In [1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
from datetime import timedelta
import datetime as dt
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.cluster import MiniBatchKMeans
import warnings
from sklearn.model_selection import train_test_split
```

In [2]:
```
data = pd.read_csv('green_tripdata_2016-12.csv',index_col=False)
data
```

Out[2]:

	VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	store_and_fwd_flag	RatecodeID	PULocationID	DOLocationID	passenger_
0	2	2016-12-01 00:00:54	2016-12-01 00:06:54	N	1	92	192	
1	2	2016-12-01 00:52:41	2016-12-01 00:54:51	N	1	92	171	
2	2	2016-12-01 00:10:39	2016-12-01 00:14:47	N	1	75	238	
3	2	2016-12-01 00:12:16	2016-12-01 00:15:31	N	1	166	151	
4	2	2016-12-01 00:29:22	2016-12-01 00:39:51	N	1	166	42	
...	...	...	...	...	...	...	...	
1224153	1	2016-12-31 23:00:16	2016-12-31 23:05:30	N	1	74	75	
1224154	1	2016-12-31 23:00:20	2016-12-31 23:04:05	N	1	42	41	
1224155	1	2016-12-31 23:00:08	2016-12-31 23:15:57	N	1	243	159	
1224156	1	2016-12-31 23:00:00	2016-12-31 23:10:15	N	1	244	120	
1224157	1	2016-12-31 23:00:10	2016-12-31 23:04:17	N	1	7	7	

1224158 rows × 19 columns

- VendorID - A code indicating the LPEP provider that provided the record. (1= Creative Mobile Technologies, LLC; 2= VeriFone Inc).
- lpep_pickup_datetime - The date and time when the meter was engaged.
- lpep_dropoff_datetime - The date and time when the meter was disengaged.

- store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. ('Y' = store and forward trip 'N' = not a store and forward trip)
- RatecodeID - The final rate code in effect at the end of the trip. (1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride)
- PULocationID - pickup location matching zone numbers to the map
- DOLocationID - drop location matching zone numbers to the map
- passenger_count - The number of passengers in the vehicle. This is a driver-entered value.
- trip_distance - The elapsed trip distance in miles reported by the taximeter.
- fare_amount - The time-and-distance fare calculated by the meter.
- extra - Miscellaneous extras and surcharges.
- mta_tax - $0.50 MTA tax that is automatically triggered based on the metered rate in use.
- tip_amount - **This field is automatically populated for credit card tips. Cash tips are not included.**
- tolls_amount - Total amount of all tolls paid in trip.
- ehail_fee - None
- improvement_surcharge - $0.30 improvement surcharge assessed on hailed trips at the flag drop. The improvement surcharge began being levied in 2015.
- total_amount - The total amount charged to passengers. Does not include cash tips.
- payment_type - A numeric code signifying how the passenger paid for the trip. (1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip)
- trip_type - A code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver. (1= Street-hail 2= Dispatch)

1. Characterize the data and comment about its quality

In [3]:
```
data.shape
```

Out[3]:
```
(1224158, 19)
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1224158 entries, 0 to 1224157
Data columns (total 19 columns):
 #   Column                 Non-Null Count    Dtype  
---  ------                 --------------    -----  
 0   VendorID               1224158 non-null  int64  
 1   lpep_pickup_datetime   1224158 non-null  object 
 2   lpep_dropoff_datetime  1224158 non-null  object 
 3   store_and_fwd_flag     1224158 non-null  object 
 4   RatecodeID             1224158 non-null  int64  
 5   PULocationID           1224158 non-null  int64  
 6   DOLocationID           1224158 non-null  int64  
 7   passenger_count        1224158 non-null  int64  
 8   trip_distance          1224158 non-null  float64
 9   fare_amount            1224158 non-null  float64
 10  extra                  1224158 non-null  float64
 11  mta_tax                1224158 non-null  float64
 12  tip_amount             1224158 non-null  float64
 13  tolls_amount           1224158 non-null  float64
 14  ehail_fee              0 non-null        float64
 15  improvement_surcharge  1224158 non-null  float64
 16  total_amount           1224158 non-null  float64
 17  payment_type           1224158 non-null  int64  
 18  trip_type              1224158 non-null  int64  
dtypes: float64(9), int64(7), object(3)
memory usage: 177.5+ MB
```

In [5]:
```
data.describe()
```

Out[5]:

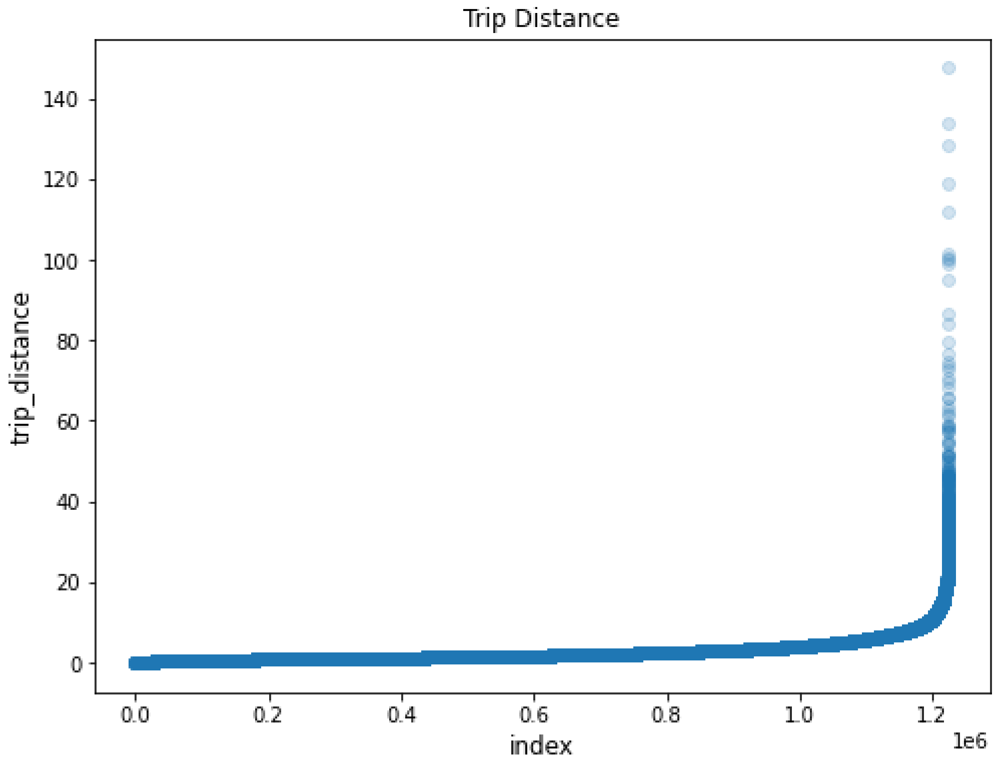
	VendorID	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount	extra	mta_
count	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e+06	1.224158e
mean	1.799278e+00	1.085079e+00	1.131729e+02	1.290862e+02	1.362447e+00	2.618179e+00	1.159368e+01	3.629562e-01	4.872578e
std	4.005404e-01	5.660195e-01	7.628387e+01	7.723367e+01	1.036924e+00	2.806160e+00	9.793499e+00	3.901575e-01	8.517500e
min	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	-4.990000e+02	-4.500000e+00	-5.00000
25%	2.000000e+00	1.000000e+00	4.900000e+01	6.100000e+01	1.000000e+00	9.800000e-01	6.000000e+00	0.000000e+00	5.000000e
50%	2.000000e+00	1.000000e+00	8.300000e+01	1.290000e+02	1.000000e+00	1.700000e+00	9.000000e+00	5.000000e-01	5.000000e
75%	2.000000e+00	1.000000e+00	1.730000e+02	1.930000e+02	1.000000e+00	3.200000e+00	1.400000e+01	5.000000e-01	5.000000e
max	2.000000e+00	6.000000e+00	2.650000e+02	2.650000e+02	9.000000e+00	1.475000e+02	1.007000e+03	4.500000e+00	5.000000e

- **There're 1224158 data samples with 19 features. There're 3 objective columns, the others are continuous variables. 'ehail_fee' are all NAN, no missing values in this table other than that.**
- **There're negative values in 'fare_amount', 'extra', 'mta_tax','tip_amount','tolls_amount','improvement_surcharge' and 'total_amount', and value 0 in 'passenger_count', 'trip_distance', which makes no sense.**
- **The maximum trip_distance of 147.5 miles, tip_amount of 250.7 dollars, tolls_amount of 298 dollars, total_amount of 1008.3 dollars are also weird.**

2. Visualize and Clean Data

Trip Distance

In [6]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["trip_distance"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('trip_distance', fontsize=12)
plt.title('Trip Distance')
plt.show()
```



