In [72]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
import warnings
import statsmodels.api as sm
import statsmodels.formula.api as smf
warnings.filterwarnings('ignore')
```

Loading Dataset

In [12]:
```
# encoding='unicode_escape' is used for removing unwanted spaces or _ - in the csv path
df=pd.read_csv(r'C:\Users\dharm\Downloads\2020_Yellow_Taxi_Trip_Data.csv',encoding='unicode_escape')
```

In [19]:
```
df.shape
```

Out[19]:
```
(24648499, 18)
```

In [13]:
```
df.head()
```

Out[13]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payme
0	1.0	01/01/2020 12:28:15 AM	01/01/2020 12:33:03 AM	1.0	1.2	1.0	N	238	239	
1	1.0	01/01/2020 12:35:39 AM	01/01/2020 12:43:04 AM	1.0	1.2	1.0	N	239	238	
2	1.0	01/01/2020 12:47:41 AM	01/01/2020 12:53:52 AM	1.0	0.6	1.0	N	238	238	
3	1.0	01/01/2020 12:55:23 AM	01/01/2020 01:00:14 AM	1.0	0.8	1.0	N	238	151	
4	2.0	01/01/2020 12:01:58 AM	01/01/2020 12:04:16 AM	1.0	0.0	1.0	N	193	193	

In [81]:
```
df_mean = np.mean(df['fare_amount'])
df_mean
```

Out[81]:
```
8.72986959585334
```

In [82]:
```
df_std = np.std(df['fare_amount'])
df_std
```

Out[82]:
```
3.595938337341993
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24648499 entries, 0 to 24648498
Data columns (total 18 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   VendorID               float64
 1   tpep_pickup_datetime   object 
 2   tpep_dropoff_datetime  object 
 3   passenger_count        float64
 4   trip_distance          float64
 5   RatecodeID             float64
 6   store_and_fwd_flag     object 
 7   PULocationID           int64  
 8   DOLocationID           int64  
 9   payment_type           float64
 10  fare_amount            float64
 11  extra                  float64
 12  mta_tax                float64
 13  tip_amount             float64
 14  tolls_amount           float64
 15  improvement_surcharge  float64
 16  total_amount           float64
 17  congestion_surcharge   float64
dtypes: float64(13), int64(2), object(3)
memory usage: 3.3+ GB
```

```
# EDA
```

# EDA

```
In [6]: df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
        df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-6-48671b08eba7> in <module>
----> 1 df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
      2 df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])

~\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exac
t, unit, infer_datetime_format, origin, cache)
    803             result = arg.map(cache_array)
    804         else:
--> 805             values = convert_listlike(arg._values, format)
    806             result = arg._constructor(values, index=arg.index, name=arg.name)
    807     elif isinstance(arg, (ABCDataFrame, abc.MutableMapping)):

~\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py in _convert_listlike_datetimes(arg, format, name, tz, unit, error
s, infer_datetime_format, dayfirst, yearfirst, exact)
    463         assert format is None or infer_datetime_format
    464         utc = tz == "utc"
--> 465         result, tz_parsed = objects_to_datetime64ns(
    466             arg,
    467             dayfirst=dayfirst,

~\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py in objects_to_datetime64ns(data, dayfirst, yearfirst, utc, error
s, require_iso8601, allow_object)
   2073
   2074     try:
-> 2075         result, tz_parsed = tslib.array_to_datetime(
   2076             data,
   2077             errors=errors,

pandas\_libs\tslib.pyx in pandas._libs.tslib.array_to_datetime()

pandas\_libs\tslib.pyx in pandas._libs.tslib.array_to_datetime()

pandas\_libs\tslibs\parsing.pyx in pandas._libs.tslibs.parsing.parse_datetime_string()

~\anaconda3\lib\site-packages\dateutil\parser\_parser.py in parse(timestr, parserinfo, **kwargs)
   1372         return parser(parserinfo).parse(timestr, **kwargs)
   1373     else:
-> 1374         return DEFAULTPARSER.parse(timestr, **kwargs)
   1375
   1376

~\anaconda3\lib\site-packages\dateutil\parser\_parser.py in parse(self, timestr, default, ignoretz, tzinfos, **kwargs)
    644                                                   second=0, microsecond=0)
    645
--> 646         res, skipped_tokens = self._parse(timestr, **kwargs)
    647
    648         if res is None:

