In [60]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as st
import statsmodels.api as sm
from scipy.stats import linregress
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('yellow_tripdata_2020-01.csv')
df.head()
```

Out[2]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	tri
0	1.0	2020-01-01 00:28:15	2020-01-01 00:33:03	1.0	
1	1.0	2020-01-01 00:35:39	2020-01-01 00:43:04	1.0	
2	1.0	2020-01-01 00:47:41	2020-01-01 00:53:52	1.0	
3	1.0	2020-01-01 00:55:23	2020-01-01 01:00:14	1.0	
4	2.0	2020-01-01 00:01:58	2020-01-01 00:04:16	1.0	

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

Out[3]:
```
(6405008, 18)
```

In [4]:
```
df.dtypes
```

Out[4]:
```
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 [5]:
```
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
```

In [6]:
```
df['duration'] = df['tpep_dropoff_datetime'] - df['tpep_pickup_datetime']
df['duration'] = df['duration'].dt.total_seconds()/60
df
```

Out[6]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_co
0	1.0	2020-01-01 00:28:15	2020-01-01 00:33:03	
1	1.0	2020-01-01 00:35:39	2020-01-01 00:43:04	
2	1.0	2020-01-01 00:47:41	2020-01-01 00:53:52	
3	1.0	2020-01-01 00:55:23	2020-01-01 01:00:14	
4	2.0	2020-01-01 00:01:58	2020-01-01 00:04:16	
...	...	...	...	
6405003	NaN	2020-01-31 22:51:00	2020-01-31 23:22:00	
6405004	NaN	2020-01-31 22:10:00	2020-01-31 23:26:00	
6405005	NaN	2020-01-31 22:50:07	2020-01-31 23:17:57	
6405006	NaN	2020-01-31 22:25:53	2020-01-31 22:48:32	
6405007	NaN	2020-01-31 22:44:00	2020-01-31 23:06:00	

6405008 rows × 19 columns

In [7]:
```
df = df[['passenger_count','payment_type','fare_amount','trip_distance','durat
df
```

Out[7]:

	passenger_count	payment_type	fare_amount	trip_distance	duratio
0	1.0	1.0	6.00	1.20	4.80000(
1	1.0	1.0	7.00	1.20	7.41666
2	1.0	1.0	6.00	0.60	6.18333
3	1.0	1.0	5.50	0.80	4.85000(
4	1.0	2.0	3.50	0.00	2.30000(
...	...	...	...	...	..
6405003	NaN	NaN	17.59	3.24	31.00000(
6405004	NaN	NaN	46.67	22.13	76.00000(
6405005	NaN	NaN	48.85	10.51	27.83333
6405006	NaN	NaN	27.17	5.49	22.65000(
6405007	NaN	NaN	54.56	11.60	22.00000(