In [7]:
```
data["trip_distance"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

Out[7]:
```
0.00      0.00
0.01      0.00
0.25      0.98
0.50      1.70
0.75      3.20
0.99     13.97
1.00    147.50
Name: trip_distance, dtype: float64
```

The 99 quantile is 13.97 miles while the maximum is 147.5 miles

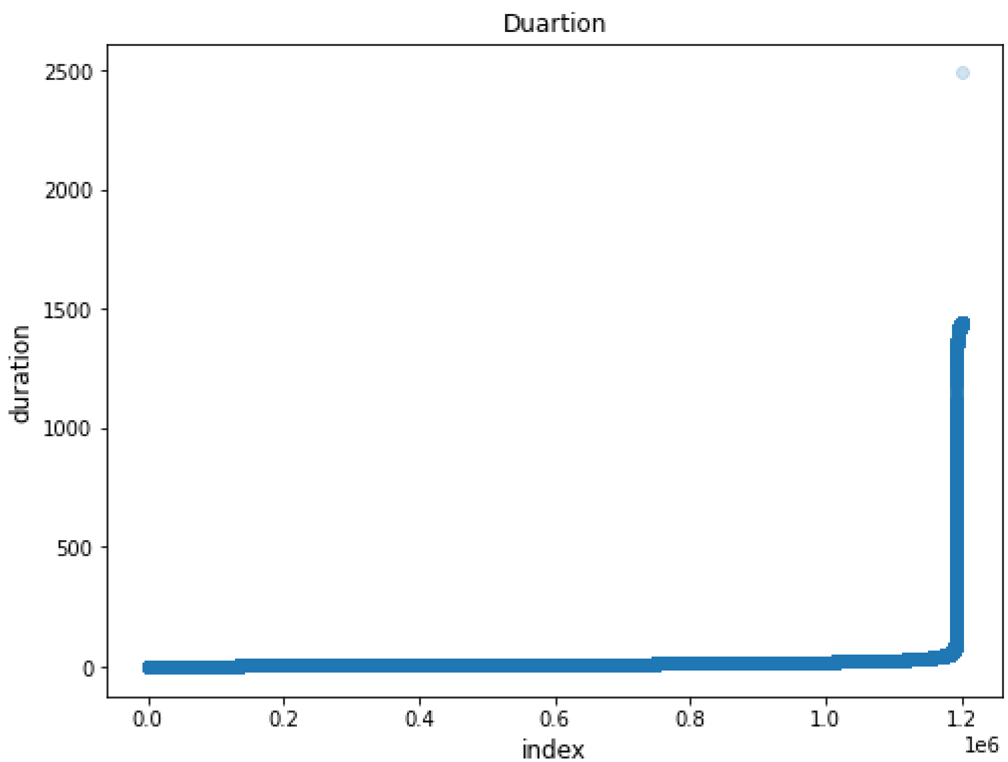
clipping trip distance longer than 20 miles and shorter than 0.1 mile.

In [8]:
```
data = data.loc[(data["trip_distance"]<=20)&(data["trip_distance"]>0.1)].reset_index(drop=True)
```

Trip Duration

In [9]:
```
data.lpep_pickup_datetime = pd.to_datetime(data.lpep_pickup_datetime)
data.lpep_dropoff_datetime = pd.to_datetime(data.lpep_dropoff_datetime)
data['duration'] = (data.lpep_dropoff_datetime - data.lpep_pickup_datetime).dt.total_seconds() / 60
```

In [10]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["duration"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('duration', fontsize=12)
plt.title('Duartion')
plt.show()
```



In [11]:
```
data["duration"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

Out[11]:
```
0.00       0.000000
0.01       1.583333
0.25       5.883333
0.50       9.750000
0.75      16.116667
0.99      64.433333
1.00    2487.900000
Name: duration, dtype: float64
```

The 99 quantile is 64.5 min while the maximum is 1439.9 min

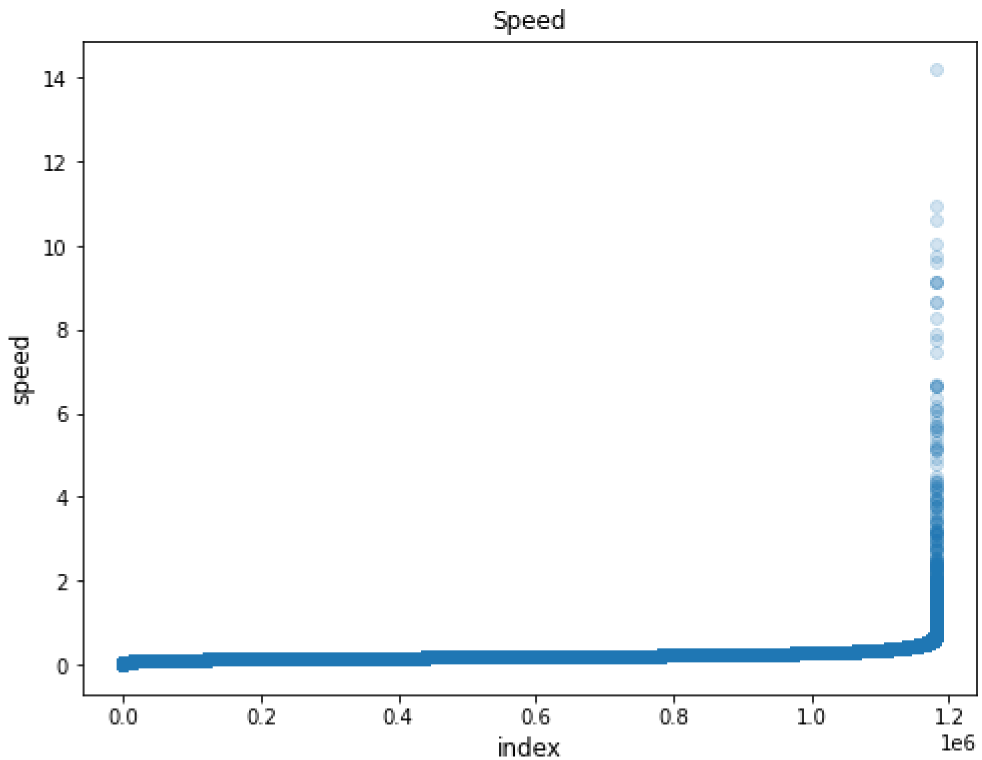
clipping trip distance longer than 60 min and less that one minute

In [12]:
```
data = data.loc[(data["duration"]<=60)&(data["duration"]>=1)].reset_index(drop=True)
```

Speed

In [13]:
```
data['speed'] = data.trip_distance/data.duration
```

In [14]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["speed"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('speed', fontsize=12)
plt.title('Speed')
plt.show()
```