~\anaconda3\lib\site-packages\dateutil\parser\_parser.py in _parse(self, timestr, dayfirst, yearfirst, fuzzy, fuzzy_with_token
s)
    870             return None, None
    871
--> 872         if not info.validate(res):
    873             return None, None
    874

~\anaconda3\lib\site-packages\dateutil\parser\_parser.py in validate(self, res)
    390
    391         if ((res.tzoffset == 0 and not res.tzname) or
--> 392              (res.tzname == 'Z' or res.tzname == 'z')):
    393             res.tzname = "UTC"
    394             res.tzoffset = 0

KeyboardInterrupt:
```

In [7]:
```
df.dtypes
```

Out[7]:
```
VendorID                 float64
tpep_pickup_datetime      object
tpep_dropoff_datetime     object
passenger_count          float64
trip_distance            float64
RatecodeID               float64
store_and_fwd_flag        object
PULocationID               int64
DOLocationID               int64
payment_type             float64
fare_amount              float64
extra                    float64
mta_tax                  float64
tip_amount               float64
tolls_amount             float64
improvement_surcharge    float64
total_amount             float64
congestion_surcharge     float64
dtype: object
```

In [ ]:
```
df['duration'] = [df['tpep_dropoff_datetime'] - df['tpep_pickup_datetime']].dt.total_seconds()/60
df['duration']
```

In [14]:
```
df = df[['passenger_count','payment_type','fare_amount','trip_distance','tpep_pickup_datetime','tpep_dropoff_datetime']]
df
```

Out[14]:

	passenger_count	payment_type	fare_amount	trip_distance	tpep_pickup_datetime	tpep_dropoff_datetime
0	1.0	1.0	6.00	1.20	01/01/2020 12:28:15 AM	01/01/2020 12:33:03 AM
1	1.0	1.0	7.00	1.20	01/01/2020 12:35:39 AM	01/01/2020 12:43:04 AM
2	1.0	1.0	6.00	0.60	01/01/2020 12:47:41 AM	01/01/2020 12:53:52 AM
3	1.0	1.0	5.50	0.80	01/01/2020 12:55:23 AM	01/01/2020 01:00:14 AM
4	1.0	2.0	3.50	0.00	01/01/2020 12:01:58 AM	01/01/2020 12:04:16 AM
...	...	...	...	...	...	...
24648494	NaN	NaN	32.49	9.22	12/31/2020 11:44:35 PM	01/01/2021 12:01:22 AM
24648495	NaN	NaN	13.22	4.79	12/31/2020 11:41:36 PM	12/31/2020 11:50:32 PM
24648496	NaN	NaN	69.31	28.00	12/31/2020 11:01:17 PM	12/31/2020 11:40:37 PM
24648497	NaN	NaN	35.95	7.08	12/31/2020 11:31:29 PM	12/31/2020 11:44:22 PM
24648498	NaN	NaN	17.09	2.35	12/31/2020 11:12:48 PM	12/31/2020 11:24:51 PM

24648499 rows × 6 columns

In [15]:
```
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
```

In [21]:
```
df['duration'] = [df['tpep_pickup_datetime'] - df['tpep_dropoff_datetime']]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-0edbe26973a8> in <module>
----> 1 df['duration'] = [df['tpep_pickup_datetime'] - df['tpep_dropoff_datetime']]

~\anaconda3\lib\site-packages\pandas\core\frame.py in __setitem__(self, key, value)
   3161         else:
   3162             # set column
-> 3163             self._set_item(key, value)
   3164 
   3165     def _setitem_slice(self, key: slice, value):

~\anaconda3\lib\site-packages\pandas\core\frame.py in _set_item(self, key, value)
   3240         """
   3241         self._ensure_valid_index(value)
-> 3242         value = self._sanitize_column(key, value)
   3243         NDFrame._set_item(self, key, value)
   3244 

~\anaconda3\lib\site-packages\pandas\core\frame.py in _sanitize_column(self, key, value, broadcast)
   3897 
   3898             # turn me into an ndarray
-> 3899             value = sanitize_index(value, self.index)
   3900             if not isinstance(value, (np.ndarray, Index)):
   3901                 if isinstance(value, list) and len(value) > 0:

~\anaconda3\lib\site-packages\pandas\core\internals\construction.py in sanitize_index(data, index)
    749     """
    750     if len(data) != len(index):
--> 751         raise ValueError(
    752             "Length of values "
    753             f"({len(data)}) "