6405008 rows × 5 columns

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
passenger_count    65441
payment_type       65441
fare_amount            0
trip_distance          0
duration               0
dtype: int64
```

In [9]:
```
(65441/len(df))*100
```

Out[9]:
```
1.021716132126611
```

In [10]:
```
df.dropna(inplace = True)
```

In [11]:
```
df
```

Out[11]:

	passenger_count	payment_type	fare_amount	trip_distance	duratio
0	1.0	1.0	6.0	1.20	4.80000
1	1.0	1.0	7.0	1.20	7.41666
2	1.0	1.0	6.0	0.60	6.18333
3	1.0	1.0	5.5	0.80	4.85000
4	1.0	2.0	3.5	0.00	2.30000
...	...	...	...	...	...
6339562	1.0	1.0	11.0	2.10	14.23333
6339563	1.0	1.0	13.0	2.13	19.00000
6339564	1.0	1.0	12.5	2.55	16.28333
6339565	1.0	2.0	8.5	1.61	9.63333
6339566	1.0	1.0	0.0	0.00	1.06666

6339567 rows × 5 columns

In [12]:
```
df['passenger_count'] = df['passenger_count'].astype('int64')
df['payment_type'] = df['payment_type'].astype('int64')
```

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

Out[13]:

	passenger_count	payment_type	fare_amount	trip_distance	duratio
2056	1	2	7.0	0.00	0.00000
2441	1	1	52.0	0.00	0.20000
2446	2	1	9.5	1.70	13.06666
2465	1	1	4.0	0.40	3.08333
3344	1	1	6.0	1.20	5.35000
...	...	...	...	...	..
6339558	1	2	8.0	1.63	8.80000
6339559	1	1	8.5	1.81	8.01666
6339560	1	2	6.5	0.98	6.90000
6339562	1	1	11.0	2.10	14.23333
6339565	1	2	8.5	1.61	9.63333

3331706 rows × 5 columns

In [14]:
```
df.drop_duplicates(inplace = True)
```

In [15]:
```
df.shape
```

Out[15]:
```
(3007861, 5)
```

In [16]:
```
df['passenger_count'].value_counts(normalize = True)
```

Out[16]:
```
passenger_count
1    0.581981
2    0.190350
3    0.066360
5    0.062937
6    0.039272
4    0.036046
0    0.023033
7    0.000009
9    0.000006
8    0.000006
Name: proportion, dtype: float64
```

In [17]:
```
df['payment_type'].value_counts(normalize = True)
```

Out[17]:
```
payment_type
1    6.782670e-01
2    3.075731e-01
3    8.721480e-03
4    5.438084e-03
5    3.324622e-07
Name: proportion, dtype: float64
```

In [18]:
```
df = df[(df['passenger_count'] > 0) & (df['passenger_count'] < 6)]
df = df[df['payment_type'] < 3]
```

In [19]:
```
df['payment_type'].replace([1,2],['Card','Cash'], inplace = True)
```

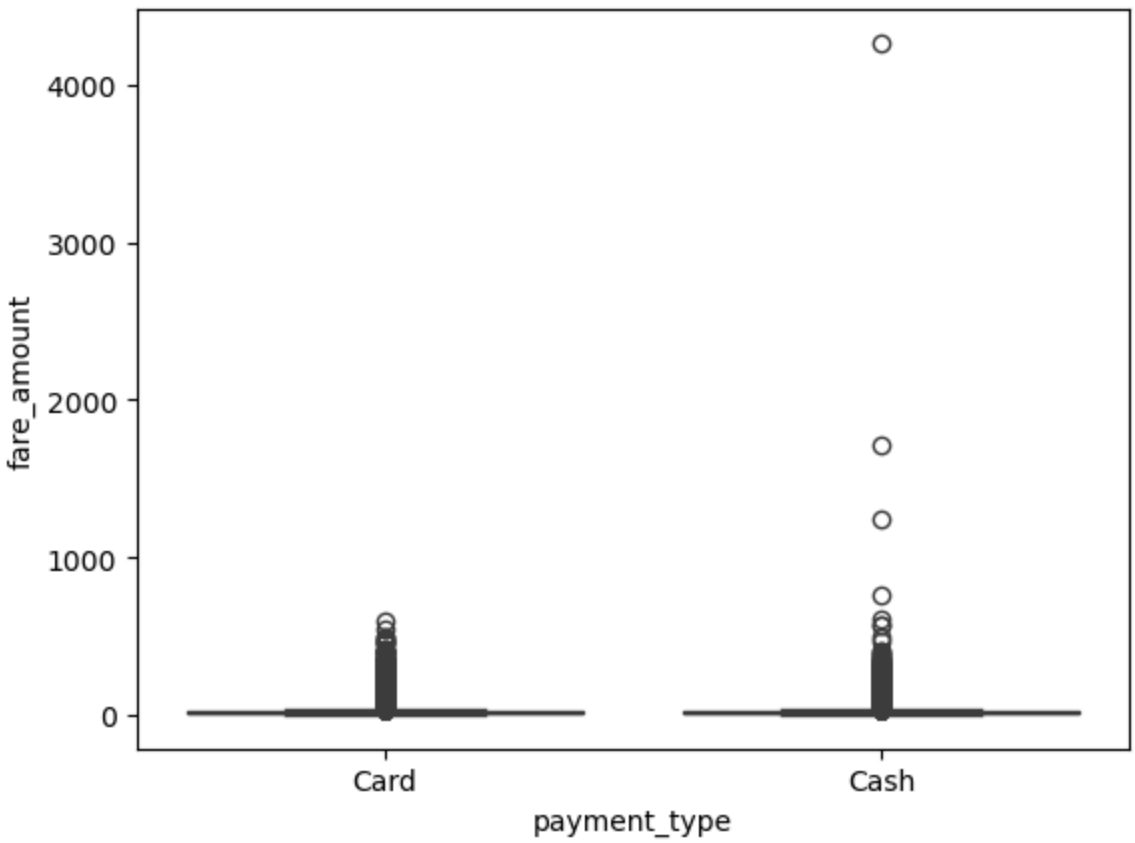
In [20]:
```
df.describe()
```

Out[20]:

	passenger_count	fare_amount	trip_distance	duration
count	2.780283e+06	2.780283e+06	2.780283e+06	2.780283e+06
mean	1.733386e+00	1.780567e+01	4.536729e+00	2.415478e+01
std	1.176652e+00	1.506997e+01	4.895890e+00	9.260031e+01
min	1.000000e+00	-5.000000e+02	-2.218000e+01	-2.770367e+03
25%	1.000000e+00	9.000000e+00	1.500000e+00	9.883333e+00
50%	1.000000e+00	1.300000e+01	2.730000e+00	1.573333e+01
75%	2.000000e+00	2.100000e+01	5.470000e+00	2.336667e+01
max	5.000000e+00	4.265000e+03	2.628800e+02	8.525117e+03

In [21]:
```
df = df[df['fare_amount'] > 0]
df = df[df['trip_distance'] > 0]
df = df[df['duration'] > 0]
```

In [22]:
```
sns.boxplot(data = df, y = 'fare_amount', x = 'payment_type')
plt.show()
```



In [23]:
```
for col in ['fare_amount','trip_distance','duration']:
    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 [24]:
```
df
```