In [15]:
```
data.speed.describe()
```

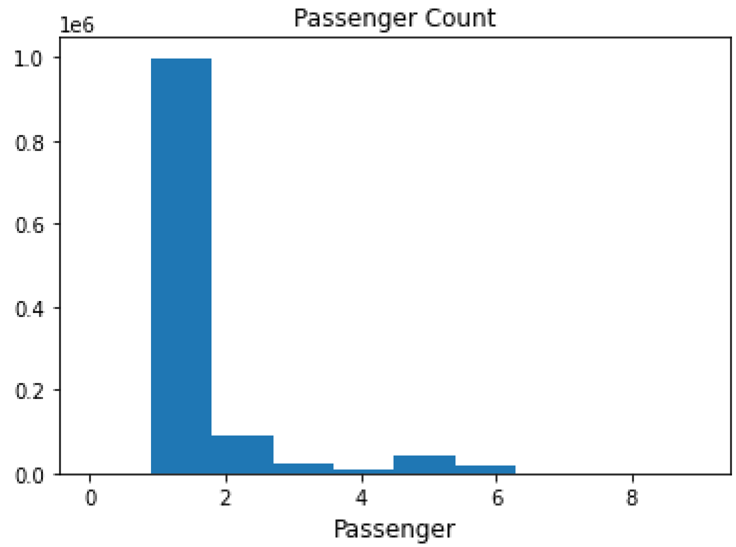
Out[15]:
```
count    1.181390e+06
mean     2.057084e-01
std      1.003032e-01
min      2.071346e-03
25%      1.496104e-01
50%      1.876161e-01
75%      2.376238e-01
max      1.419048e+01
Name: speed, dtype: float64
```

Delete records with speed more than 1 mile/min.

In [16]:
```
data = data.loc[data.speed<=1].reset_index(drop=True)
```

Passenger Count

In [17]:
```
plt.hist(data.passenger_count)
plt.title('Passenger Count')
plt.xlabel('Passenger', fontsize=12)
plt.show()
```



In [18]:
```
data["passenger_count"].value_counts()
```

Out[18]:
```
1    997993
2     89920
5     42155
3     23273
6     20086
4      7666
0        29
8         8
7         7
9         1
Name: passenger_count, dtype: int64
```

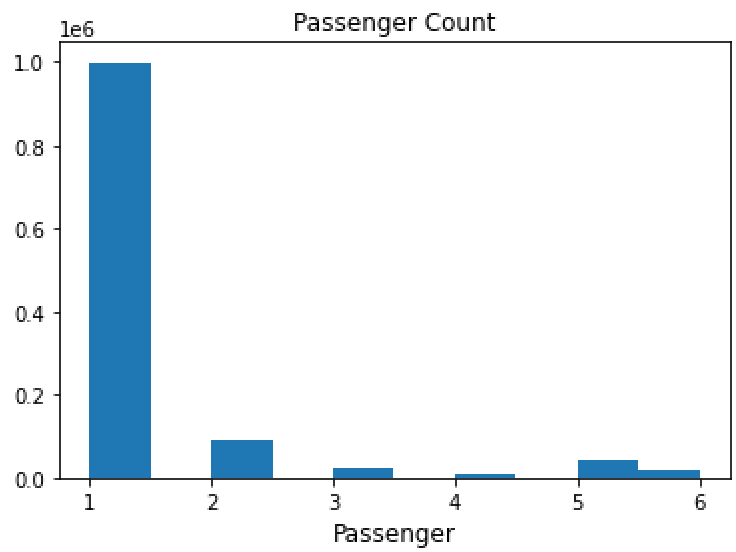
In [19]:
```
data["passenger_count"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

Out[19]:
```
0.00    0.0
0.01    1.0
0.25    1.0
0.50    1.0
0.75    1.0
0.99    6.0
1.00    9.0
Name: passenger_count, dtype: float64
```

Delete records with 0 pssenger and passenger_count greater than 6.

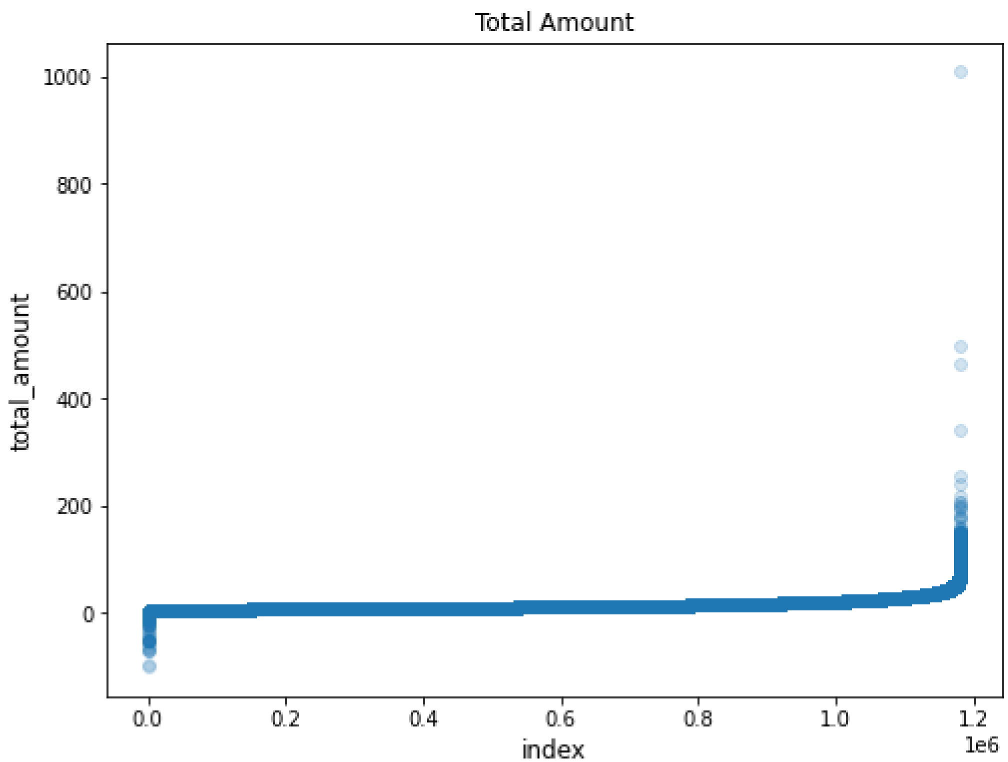
In [20]:
```
data = data.loc[(data["passenger_count"]>0) & (data["passenger_count"]<=6)].reset_index(drop=True)
```

In [21]:
```
plt.hist(data.passenger_count)
plt.title('Passenger Count')
plt.xlabel('Passenger', fontsize=12)
plt.show()
```



Total Amount

In [22]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["total_amount"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('total_amount', fontsize=12)
plt.title('Total Amount')
plt.show()
```