ValueError: Length of values (1) does not match length of index (24648499)
```

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
passenger_count          809568
payment_type             809568
fare_amount                   0
trip_distance                 0
tpep_pickup_datetime          0
tpep_dropoff_datetime         0
dtype: int64
```

In [23]:
```
(809568/len(df) * 100)
```

Out[23]:
```
3.2844515197456854
```

In [24]:
```
df.dropna(inplace = True) #inplace = True will save the operation in the dataframe (just like save button in the excel)
df
```

Out[24]:

	passenger_count	payment_type	fare_amount	trip_distance	tpep_pickup_datetime	tpep_dropoff_datetime
0	1.0	1.0	6.0	1.20	2020-01-01 00:28:15	2020-01-01 00:33:03
1	1.0	1.0	7.0	1.20	2020-01-01 00:35:39	2020-01-01 00:43:04
2	1.0	1.0	6.0	0.60	2020-01-01 00:47:41	2020-01-01 00:53:52
3	1.0	1.0	5.5	0.80	2020-01-01 00:55:23	2020-01-01 01:00:14
4	1.0	2.0	3.5	0.00	2020-01-01 00:01:58	2020-01-01 00:04:16
...	...	...	...	...	...	...
24549234	1.0	2.0	33.0	11.30	2020-12-31 23:05:33	2020-12-31 23:31:36
24549235	1.0	1.0	9.0	2.18	2020-12-31 22:57:20	2020-12-31 23:05:33
24549236	1.0	1.0	9.5	2.52	2020-12-31 23:40:35	2020-12-31 23:48:43
24549237	1.0	1.0	4.5	0.59	2020-12-31 23:54:57	2020-12-31 23:57:39
24549238	1.0	2.0	18.5	6.06	2020-12-31 23:11:16	2020-12-31 23:24:08

23838931 rows × 6 columns

In [28]:
```
df['passenger_count'] = df['passenger_count'].astype('int64')  #astype will convert tye datatype
df['payment_type'] = df['payment_type'].astype('int64')  #astype will convert tye datatype
```

In [26]:
```
df[df.duplicated()]
```

Out[26]:

	passenger_count	payment_type	fare_amount	trip_distance	tpep_pickup_datetime	tpep_dropoff_datetime
39458	1	1	13.5	3.0	2020-01-01 02:29:50	2020-01-01 02:47:07
561748	1	1	4.0	0.4	2020-01-04 11:07:40	2020-01-04 11:10:37
967243	1	1	5.0	0.8	2020-01-06 16:50:12	2020-01-06 16:54:17
1060000	1	1	6.0	1.0	2020-01-07 08:22:23	2020-01-07 08:28:24
1845592	1	1	4.0	0.5	2020-01-10 19:28:12	2020-01-10 19:31:00
...	...	...	...	...	...	...
22084090	1	1	5.5	0.7	2020-11-09 13:24:06	2020-11-09 13:30:09
23388486	1	1	5.5	1.0	2020-12-04 19:32:16	2020-12-04 19:37:11
24232098	1	1	5.5	0.7	2020-12-23 10:49:44	2020-12-23 10:55:31
24240232	1	1	11.0	2.0	2020-12-23 12:34:30	2020-12-23 12:49:18
24520705	1	1	5.0	0.8	2020-12-31 12:21:11	2020-12-31 12:25:31

66 rows × 6 columns

In [29]:
```
df.drop_duplicates(inplace=True)
```

In [30]:
```
df['passenger_count'].value_counts(normalize = True)  #value_counts gives frequency for the categories and how much time it is pr
```

Out[30]:
```
1    0.734563
2    0.140490
3    0.036607
5    0.031533
0    0.020529
6    0.019906
4    0.016364
7    0.000004
8    0.000002
9    0.000002
Name: passenger_count, dtype: float64
```

In [31]:
```
df['payment_type'].value_counts(normalize = True)
```

Out[31]:
```
1    7.325687e-01
2    2.579158e-01
3    6.059978e-03
4    3.454862e-03
5    6.292246e-07
Name: payment_type, dtype: float64
```

In [33]:
```
df=df[df['payment_type']<3]
df=df[(df['passenger_count'] >0) & (df['passenger_count']<6)]
```