Out[24]:

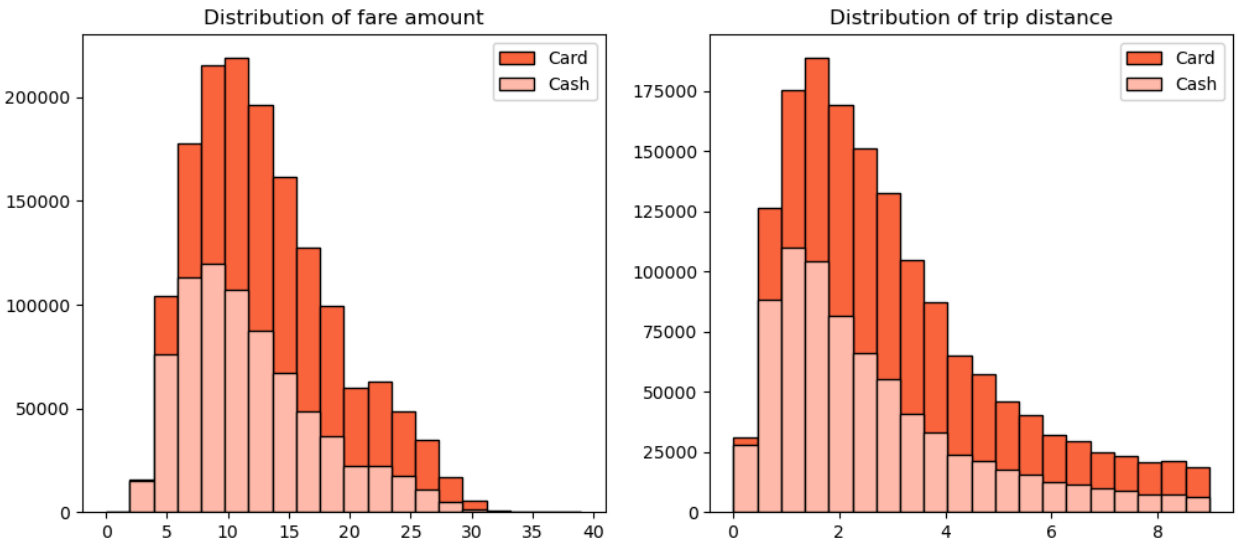
	passenger_count	payment_type	fare_amount	trip_distance	duratio
0	1	Card	6.0	1.20	4.80000
1	1	Card	7.0	1.20	7.41666
2	1	Card	6.0	0.60	6.18333
3	1	Card	5.5	0.80	4.85000
5	1	Cash	2.5	0.03	0.88333
...	...	...	...	...	..
6339550	4	Card	10.5	2.40	12.38333
6339555	3	Card	10.0	2.09	14.80000
6339561	1	Card	17.5	4.11	21.50000
6339563	1	Card	13.0	2.13	19.00000
6339564	1	Card	12.5	2.55	16.28333

