In [2]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split


from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error


from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from threadpoolctl import threadpool_limits
```

In [3]:
```
df = pd.read_csv(r"C:\Users\NASIR KHAN\Downloads\NYC.csv")
```

In [4]:
```
df.head(5)
```

Out[4]:

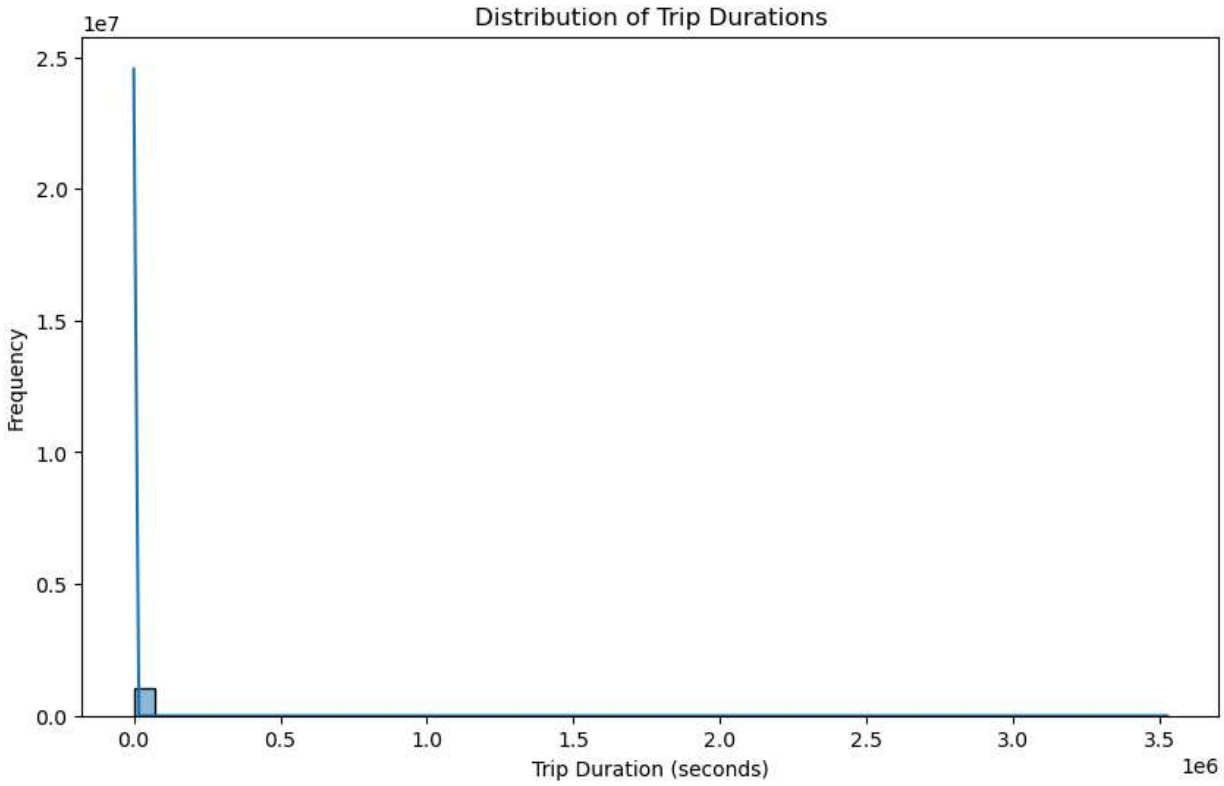
	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	picku
0	id2875421	2	3/14/2016 17:24	3/14/2016 17:32	1	-73.982155	
1	id2377394	1	6/12/2016 0:43	6/12/2016 0:54	1	-73.980415	
2	id3858529	2	1/19/2016 11:35	1/19/2016 12:10	1	-73.979027	