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

In [35]:
```
df['payment_type'].replace([1,2],['Card','Cash'],inplace=True)
```

Descriptive statistics (decsribe helps to track outliers)

In [ ]:
```
df.describe()
```

In [36]:
```
df=df[df['passenger_count'] >0]
df=df[df['trip_distance'] >0]
df=df[df['fare_amount'] > 0]
```

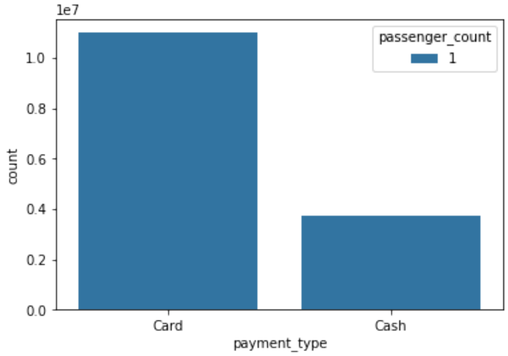
Making countplot to identify general trends in the data

In [77]:
```
ax = sns.countplot(x='payment_type',hue='passenger_count',data = df)  # hue is used for kis basis pr chaie

for bars in ax.containers:
    ax.bar_label(bars)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-77-e37c1b6ce327> in <module>
      2 
      3 for bars in ax.containers:
----> 4     ax.bar_label(bars)

AttributeError: 'AxesSubplot' object has no attribute 'bar_label'
```



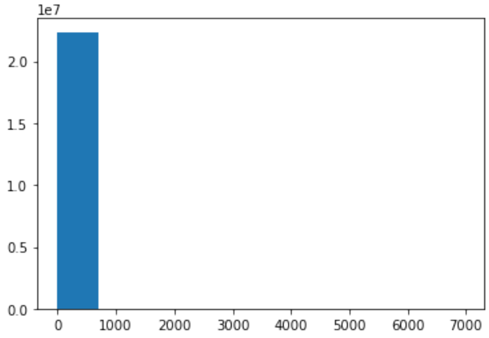
In [ ]:
```
df['month']=df['reservation_date'].dt.month




plt.figure(figsize=(8,4))
ax1=sns.countplot(x='hotel',hue='is_canceled',data=df,palette='Blues')
legend_labels_, = ax1.get_legend_handles_labels()
ax1.legend(bbox_to_anchor(1,1))
plt.title('Reservation hotels in different city',size=20)
plt.xlabel=('No of hotels')
plt.ylabel=(['Not Canceled','Canceled'])
plt.show()
```

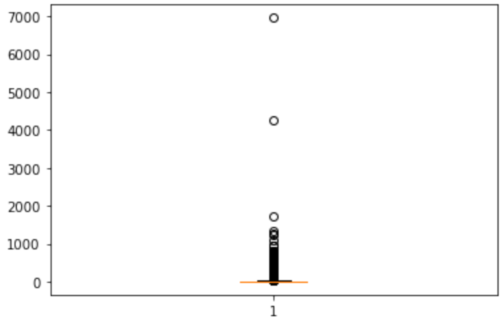
In [39]:
```
# histogram and box plot helps in identifying outliers in this case
plt.hist(df['fare_amount'])
```

Out[39]:
```
(array([2.2365779e+07, 1.8000000e+01, 1.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 1.0000000e+00]),
 array([1.000000e-02, 6.964090e+02, 1.392808e+03, 2.089207e+03,
        2.785606e+03, 3.482005e+03, 4.178404e+03, 4.874803e+03,
        5.571202e+03, 6.267601e+03, 6.964000e+03]),
 <BarContainer object of 10 artists>)
```



```
In [40]: plt.boxplot(df['fare_amount'])
```

```
Out[40]: {'whiskers': [<matplotlib.lines.Line2D at 0x26c93346520>,
           <matplotlib.lines.Line2D at 0x26c93346880>],
          'caps': [<matplotlib.lines.Line2D at 0x26c93346be0>,
           <matplotlib.lines.Line2D at 0x26c93346f40>],
          'boxes': [<matplotlib.lines.Line2D at 0x26c933461c0>],
          'medians': [<matplotlib.lines.Line2D at 0x26c933572e0>],
          'fliers': [<matplotlib.lines.Line2D at 0x26c93357640>],
          'means': []}
```