2297908 rows × 5 columns

In [26]:

```
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
plt.title('Distribution of fare amount')
plt.hist(df[df['payment_type'] == 'Card']['fare_amount'], histtype = 'barstack
plt.hist(df[df['payment_type'] == 'Cash']['fare_amount'], histtype = 'barstack
plt.legend()

plt.subplot(1,2,2)
plt.title('Distribution of trip distance')
plt.hist(df[df['payment_type'] == 'Card']['trip_distance'], histtype = 'barsta
plt.hist(df[df['payment_type'] == 'Cash']['trip_distance'], histtype = 'barsta
plt.legend()
plt.show()
```

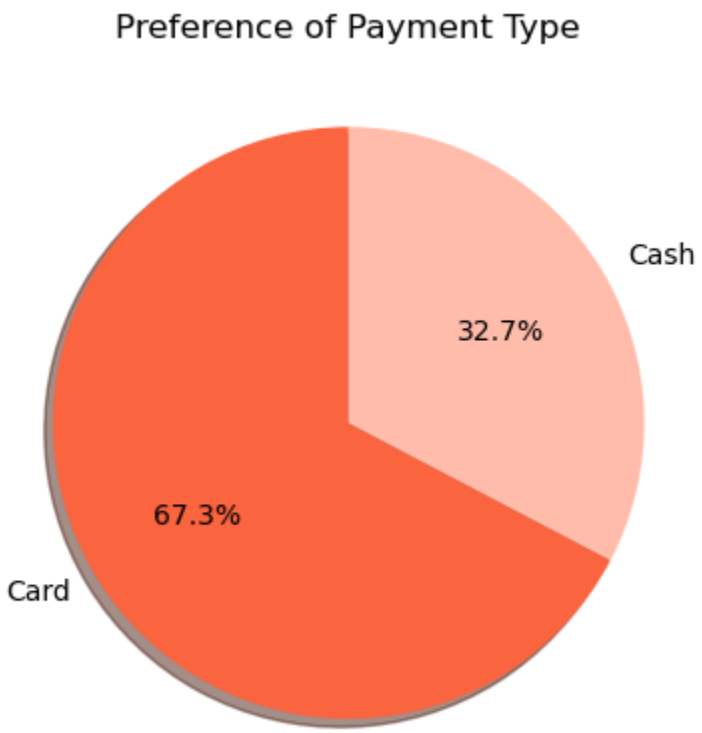


In [27]:
```
df.groupby('payment_type').agg({'fare_amount':['mean','std'], 'trip_distance':
```

Out[27]:

	fare_amount		trip_distance	
	mean	std	mean	std
payment_type				
Card	13.112493	5.849281	2.992237	1.99274
Cash	11.758005	5.613038	2.602207	1.91372

In [29]:
```
plt.title('Preference of Payment Type')
plt.pie(df['payment_type'].value_counts(normalize = True), labels = df['paymer
        startangle = 90, shadow = True, autopct = '%1.1f%%', colors = ['#FA643
plt.show()
```