In [23]:
```
data["total_amount"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

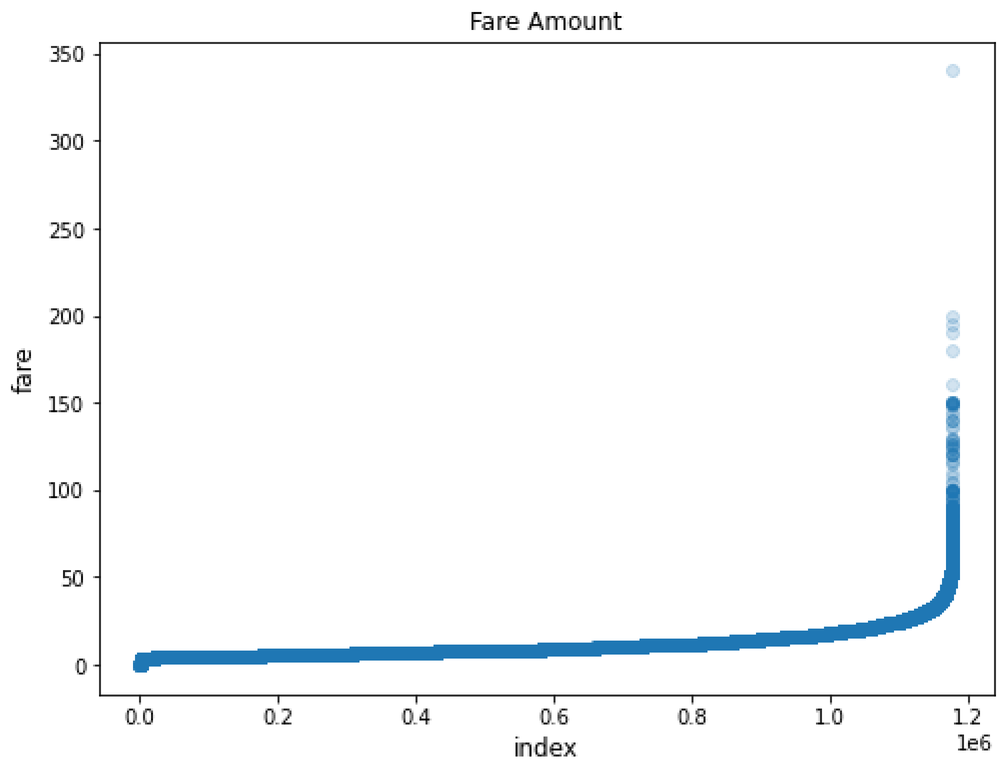
Out[23]:
```
0.00    -100.0
0.01       4.3
0.25       7.8
0.50      10.8
0.75      16.3
0.99      47.3
1.00    1008.3
Name: total_amount, dtype: float64
```

Delete total amount larger than 400 and less that 0

In [24]:
```
data = data.loc[(data["total_amount"]<=400)&(data["total_amount"]>0)].reset_index(drop=True)
```

Fare Amount

In [25]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["fare_amount"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('fare', fontsize=12)
plt.title('Fare Amount')
plt.show()
```



In [26]:
```
data["fare_amount"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

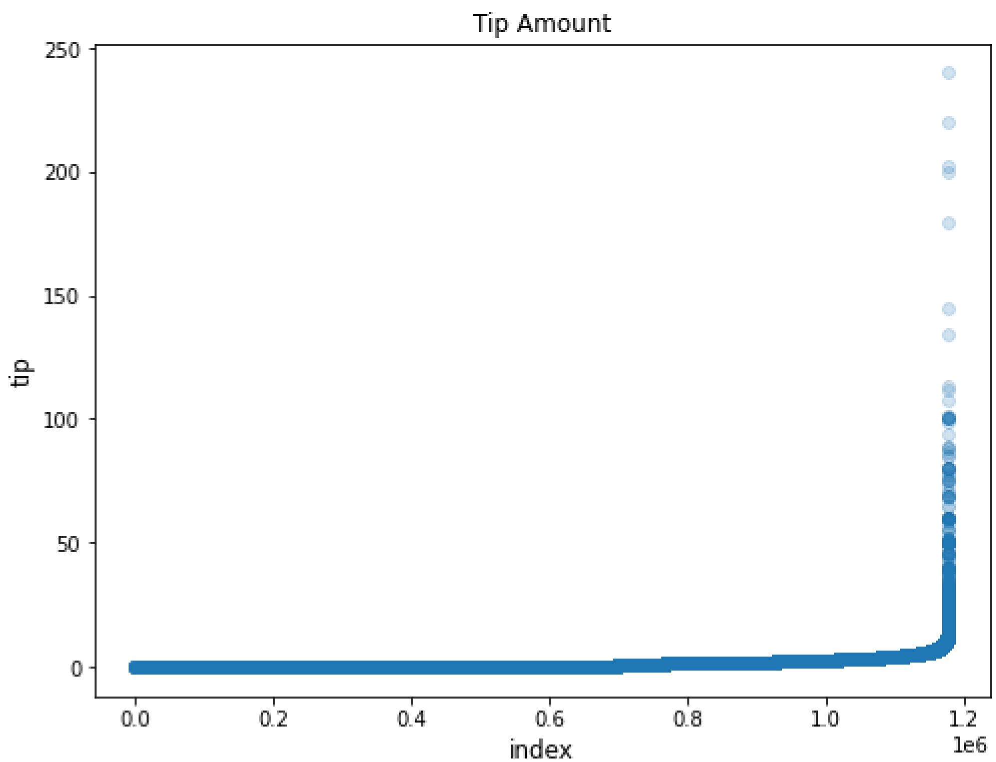
Out[26]:
```
0.00      0.0
0.01      3.5
0.25      6.0
0.50      9.0
0.75     14.0
0.99     40.0
1.00    340.0
Name: fare_amount, dtype: float64
```

Delete records with 0 fare amount.

In [27]:
```
data = data.loc[data.fare_amount>0].reset_index(drop=True)
```

Tip Amount

In [28]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(data.shape[0]), np.sort(data["tip_amount"].values), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('tip', fontsize=12)
plt.title('Tip Amount')
plt.show()
```