3	id3504673	2	4/6/2016 19:32	4/6/2016 19:39	1	-74.010040	
4	id2181028	2	3/26/2016 13:30	3/26/2016 13:38	1	-73.973053	

Trip Duration Distribution (Histogram):

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

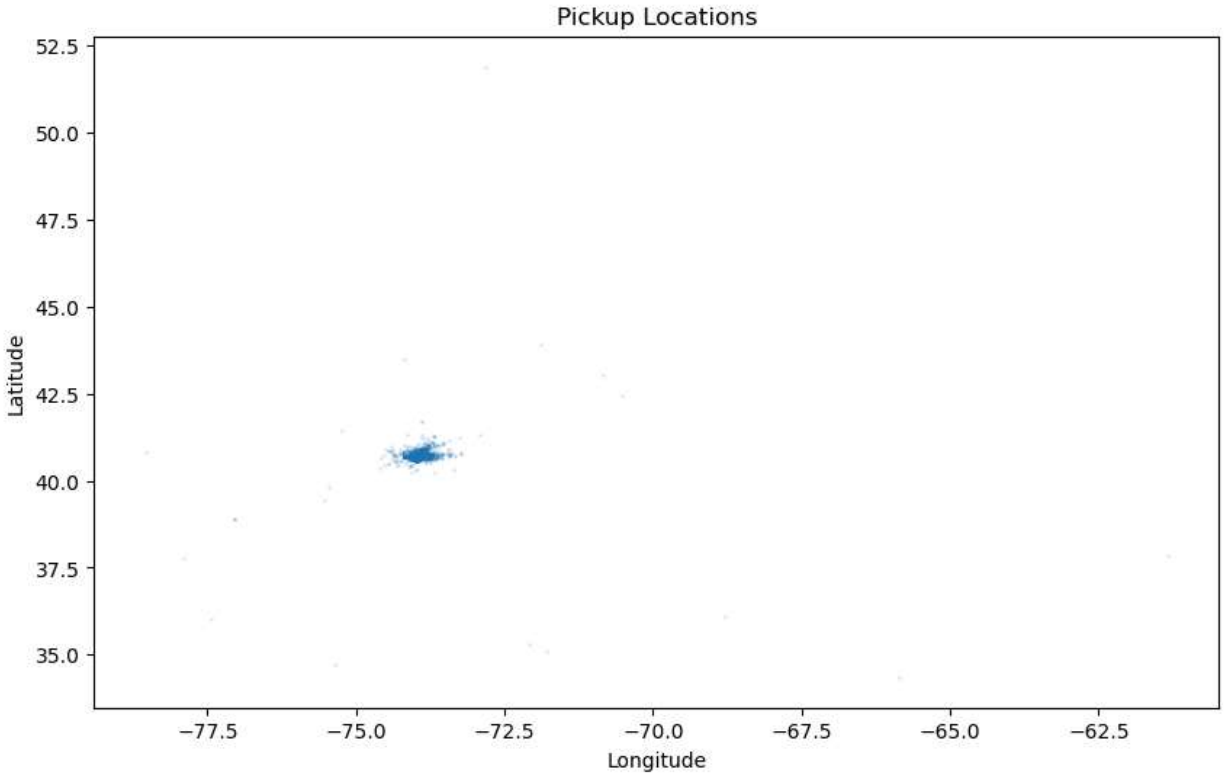
Out[5]:
```
Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration'],
      dtype='object')
```

In [6]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='trip_duration', bins=50, kde=True)
plt.title('Distribution of Trip Durations')
plt.xlabel('Trip Duration (seconds)')
plt.ylabel('Frequency')
plt.show()
```