Removing outlier using Z score or Interquartile range. Here we will use interquartile range

```
In [42]: for col in ['fare_amount','trip_distance','passenger_count']:
             q1 = df[col].quantile(0.25)
             q3 = df[col].quantile(0.75)
             IQR = q3-q1

             lower_bound = q1 - 1.5 * IQR
             upper_bound = q3 + 1.5 * IQR

             df=df[(df[col] >= lower_bound) & (df[col]<= upper_bound) ]
```

```
In [43]: df
```

Out[43]:

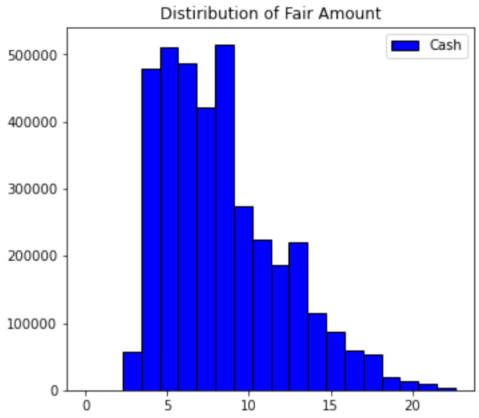
	passenger_count	payment_type	fare_amount	trip_distance	tpep_pickup_datetime	tpep_dropoff_datetime
0	1	Card	6.0	1.20	2020-01-01 00:28:15	2020-01-01 00:33:03
1	1	Card	7.0	1.20	2020-01-01 00:35:39	2020-01-01 00:43:04
2	1	Card	6.0	0.60	2020-01-01 00:47:41	2020-01-01 00:53:52
3	1	Card	5.5	0.80	2020-01-01 00:55:23	2020-01-01 01:00:14
5	1	Cash	2.5	0.03	2020-01-01 00:09:44	2020-01-01 00:10:37
...	...	...	...	...	...	...
24549230	1	Cash	6.0	1.20	2020-12-31 23:08:58	2020-12-31 23:14:28
24549233	1	Card	7.0	1.83	2020-12-31 22:55:17	2020-12-31 23:01:28
24549235	1	Card	9.0	2.18	2020-12-31 22:57:20	2020-12-31 23:05:33
24549236	1	Card	9.5	2.52	2020-12-31 23:40:35	2020-12-31 23:48:43
24549237	1	Card	4.5	0.59	2020-12-31 23:54:57	2020-12-31 23:57:39

14719854 rows × 6 columns

Fare amount and Trip distance analysis by payment mode

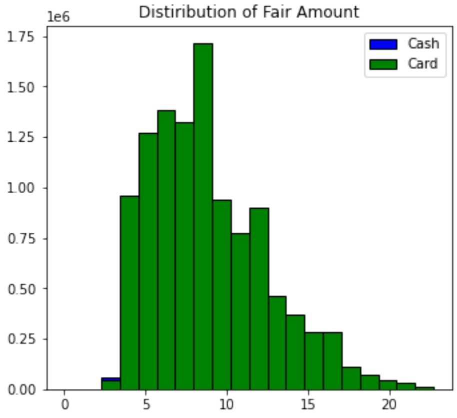
In [54]:
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.title('Distiribution of Fair Amount') # to give title
plt.hist(df[df['payment_type'] == 'Cash']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k', color="blue",label='Cash'
#plt.hist(df[df['payment_type'] == 'Card']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k',color = "green",label='Ca
plt.legend()   # plt.legend shows the label of the hist here cash and card
```

Out[54]:
```
<matplotlib.legend.Legend at 0x26e141d3640>
```

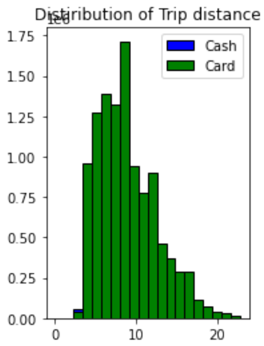


In [55]:
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.title('Distiribution of Fair Amount') # to give title
plt.hist(df[df['payment_type'] == 'Cash']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k', color="blue",label='Cash'
plt.hist(df[df['payment_type'] == 'Card']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k',color = "green",label='Car
plt.legend()   # plt.legend shows the label of the hist here cash and card
```