In [29]:
```
data["tip_amount"].quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

Out[29]:
```
0.00      0.00
0.01      0.00
0.25      0.00
0.50      0.00
0.75      1.86
0.99      7.84
1.00    240.00
Name: tip_amount, dtype: float64
```

Delete records with which tips is larger than fare amount

In [30]:
```
data = data.loc[data.tip_amount<data.fare_amount].reset_index(drop=True)
```

In [31]:
```
data.describe()
```

Out[31]:

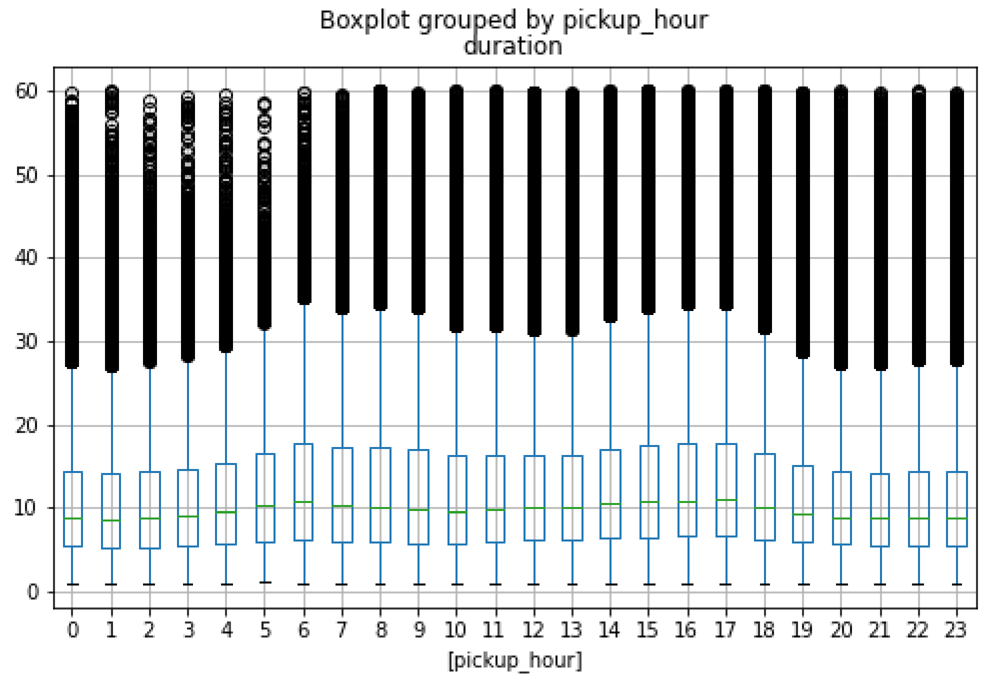
	VendorID	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount	extra	mta_t
count	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+06	1.176458e+
mean	1.800384e+00	1.045240e+00	1.127070e+02	1.290701e+02	1.363125e+00	2.580974e+00	1.131171e+01	3.669749e-01	4.944996e-
std	3.997120e-01	4.155648e-01	7.613657e+01	7.720240e+01	1.038978e+00	2.482266e+00	7.698733e+00	3.794237e-01	5.215316e-
min	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.100000e-01	1.000000e-02	0.000000e+00	0.000000e+
25%	2.000000e+00	1.000000e+00	4.900000e+01	6.100000e+01	1.000000e+00	1.000000e+00	6.000000e+00	0.000000e+00	5.000000e-
50%	2.000000e+00	1.000000e+00	8.200000e+01	1.290000e+02	1.000000e+00	1.710000e+00	9.000000e+00	5.000000e-01	5.000000e-
75%	2.000000e+00	1.000000e+00	1.680000e+02	1.930000e+02	1.000000e+00	3.200000e+00	1.400000e+01	5.000000e-01	5.000000e-
max	2.000000e+00	6.000000e+00	2.650000e+02	2.650000e+02	6.000000e+00	2.000000e+01	3.400000e+02	4.500000e+00	5.000000e-

3. Find interesting trip statistics grouped by hour

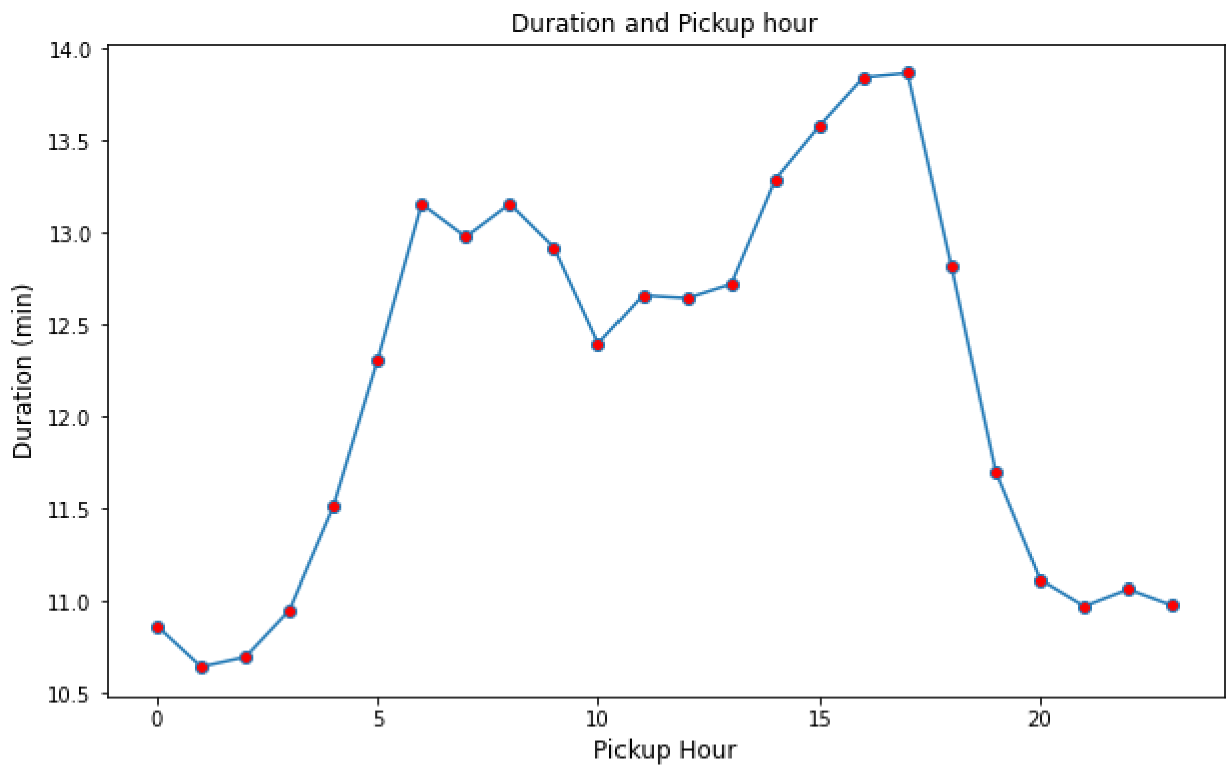
In [32]:
```
data['pickup_hour'] = data['lpep_pickup_datetime'].dt.hour
data['dropoff_hour'] = data['lpep_dropoff_datetime'].dt.hour
data['pickup_weekday'] = data['lpep_pickup_datetime'].dt.weekday
data['dropoff_weekday'] = data['lpep_dropoff_datetime'].dt.weekday
```

In [33]:
```
metric = 'pickup_hour'
y = 'duration'
data.boxplot(column=y, by=[metric],figsize=(8,5))
```

Out[33]:
```
<AxesSubplot:title={'center':'duration'}, xlabel='[pickup_hour]'>
```



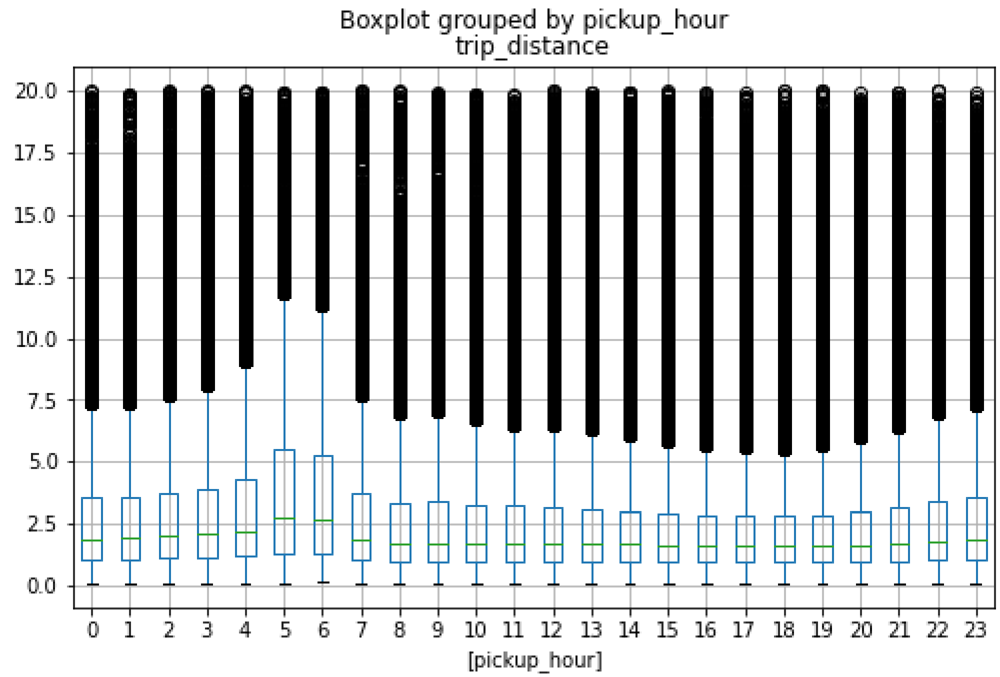
In [34]:
```
duration_hour = data.groupby(by='pickup_hour',as_index=False)['duration'].mean()