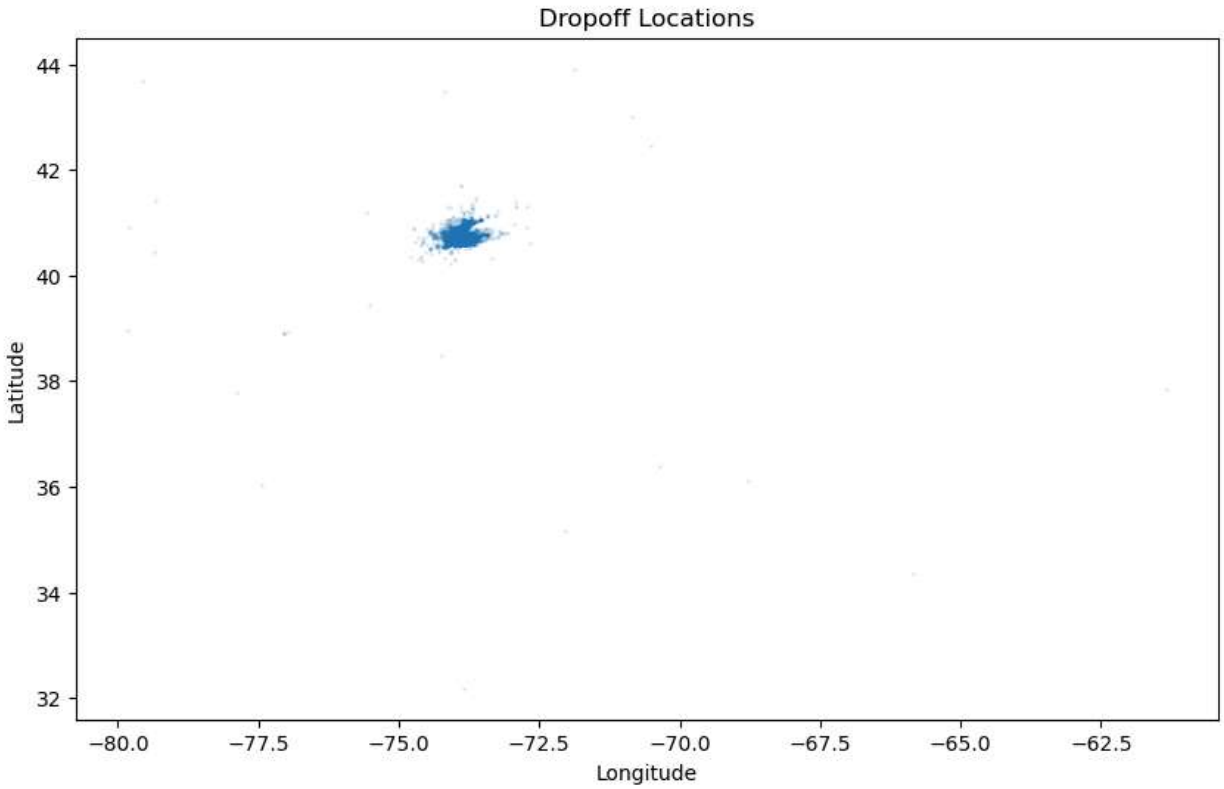


Pickup Locations (Scatter Plot):

In [7]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df['pickup_longitude'], df['pickup_latitude'], alpha=0.1, s=1)
plt.title('Pickup Locations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