In [35]:
```
passenger_count = df.groupby(['payment_type', 'passenger_count'])[['passenger_
passenger_count.rename(columns = {'passenger_count':'count'}, inplace = True)
passenger_count.reset_index(inplace = True)
```

In [39]:
```
passenger_count['perc'] = round(passenger_count['count']/passenger_count['cour
```

In [40]:
```
passenger_count
```

Out[40]:

	payment_type	passenger_count	count	perc
0	Card	1	909245	39.57
1	Card	2	327661	14.26
2	Card	3	122412	5.33
3	Card	4	63676	2.77
4	Card	5	124045	5.40
5	Cash	1	460550	20.04
6	Cash	2	155472	6.77
7	Cash	3	54506	2.37
8	Cash	4	32715	1.42
9	Cash	5	47626	2.07

In [43]:

```
df2 = pd.DataFrame(columns = ['payment_type',1,2,3,4,5])
df2['payment_type'] = ['Card','Cash']
df2.iloc[0,1:] = passenger_count.iloc[0:5,-1]
df2.iloc[1,1:] = passenger_count.iloc[5:,-1]
df2
```

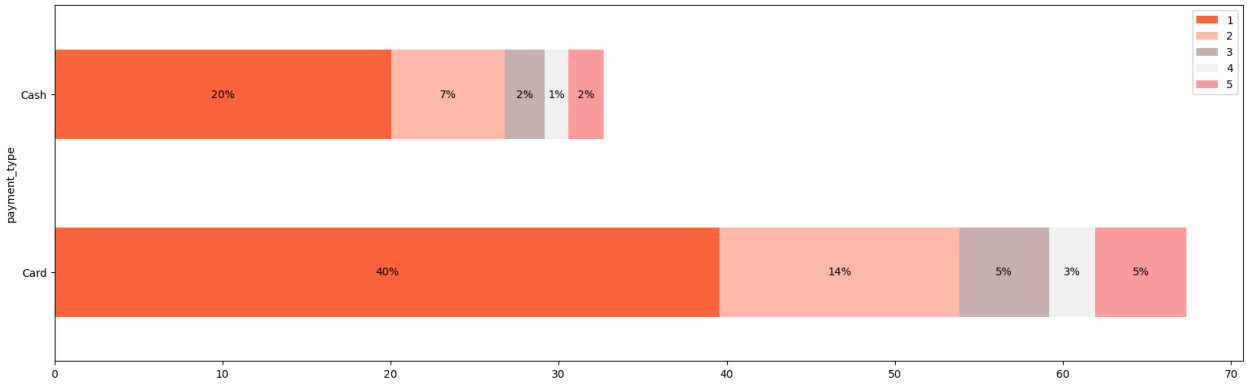
Out[43]:

	payment_type	1	2	3	4	5
0	Card	39.57	14.26	5.33	2.77	5.4
1	Cash	20.04	6.77	2.37	1.42	2.07