plt.figure(figsize=(10,6))
plt.plot(duration_hour['pickup_hour'], duration_hour['duration'],marker='o', markerfacecolor='red',label
plt.xlabel('Pickup Hour', fontsize=12)
plt.ylabel('Duration (min)', fontsize=12)
plt.title('Duration and Pickup hour')
plt.show()
```



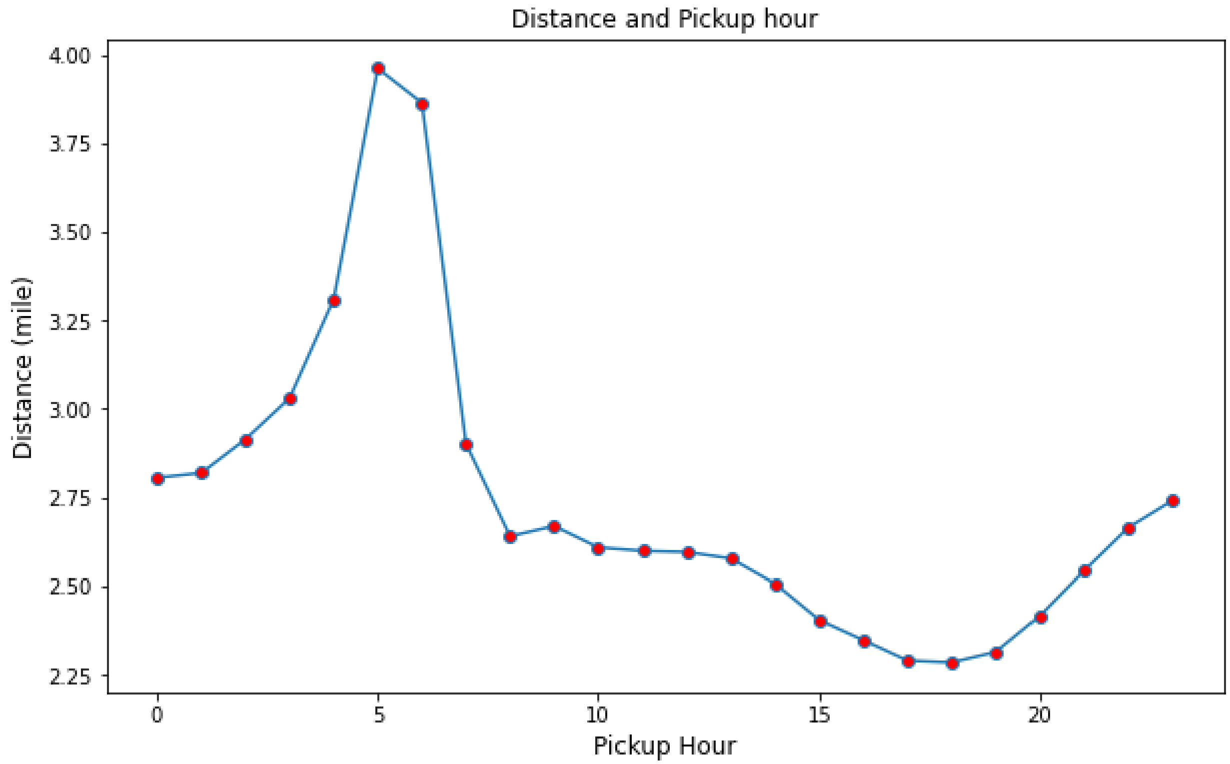
The trip duration at night (7pm - 4am) is shorter than any other time during the day.

In [35]:
```
metric = 'pickup_hour'
y = 'trip_distance'
data.boxplot(column=y, by=[metric],figsize=(8,5))
```

Out[35]: <AxesSubplot:title={'center':'trip_distance'}, xlabel='[pickup_hour]'>



In [36]:
```
distance_hour = data.groupby(by='pickup_hour',as_index=False)['trip_distance'].mean()
plt.figure(figsize=(10,6))
plt.plot(distance_hour['pickup_hour'], distance_hour['trip_distance'], marker='o', markerfacecolor='red'
plt.xlabel('Pickup Hour', fontsize=12)
plt.ylabel('Distance (mile)', fontsize=12)
plt.title('Distance and Pickup hour')
plt.show()
```



Trip distance around 5am-6am is more likely to be longer than any other time during the day.

4. What kind of trip yields better tips?

According to the data dictionary on the website, the records only include tip information paid by credit card. So delete records with other payment method.

When defining better tips, it's more reasonable to evaluate tip percentage other than its absolute value. So the model is built to predict which trip yields a higher tip percentage.

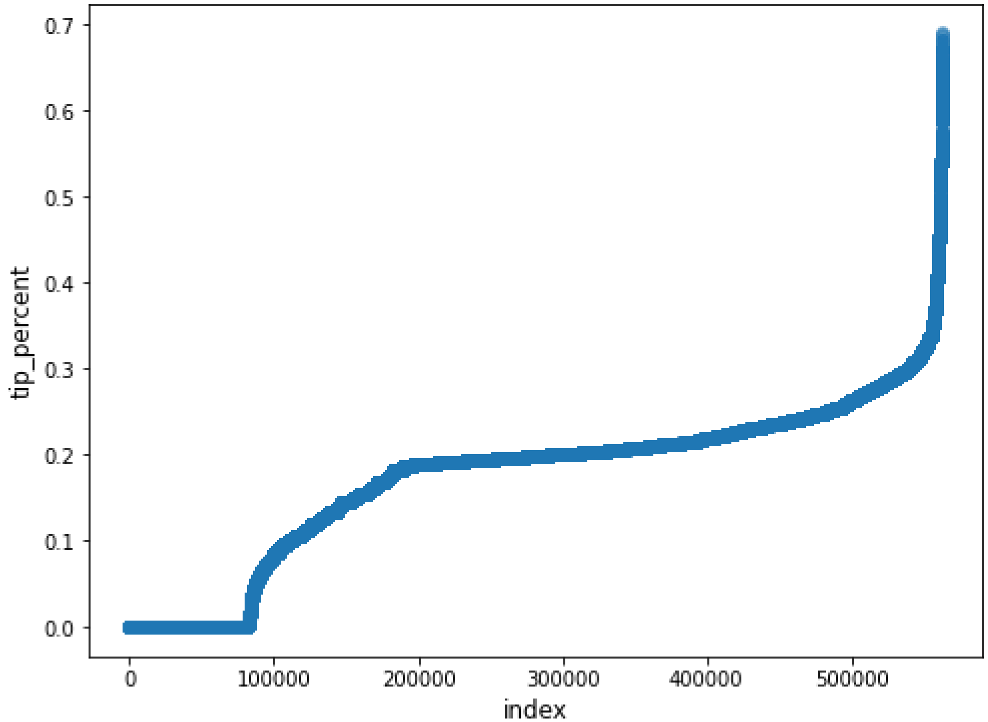
In [37]:
```
data['tip_percent'] = data.tip_amount/data.fare_amount
```

In [38]:
```
df = data.loc[data.payment_type==1].reset_index(drop=True)
```

In [39]:
```
df.tip_percent.quantile([.0, 0.01, 0.25, .5, 0.75, 0.99, 1])
```

Out[39]:
```
0.00    0.000000
0.01    0.000000
0.25    0.142857
0.50    0.217931
0.75    0.252000
0.99    0.440741
1.00    0.993333
Name: tip_percent, dtype: float64
```

In [40]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(df.shape[0]), np.log(np.sort(df["tip_percent"].values)+1), alpha=0.2)
plt.xlabel('index', fontsize=12)
plt.ylabel('tip_percent', fontsize=12)
plt.show()
```