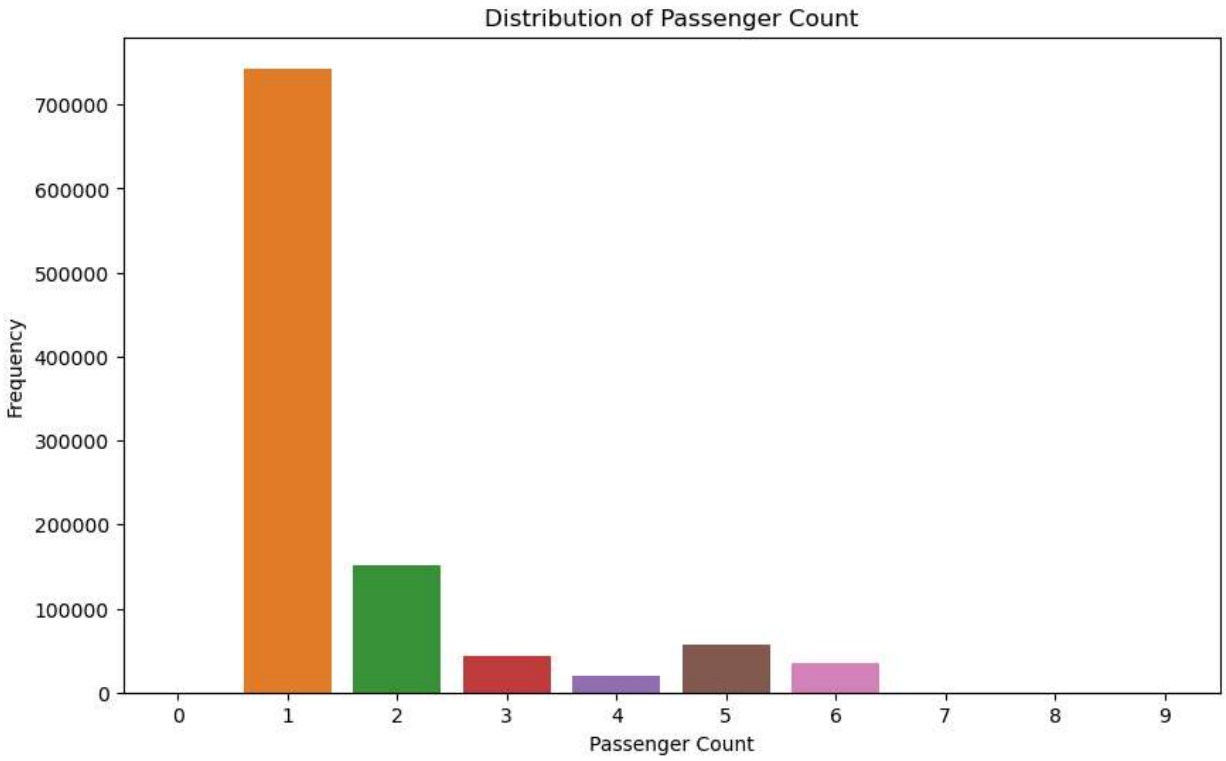


In [8]:
```
# Dropoff Locations
plt.figure(figsize=(10, 6))
plt.scatter(df['dropoff_longitude'], df['dropoff_latitude'], alpha=0.1, s=1)
plt.title('Dropoff Locations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



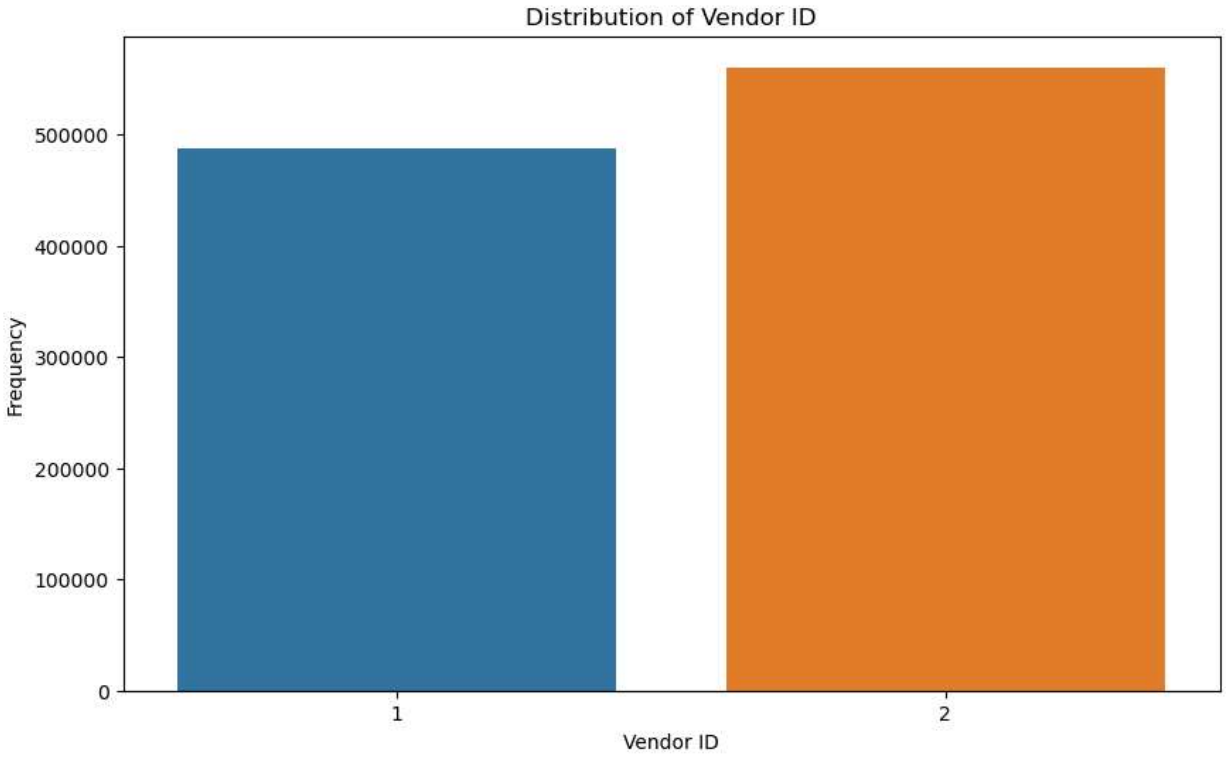
Passenger Count Distribution (Count Plot):

In [9]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='passenger_count')
plt.title('Distribution of Passenger Count')
plt.xlabel('Passenger Count')
plt.ylabel('Frequency')
plt.show()
```



