957 non-null  float64
 15  pickup_location_id   1038957 non-null  int64
 16  dropoff_location_id  1038957 non-null  int64
 17  pickup_hour          1038957 non-null  int32
 18  dropoff_hour         1038957 non-null  int32
dtypes: datetime64[ns](2), float64(8), int32(2), int64(6), object(1)
memory usage: 150.6+ MB
```

In [24]:
```
passenger_pickup=df.groupby(['pickup_hour'])['passenger_count'].sum()
passenger_dropoff=df.groupby(['dropoff_hour'])['passenger_count'].sum()
```

In [25]:
```
hours_pickup =[hour for hour, df in df.groupby(['pickup_hour'])]
```

In [26]:
```
hours_pickup = pd.Series(hours_pickup)
```

In [27]:
```
plt.plot(hours_pickup.index, passenger_pickup)
plt.xticks(hours_pickup.index, rotation =45, size=10)
plt.xlabel('hour_pickup')
plt.ylabel('passenger_pickup(sum)')
plt.title(' Passenger pickup hour distribution analysis')
plt.grid()
plt.show()
```

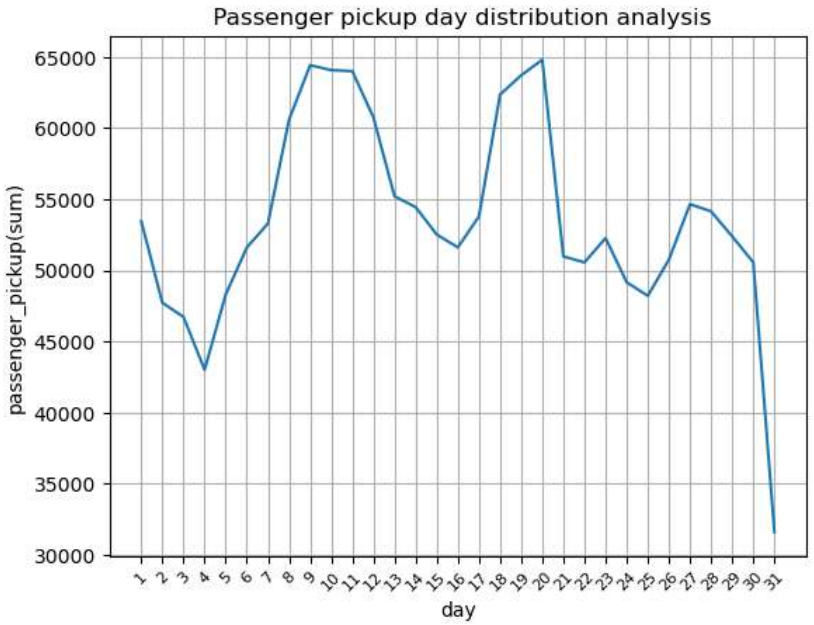


In [ ]:
```
```

In [28]:
```
df['month_pickup'] = df['pickup_datetime'].dt.month
df['day_pickup'] =df['pickup_datetime'].dt.day
```

In [29]:
```
passenger_pickup_day = df.groupby(['day_pickup'])['passenger_count'].sum()
```

In [30]:
```
plt.plot(passenger_pickup_day.index, passenger_pickup_day)
plt.xticks(passenger_pickup_day.index, rotation =45, size=8)
plt.xlabel('day')
plt.ylabel('passenger_pickup(sum)')
plt.title(' Passenger pickup day distribution analysis')
plt.grid()
plt.show()
```



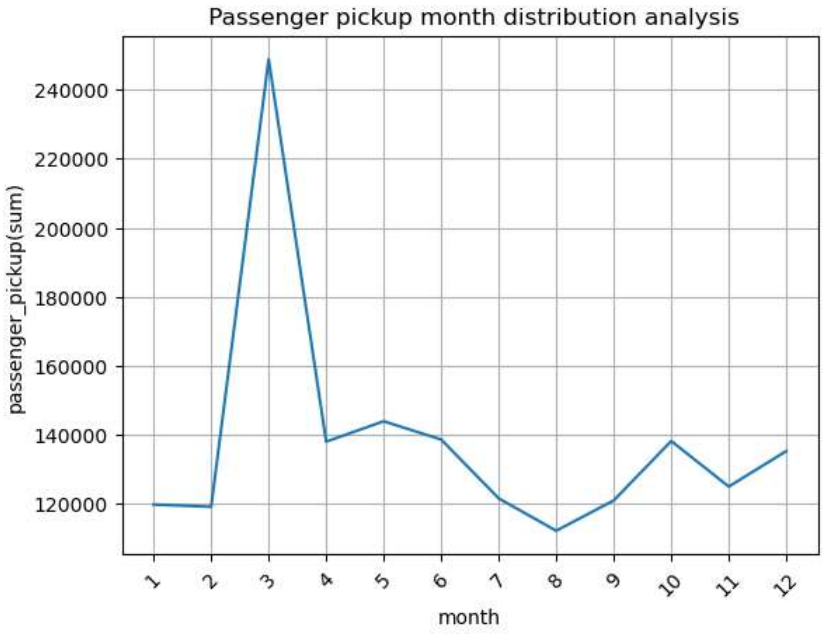
In [31]:
```
df['month_pickup']= pd.Series(df['month_pickup'])
```

In [32]:
```
passenger_pickup_month = df.groupby(['month_pickup'])['passenger_count'].sum()
```

In [33]:
```
passenger_pickup_month.index
```

Out[33]:
```
Index([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], dtype='int32', name='month_pickup')
```

In [34]:
```
plt.plot(passenger_pickup_month.index, passenger_pickup_month)
plt.xticks(passenger_pickup_month.index, rotation =45, size=10)
plt.xlabel('month')
plt.ylabel('passenger_pickup(sum)')
plt.title(' Passenger pickup month distribution analysis')
plt.grid()
plt.show()
```



Q3: What is the busiest location by hourly?

The busiest loaction for pickup passengers is location 265

In [35]:
```
grouped2 = df.groupby('pickup_hour').agg({'passenger_count': 'sum', 'pickup_location_id': 'idxmax'}).reset_index()
```

In [36]:
```
grouped2['highest_passenger_number_location']= df.loc[grouped2['pickup_location_id']]['pickup_location_id'].values