In [41]:
```
loction = pd.read_csv('taxi+_zone_lookup.csv')
loction
```

Out[41]:

	LocationID	Borough	Zone	service_zone
0	1	EWR	Newark Airport	EWR
1	2	Queens	Jamaica Bay	Boro Zone
2	3	Bronx	Allerton/Pelham Gardens	Boro Zone
3	4	Manhattan	Alphabet City	Yellow Zone
4	5	Staten Island	Arden Heights	Boro Zone
...	...	...	...	...
260	261	Manhattan	World Trade Center	Yellow Zone
261	262	Manhattan	Yorkville East	Yellow Zone
262	263	Manhattan	Yorkville West	Yellow Zone
263	264	Unknown	NV	NaN
264	265	Unknown	NaN	NaN

265 rows × 4 columns

Map the LocationID to borough.

In [42]:
```
d = {'Manhattan':1,'Queens':2,'Brooklyn':3,'Bronx':4,'Staten Island':5,'Unknown':6,'EWR':7}
df['PUL'] = df.PULocationID.map(lambda x:loction.Borough[x-1]).map(d)
df['DOL'] = df.DOLocationID.map(lambda x:loction.Borough[x-1]).map(d)
```

In [43]:
```
df['store_and_fwd_flag'] = df['store_and_fwd_flag'].map({'N':0,'Y':1})
```

In [44]:
```
df.columns
```

Out[44]:
```
Index(['VendorID', 'lpep_pickup_datetime', 'lpep_dropoff_datetime',
       'store_and_fwd_flag', 'RatecodeID', 'PULocationID', 'DOLocationID',
       'passenger_count', 'trip_distance', 'fare_amount', 'extra', 'mta_tax',
       'tip_amount', 'tolls_amount', 'ehail_fee', 'improvement_surcharge',
       'total_amount', 'payment_type', 'trip_type', 'duration', 'speed',
       'pickup_hour', 'dropoff_hour', 'pickup_weekday', 'dropoff_weekday',
       'tip_percent', 'PUL', 'DOL'],
      dtype='object')
```

In [45]:
```
ind = [0, 3, 4,15,16] + list(range(7, 13)) + list(range(18,22)) + [23,24,25,26,27]
features = df.columns[ind]
features
```

Out[45]:
```
Index(['VendorID', 'store_and_fwd_flag', 'RatecodeID', 'improvement_surcharge',
       'total_amount', 'passenger_count', 'trip_distance', 'fare_amount',
       'extra', 'mta_tax', 'tip_amount', 'trip_type', 'duration', 'speed',
       'pickup_hour', 'pickup_weekday', 'dropoff_weekday', 'tip_percent',
       'PUL', 'DOL'],
      dtype='object')
```

In [46]:
```
train,val = train_test_split(df[features],test_size=0.2)
```

Target Encoding

In [47]:
```
for col in ['pickup_weekday','pickup_hour','PUL', 'DOL']:
    gby = train.groupby(col).mean()[['duration','tip_amount']]
    gby.columns = ['%s_gby_%s' % (c, col) for c in gby.columns]
    train = pd.merge(train, gby, how='left', left_on=col, right_index=True)
    val = pd.merge(val, gby, how='left', left_on=col, right_index=True)
```

In [48]:
```
X_train = train.drop(columns=['tip_amount','tip_percent'])
y_train = train['tip_percent']
X_val = val.drop(columns=['tip_amount','tip_percent'])
y_val = val['tip_percent']
```

XGBoost Model

In [49]:
```
xgb_pars = {'min_child_weight': 50, 'eta': 0.1, 'colsample_bytree': 0.3, 'max_depth': 10,
            'subsample': 0.8, 'lambda': 1., 'nthread': -1, 'booster' : 'gbtree',
            'eval_metric': 'rmse', 'objective': 'reg:squarederror'}
```

In [50]:
```
dtrain = xgb.DMatrix(X_train.values, label=y_train)
dvalid = xgb.DMatrix(X_val.values,label=y_val)
watchlist = [(dtrain, 'train'), (dvalid, 'valid')]
```

In [51]:
```
t0 = dt.datetime.now()
gbm = xgb.train(xgb_pars, dtrain, 600, watchlist, early_stopping_rounds=50,
                maximize=False, verbose_eval=10)
t1 = dt.datetime.now()
print('Time fitting xgb: %i seconds' % (t1 - t0).seconds)
```

```
[1]	train-rmse:0.26846	valid-rmse:0.26866
[2]	train-rmse:0.24300	valid-rmse:0.24321
[3]	train-rmse:0.22269	valid-rmse:0.22293
[4]	train-rmse:0.20475	valid-rmse:0.20504
[5]	train-rmse:0.18837	valid-rmse:0.18870
[6]	train-rmse:0.17444	valid-rmse:0.17483
[7]	train-rmse:0.16228	valid-rmse:0.16272
[8]	train-rmse:0.15169	valid-rmse:0.15219
[9]	train-rmse:0.14254	valid-rmse:0.14310
[11]	train-rmse:0.12579	valid-rmse:0.12644
[12]	train-rmse:0.11993	valid-rmse:0.12064
[13]	train-rmse:0.11497	valid-rmse:0.11575
[14]	train-rmse:0.11085	valid-rmse:0.11167
[15]	train-rmse:0.10734	valid-rmse:0.10820
[16]	train-rmse:0.10440	valid-rmse:0.10530
[17]	train-rmse:0.10193	valid-rmse:0.10289
[18]	train-rmse:0.09985	valid-rmse:0.10088
[19]	train-rmse:0.09413	valid-rmse:0.09512
[21]	train-rmse:0.09149	valid-rmse:0.09258
```

The final RMSE for the validation test to predict a high tip percentage reaches 0.027.