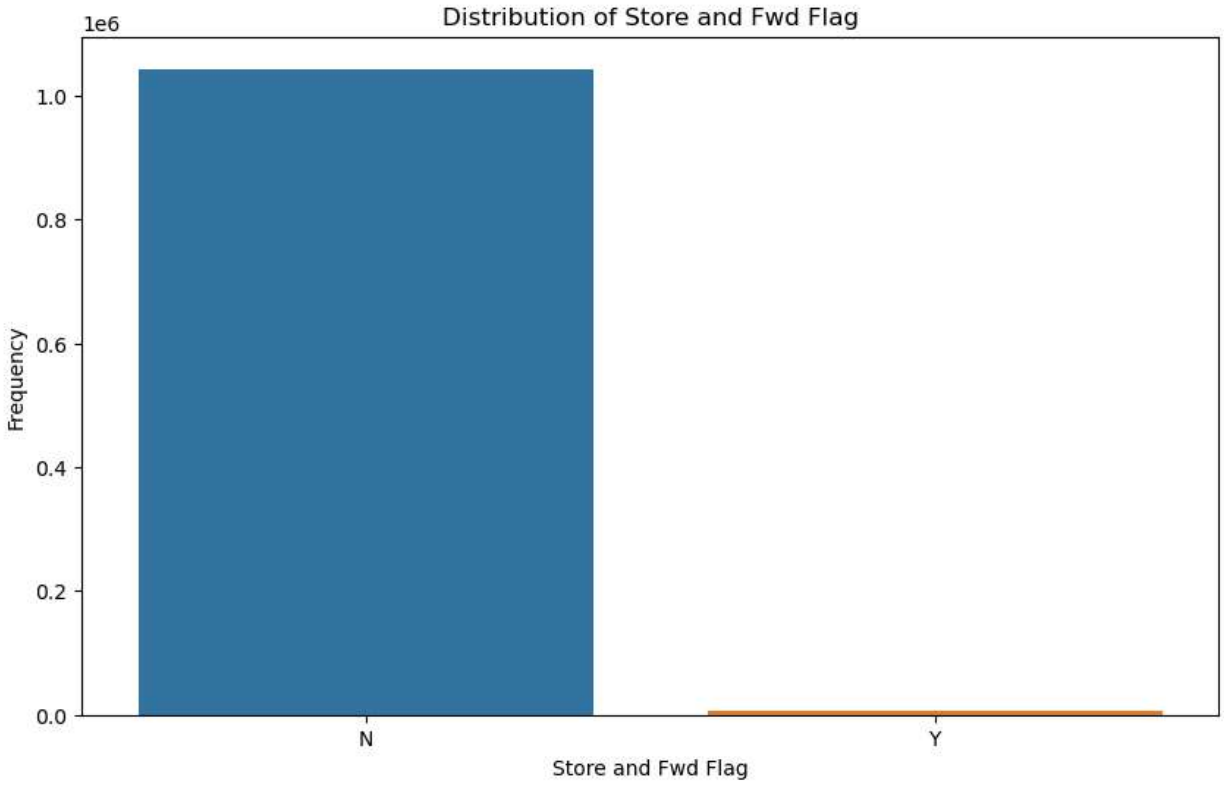
Vendor ID Distribution (Count Plot):