Out[55]:
```
<matplotlib.legend.Legend at 0x26e141eda90>
```



In [51]:
```
plt.subplot(1,2,1)
plt.title('Distiribution of Trip distance') # to give title
plt.hist(df[df['payment_type'] == 'Cash']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k',color = 'blue',label='Cash
plt.hist(df[df['payment_type'] == 'Card']['fare_amount'],histtype='barstacked',bins=20,edgecolor = 'k',color = 'green',label='Car
plt.legend()   # plt.legend shows the label of the hist here cash and card
plt.show()
```

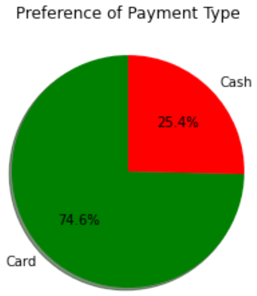


In [57]:
```
df.groupby('payment_type').agg({'fare_amount':['mean','std'],'trip_distance':['mean','std']})
```

Out[57]:

	fare_amount		trip_distance	
	mean	std	mean	std
payment_type				
Card	8.861820	3.575196	1.693476	0.966234
Cash	8.341731	3.628560	1.561408	0.991631

In [60]:
```
plt.title('Preference of Payment Type')
plt.pie(df['payment_type'].value_counts(normalize=True), labels = df['payment_type'].value_counts().index,
        startangle=90,shadow=True,autopct='%1.1f%%',colors=['green','red'])
plt.show()
```



In [65]:
```
passenger_count = df.groupby(['payment_type','passenger_count'])[['passenger_count']].count()
passenger_count.rename(columns = {'passenger_count':'count'},inplace=True)
passenger_count.reset_index(inplace = True)
```

In [66]:
```
passenger_count['perc'] = (passenger_count['count']/passenger_count['count'].sum()) *100
```

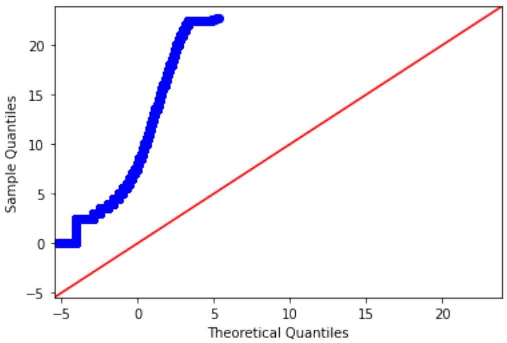
In [67]:
```
passenger_count
```

Out[67]:

	payment_type	passenger_count	count	perc
0	Card	1	10985322	74.629286
1	Cash	1	3734532	25.370714

In [74]:

```
sm.qqplot(df['fare_amount'],line='45')
plt.show()
```



In [69]:

```
card_sample = df[df['payment_type']=='Card']['fare_amount']
cash_sample = df[df['payment_type']=='Cash']['fare_amount']
```

In [75]:

```
t_stats,p_value = st.ttest_ind(a=card_sample, b=cash_sample, equal_var=False)
print('T_statistic',t_stats,'P-value',p_value)
```

```
T_statistic 240.17658261095323 P-value 0.0
```

In [ ]:

```
cancelled_data = df[df['is_cancelled'] ==1]
top_10_country = cancelled_data['Country'].value_counts()[:10]  #:10 sort data in descending count order by cancelled
plt.figure(figsize=(8,8))
plt.title('Top 10 countries with reservation cancelled')
plt.pie(top_10_country, autopct='.2f%',labels=top_10_country.index) #index returns the value based on top_10_country list
plt.show()
```

In [78]:

```
### Z test
age_mean = np.mean(age)
print(age_mean)
```

In [ ]:

```
from statsmodels.stats import weightstats as stests

ztest,pvalue = stests.ztest(data['age'],value=30)
print(float(pvalue))
```

In [ ]:

```
## Anova
from scipy.stats import f_oneway
# Performance when each of the engine
# oil is applied
performance1 = [89, 89, 88, 78, 79]
performance2 = [93, 92, 94, 89, 88]
performance3 = [89, 88, 89, 93, 90]
performance4 = [81, 78, 81, 92, 82]

# Conduct the one-way ANOVA
f_oneway(performance1, performance2, performance3, performance4)
```

In [ ]:

```
### Chi square
from scipy.stats import chi2_contingency

# defining the table
data = [[207, 282, 241], [234, 242, 232]]
stat, p, dof, expected = chi2_contingency(data)

# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```