Feature Importance

In [52]:
```
features = X_train.columns
```

In [53]:
```
feature_importance_dict = gbm.get_fscore()
fs = ['f%i' % i for i in range(len(features))]
f1 = pd.DataFrame({'f': list(feature_importance_dict.keys()), 'importance': list(feature_importance_dict
f2 = pd.DataFrame({'f': fs, 'feature_name': features})
feature_importance = pd.merge(f1, f2, how='right', on='f')
feature_importance = feature_importance.fillna(0)
```

In [54]:
```
feature_importance.sort_values('importance', ascending=False)
```

Out[54]:

	f	importance	feature_name
4	f4	20864	total_amount
11	f11	19181	duration
6	f6	17319	trip_distance
12	f12	14991	speed
7	f7	11606	fare_amount
13	f13	9674	pickup_hour
21	f21	8578	tip_amount_gby_pickup_hour
20	f20	6901	duration_gby_pickup_hour
14	f14	6691	pickup_weekday
15	f15	6084	dropoff_weekday
19	f19	4495	tip_amount_gby_pickup_weekday
24	f24	4145	duration_gby_DOL
18	f18	4104	duration_gby_pickup_weekday
17	f17	3919	DOL
5	f5	3784	passenger_count
16	f16	3534	PUL
22	f22	3152	duration_gby_PUL
23	f23	2895	tip_amount_gby_PUL
25	f25	2587	tip_amount_gby_DOL
0	f0	2373	VendorID
8	f8	1881	extra
2	f2	723	RatecodeID
9	f9	384	mta_tax
10	f10	358	trip_type

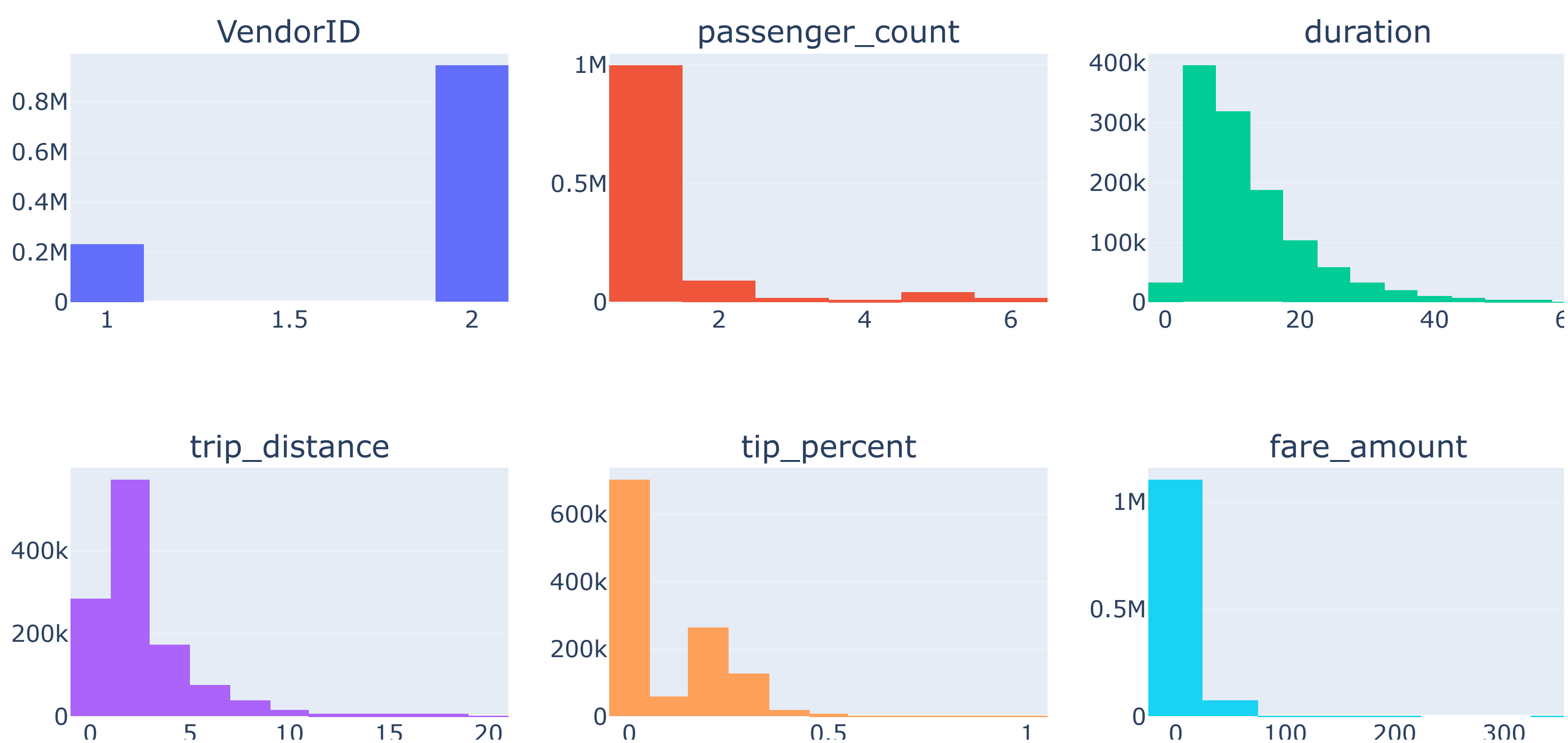
	f	importance	feature_name
3	f3	345	improvement_surcharge
1	f1	83	store_and_fwd_flag

5. Visualize the data to understand trip patterns

In [55]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=2,
                    cols=3,
                    subplot_titles=['VendorID', 'passenger_count', 'duration','trip_distance','tip_perce

fig.append_trace(go.Histogram(x=data['VendorID'], nbinsx=10),1,1)
fig.append_trace(go.Histogram(x=data['passenger_count'], nbinsx=10),1,2)
fig.append_trace(go.Histogram(x=data['duration'], nbinsx=20),1,3)
fig.append_trace(go.Histogram(x=data['trip_distance'], nbinsx=10),2,1)
fig.append_trace(go.Histogram(x=data['tip_percent'], nbinsx=10),2,2)
fig.append_trace(go.Histogram(x=data['fare_amount'], nbinsx=10),2,3)
fig.show()
```



- **Among the cleaned data, VeriFone Inc provides about 4 times records than Creative Mobile Technologies, LLC.**
- **1 passenger is the most common scenario.**
- **Most trips last around 2.5 min to 20 min.**
- **Most trip distances are around 1 mile to 3 miles.**
- **Most passengers won't pay the tip. Among tip payers, people are most willingly to pay 15% - 25% fare amount fee as the tip.**
- **Most fare amount are under $25.**