In [10]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='vendor_id')
plt.title('Distribution of Vendor ID')
plt.xlabel('Vendor ID')
plt.ylabel('Frequency')
plt.show()
```



Store and Fwd Flag Distribution (Count Plot):

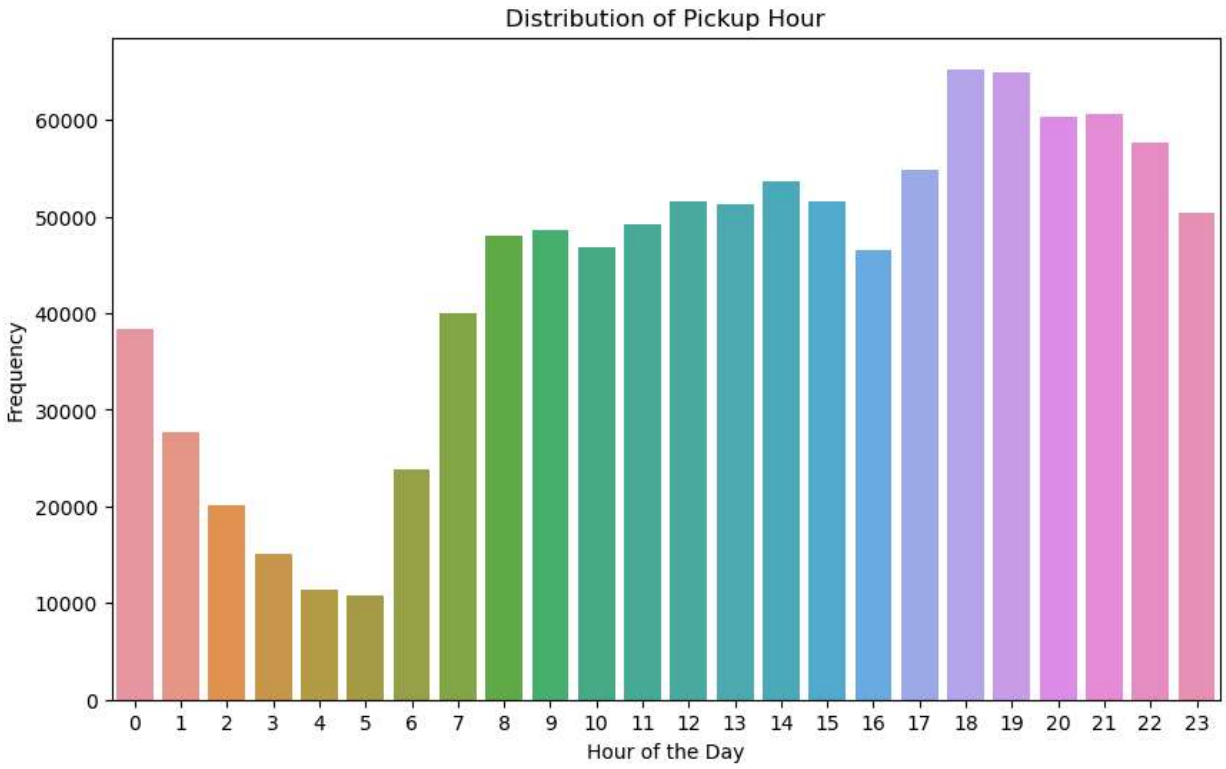
In [11]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='store_and_fwd_flag')
plt.title('Distribution of Store and Fwd Flag')
plt.xlabel('Store and Fwd Flag')
plt.ylabel('Frequency')
plt.show()
```