# df.loc[grouped['pickup_location_id']] is used to retrieve rows from the original DataFrame df based on the indices specified in grouped['pickup_location_id'].
# This essentially selects the rows from df that correspond to the locations with the highest passenger count for each hour.
# ['pickup_location_id'].values is used to extract the 'pickup_location_id' column from the selected rows as a NumPy array.
```

In [37]:
```
grouped2
```

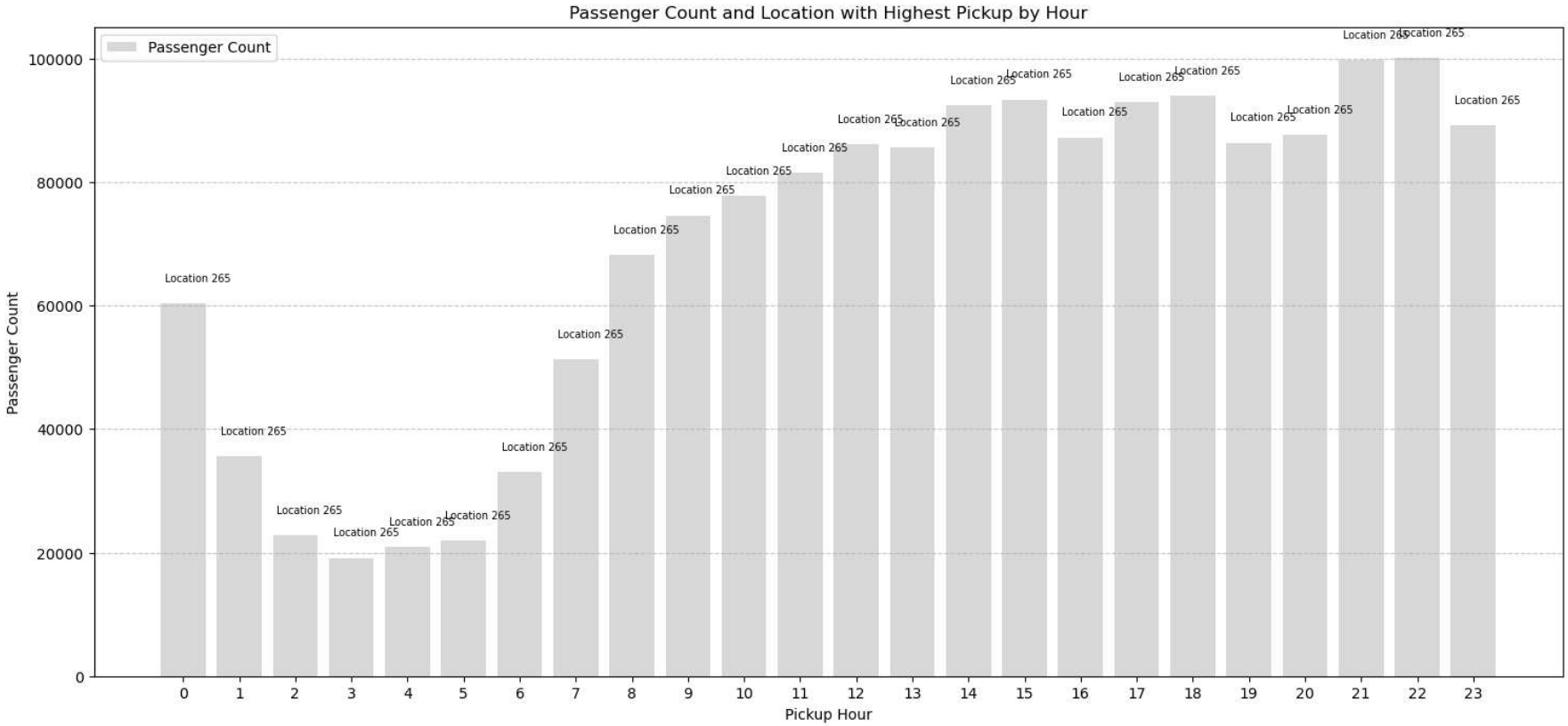
Out[37]:

	pickup_hour	passenger_count	pickup_location_id	highest_passenger_number_location
0	0	60376	2589	265
1	1	35549	6650	265
2	2	22822	8291	265
3	3	19122	389	265
4	4	20880	19948	265
5	5	21918	914	265
6	6	33001	46200	265
7	7	51242	90035	265
8	8	68156	12766	265
9	9	74581	7367	265
10	10	77712	12097	265
11	11	81523	3057	265
12	12	86105	20942	265
13	13	85586	9911	265
14	14	92445	28294	265
15	15	93365	2434	265
16	16	87231	5273	265
17	17	92953	8828	265
18	18	93902	14049	265
19	19	86341	762	265
20	20	87628	445	265
21	21	99744	45956	265
22	22	100113	627	265
23	23	89210	3995	265

In [38]:

```
# Create plots for 'passenger_count' and annotate with the highest location
plt.figure(figsize=(18, 8))
plt.bar(grouped2['pickup_hour'], grouped2['passenger_count'], label='Passenger Count', alpha=0.3, color='grey')

# Annotate the location with the highest passenger count
for index, row in grouped2.iterrows():
    plt.annotate(f'Location {row["highest_passenger_number_location"]}',
                 (row["pickup_hour"], row["passenger_count"] + 10),  # Adjust the position for annotation
                 textcoords="offset points",
                 xytext=(10, 15),  # Offset the annotation text
                 ha='center',
                fontsize=7)

# Set labels and title
```

```
plt.xlabel('Pickup Hour')
plt.ylabel('Passenger Count')
plt.title('Passenger Count and Location with Highest Pickup by Hour')
plt.xticks(grouped2['pickup_hour'])

# Add a legend
plt.legend()

# Show the plot
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Q4: What are the most common pickup and dropoff locations?

Location 265 is the most common pickup and dropoff location

In [47]:
```
grouped3 = df.groupby('pickup_hour').agg({'passenger_count': 'sum', 'pickup_location_id': 'idxmax', 'dropoff_location_id':'idxmax'}).reset_index()
grouped3['highest_passenger_number_pickup_location']= df.loc[grouped3['pickup_location_id']]['pickup_location_id'].values
grouped3['highest_passenger_number_drop_location'] =df.loc[grouped3['dropoff_location_id']]['dropoff_location_id'].values
```

In [48]:
```
grouped3=grouped3.rename(columns={'pickup_location_id':'highest_pickup_number', 'dropoff_location_id':'highest_dropoff_number'})
```

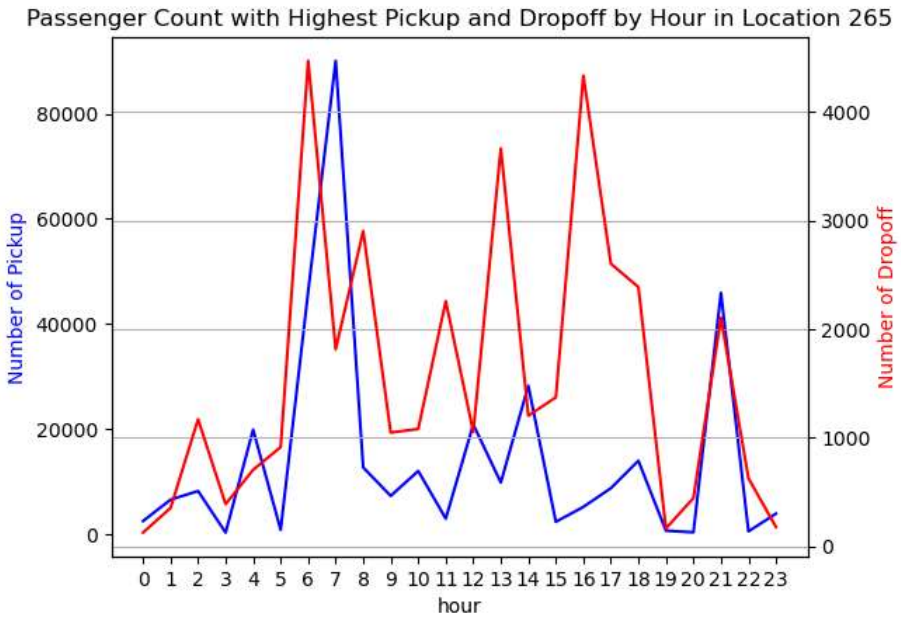
In [53]:
```
hours3=[hour for hour,df in df.groupby('pickup_hour')]
hours3=pd.Series(hours3)
```

In [54]:
```
grouped3.head()
```

Out[54]:

	pickup_hour	passenger_count	highest_pickup_number	highest_dropoff_number	highest_passenger_number_pickup_location	highest_passenger_number_drop_location
0	0	60376	2589	127	265	265
1	1	35549	6650	354	265	265
2	2	22822	8291	1170	265	265
3	3	19122	389	389	265	265
4	4	20880	19948	705	265	265

In [55]:
```
## procuding overlabeled figures
fig, ax1 =plt.subplots()
ax2 = ax1.twinx()
ax1.plot(hours3, grouped3['highest_pickup_number'], color='blue')
ax2.plot (hours3, grouped3['highest_dropoff_number'], color='red')

ax1.set_xlabel('hour')
ax1.set_ylabel('Number of Pickup', color='blue', size =10)
ax2.set_ylabel('Number of Dropoff', color ='red', size =10)
plt.xticks(hours3, rotation='vertical', size =8)
plt.title('Passenger Count with Highest Pickup and Dropoff by Hour in Location 265')
plt.grid()
plt.show()
```



Q5: Over all of the locations, how the passengers distribute

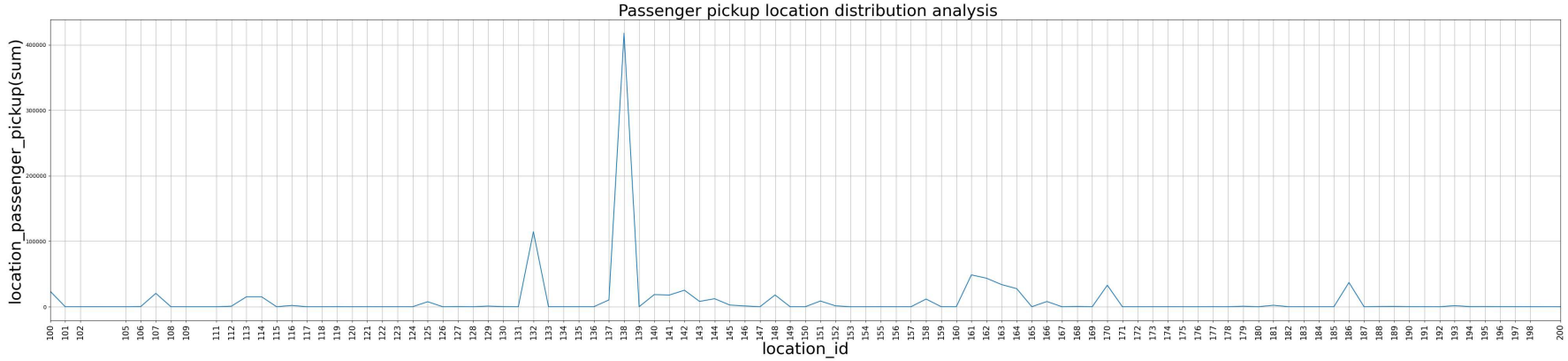
location 138 has the highest passenger pickup

In [56]:
```
location_pickup =df.groupby('pickup_location_id')['passenger_count'].sum()
```

In [57]:
```
location_ids =[location_id for location_id, df in df.groupby('pickup_location_id')]
```

In [58]:
```
location_ids_series = pd.Series([i for i in location_ids])
```

In [59]:
```
plt.figure(figsize=(50, 10))
plt.plot(location_ids_series, location_pickup)
plt.xticks(location_ids_series, rotation ='vertical', size=15)
plt.xlabel('location_id', size=30)
plt.ylabel('location_passenger_pickup(sum)', size=30)
plt.title(' Passenger pickup location distribution analysis', size=30)

plt.xlim(100, 200)
plt.grid()
plt.show()
```



In [ ]:
```

```