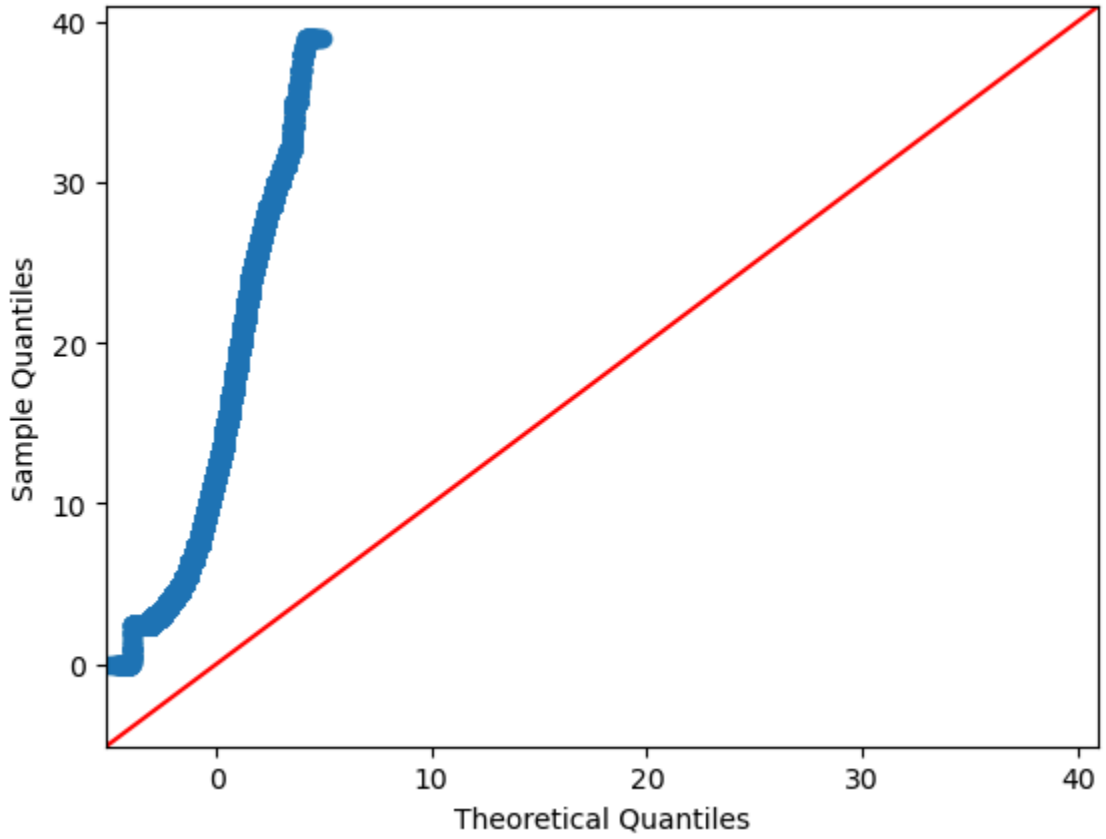
In [48]:

```
fig, ax = plt.subplots(figsize = (20,6))
df2.plot(x = 'payment_type', kind = 'barh', stacked = True, ax = ax ,color = [

for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y  = p.get_xy()
    ax.text(x + width / 2,
            y + height / 2,
            '{:.0f}%'.format(width),
            horizontalalignment = 'center',
            verticalalignment = 'center')
```



In [51]:
```
fig = sm.qqplot(df['fare_amount'], line = '45')
plt.show()
```



In [52]:
```
card_sample = df[df['payment_type'] == 'Card']['fare_amount']
cash_sample = df[df['payment_type'] == 'Cash']['fare_amount']
```

In [53]:
```
t_stats, p_value = st.ttest_ind(a = card_sample, b = cash_sample, equal_var =
print(f'T statistic: {t_stats}, p-value: {p_value}')
```

```
T statistic: 169.2111527245052, p-value: 0.0
```

In [55]:
```
if p_value < 0.05:
    print('Null hypothesis is rejected')
else:
    print('Null hypothesis is accepted')
```

```
Null hypothesis is rejected
```

In [72]:
```
result = linregress(df['duration'],df['fare_amount'])
```

In [73]:
```
print("Slope:", result.slope)
print("Intercept:", result.intercept)
print("R²:", result.rvalue**2)
print("P-value:", result.pvalue)
```

```
Slope: 0.6922301341983748
Intercept: 2.5274733774071585
R²: 0.7601694455031354
P-value: 0.0
```

In [74]:
```
print(f"Fare Amount = {result.slope:.2f} * Duration + {result.intercept:.2f}")
```

```
Fare Amount = 0.69 * Duration + 2.53
```

In [75]:
```
y_line = result.slope * df['duration'] + result.intercept

plt.scatter(df['duration'], df['fare_amount'])
plt.plot(df['duration'], y_line)
plt.xlabel("Duration")
plt.ylabel("Fare Amount")
plt.show()
```