In [12]:
```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])

# Extract hour from pickup_datetime
df['pickup_hour'] = df['pickup_datetime'].dt.hour

plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='pickup_hour')
plt.title('Distribution of Pickup Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Frequency')
plt.show()
```



In [13]:
```
missing_values = df.isnull().sum()
print(missing_values)
```

```
id                    0
vendor_id             0
pickup_datetime       0
dropoff_datetime      0
passenger_count       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     0
dropoff_latitude      0
store_and_fwd_flag    0
trip_duration         0
pickup_hour           0
dtype: int64
```

In [14]:
```
print(df.columns)
```

```
Index(['id', 'vendor_id', 'pickup_datetime', 'dropoff_datetime',
       'passenger_count', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'store_and_fwd_flag',
       'trip_duration', 'pickup_hour'],
      dtype='object')
```

As there is no missing values in the whole datset so there is no need to perform other operation on data

In [15]:
```
# Convert pickup_datetime and dropoff_datetime to datetime
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
df['dropoff_datetime'] = pd.to_datetime(df['dropoff_datetime'])



# Extract relevant time features
```

```
df['pickup_hour'] = df['pickup_datetime'].dt.hour
df['pickup_day'] = df['pickup_datetime'].dt.day
df['pickup_month'] = df['pickup_datetime'].dt.month
df['dropoff_hour'] = df['dropoff_datetime'].dt.hour
df['dropoff_day'] = df['dropoff_datetime'].dt.day
df['dropoff_month'] = df['dropoff_datetime'].dt.month

# Drop columns that won't be used in the model
df = df.drop(columns=['id', 'pickup_datetime', 'dropoff_datetime', 'store_and_fwd_flag

# Separate features and target variable
X = df.drop(columns=['trip_duration'])
y = df['trip_duration']

# Encode categorical features
X = pd.get_dummies(X, columns=['vendor_id', 'pickup_hour', 'pickup_day', 'pickup_month

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [47]:
```
pip install --upgrade threadpoolctl
```

```
Requirement already satisfied: threadpoolctl in c:\users\nasir khan\anaconda3\lib\sit
e-packages (2.2.0)Note: you may need to restart the kernel to use updated packages.

Collecting threadpoolctl
  Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl
  Attempting uninstall: threadpoolctl
    Found existing installation: threadpoolctl 2.2.0
    Uninstalling threadpoolctl-2.2.0:
      Successfully uninstalled threadpoolctl-2.2.0
Successfully installed threadpoolctl-3.5.0
```

In [16]:
```
# Limit the number of threads used by OpenBLAS, MKL, etc.

import os
os.environ['OPENBLAS_NUM_THREADS'] = '1'
os.environ['MKL_NUM_THREADS'] = '1'
os.environ['NUMEXPR_NUM_THREADS'] = '1'
os.environ['OMP_NUM_THREADS'] = '1'
```

Due to huge dataset the training took many hours to train the model and also take a lot of memory as the dataset contains more then 10 lakhs rows that why i am skipping the training part of that model

In [ ]:
```

```

In [89]:
```
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
mse_dt = mean_squared_error(y_test, y_pred_dt)
print(f"Decision Tree Mean Squared Error: {mse_dt}")
```

```
Decision Tree Mean Squared Error: 15244518.61779558
```

In [ ]:
```

```

In [ ]:

In [ ]:

SVM

In [ ]:
```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error

svr = SVR(kernel='rbf')

svr.fit(X_train, y_train)

y_pred_svr = svr.predict(X_test)

mse_svr = mean_squared_error(y_test, y_pred_svr)
print(f"SVR Mean Squared Error: {mse_svr}")
```

In [ ]: