UBER TRIPS ANALYSIS

Importing libraries

In [56]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Data Preprocessing

In [3]:
```
data=pd.read_csv('uber_data (1).csv')
```

In [4]:
```
data
```

Out[4]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	RatecodeID	store_and_fwd_flag	dropof
0	1	2016-03-01 00:00:00	2016-03-01 00:07:55	1	2.50	-73.976746	40.765152	1	N	-
1	1	2016-03-01 00:00:00	2016-03-01 00:11:06	1	2.90	-73.983482	40.767925	1	N	-
2	2	2016-03-01 00:00:00	2016-03-01 00:31:06	2	19.98	-73.782021	40.644810	1	N	-
3	2	2016-03-01 00:00:00	2016-03-01 00:00:00	3	10.78	-73.863419	40.769814	1	N	-
4	2	2016-03-01 00:00:00	2016-03-01 00:00:00	5	30.43	-73.971741	40.792183	3	N	-
...	...	...	...	...	...	...	...	...	...	
99995	1	2016-03-01 06:17:10	2016-03-01 06:22:15	1	0.50	-73.990898	40.750519	1	N	-
99996	1	2016-03-01 06:17:10	2016-03-01 06:32:41	1	3.40	-74.014488	40.718296	1	N	-
99997	1	2016-03-01 06:17:10	2016-03-01 06:37:23	1	9.70	-73.963379	40.774097	1	N	-
99998	2	2016-03-01 06:17:10	2016-03-01 06:22:09	1	0.92	-73.984901	40.763111	1	N	-
99999	1	2016-03-01 06:17:11	2016-03-01 06:22:00	1	1.00	-73.990685	40.750473	1	N	-

100000 rows × 19 columns

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

Out[5]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	RatecodeID	store_and_fwd_flag	dropoff_lon
0	1	2016-03-01 00:00:00	2016-03-01 00:07:55	1	2.50	-73.976746	40.765152	1	N	-74.0
1	1	2016-03-01 00:00:00	2016-03-01 00:11:06	1	2.90	-73.983482	40.767925	1	N	-74.0
2	2	2016-03-01 00:00:00	2016-03-01 00:31:06	2	19.98	-73.782021	40.644810	1	N	-73.9
3	2	2016-03-01 00:00:00	2016-03-01 00:00:00	3	10.78	-73.863419	40.769814	1	N	-73.9
4	2	2016-03-01 00:00:00	2016-03-01 00:00:00	5	30.43	-73.971741	40.792183	3	N	-74.1

In [6]:
```
data.describe(include='all')
```

Out[6]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	RatecodeID	store_and_fwd_
count	100000.00000	100000	100000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100
unique	NaN	38332	39981	NaN	NaN	NaN	NaN	NaN	
top	NaN	2016-03-10 09:35:06	2016-03-11 00:00:00	NaN	NaN	NaN	NaN	NaN	
freq	NaN	14	18	NaN	NaN	NaN	NaN	NaN	99
mean	1.88327	NaN	NaN	1.929170	3.034270	-73.288983	40.375220	1.040120	N
std	0.32110	NaN	NaN	1.589408	3.846951	7.089652	3.901413	0.284238	N
min	1.00000	NaN	NaN	0.000000	0.000000	-121.933327	0.000000	1.000000	N
25%	2.00000	NaN	NaN	1.000000	0.990000	-73.990959	40.738891	1.000000	N
50%	2.00000	NaN	NaN	1.000000	1.670000	-73.980202	40.755299	1.000000	N
75%	2.00000	NaN	NaN	2.000000	3.200000	-73.964203	40.769021	1.000000	N
max	2.00000	NaN	NaN	6.000000	184.400000	0.000000	41.204548	6.000000	N

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 19 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               100000 non-null  int64
 1   tpep_pickup_datetime   100000 non-null  object
 2   tpep_dropoff_datetime  100000 non-null  object
 3   passenger_count        100000 non-null  int64
 4   trip_distance          100000 non-null  float64
 5   pickup_longitude       100000 non-null  float64
 6   pickup_latitude        100000 non-null  float64
 7   RatecodeID             100000 non-null  int64
 8   store_and_fwd_flag     100000 non-null  object
 9   dropoff_longitude      100000 non-null  float64
 10  dropoff_latitude       100000 non-null  float64
 11  payment_type           100000 non-null  int64
 12  fare_amount            100000 non-null  float64
 13  extra                  100000 non-null  float64
 14  mta_tax                100000 non-null  float64
 15  tip_amount             100000 non-null  float64
 16  tolls_amount           100000 non-null  float64
 17  improvement_surcharge  100000 non-null  float64
 18  total_amount           100000 non-null  float64
dtypes: float64(12), int64(4), object(3)
memory usage: 14.5+ MB
```

In [8]:
```
data['tpep_pickup_datetime']=pd.to_datetime(data['tpep_pickup_datetime'])
data['tpep_dropoff_datetime']=pd.to_datetime(data['tpep_dropoff_datetime'])
```

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 19 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               100000 non-null  int64
 1   tpep_pickup_datetime   100000 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  100000 non-null  datetime64[ns]
 3   passenger_count        100000 non-null  int64
 4   trip_distance          100000 non-null  float64
 5   pickup_longitude       100000 non-null  float64
 6   pickup_latitude        100000 non-null  float64
 7   RatecodeID             100000 non-null  int64
 8   store_and_fwd_flag     100000 non-null  object
 9   dropoff_longitude      100000 non-null  float64
 10  dropoff_latitude       100000 non-null  float64
 11  payment_type           100000 non-null  int64
 12  fare_amount            100000 non-null  float64
 13  extra                  100000 non-null  float64
 14  mta_tax                100000 non-null  float64
 15  tip_amount             100000 non-null  float64
 16  tolls_amount           100000 non-null  float64
 17  improvement_surcharge  100000 non-null  float64
 18  total_amount           100000 non-null  float64
dtypes: datetime64[ns](2), float64(12), int64(4), object(1)
memory usage: 14.5+ MB
```

In [10]:
```
# Extract hour, day, weekday name, and month
data['pickup_hour'] = data['tpep_pickup_datetime'].dt.hour
data['pickup_day'] = data['tpep_pickup_datetime'].dt.day
data['pickup_weekday'] = data['tpep_pickup_datetime'].dt.day_name()
data['pickup_month'] = data['tpep_pickup_datetime'].dt.month
```

In [11]:
```
# Display the first few rows with the new columns
data[['tpep_pickup_datetime', 'pickup_hour', 'pickup_day', 'pickup_weekday', 'pickup_month']].head()
```

Out[11]:

	tpep_pickup_datetime	pickup_hour	pickup_day	pickup_weekday	pickup_month
0	2016-03-01	0	1	Tuesday	3
1	2016-03-01	0	1	Tuesday	3
2	2016-03-01	0	1	Tuesday	3
3	2016-03-01	0	1	Tuesday	3
4	2016-03-01	0	1	Tuesday	3

In [ ]:

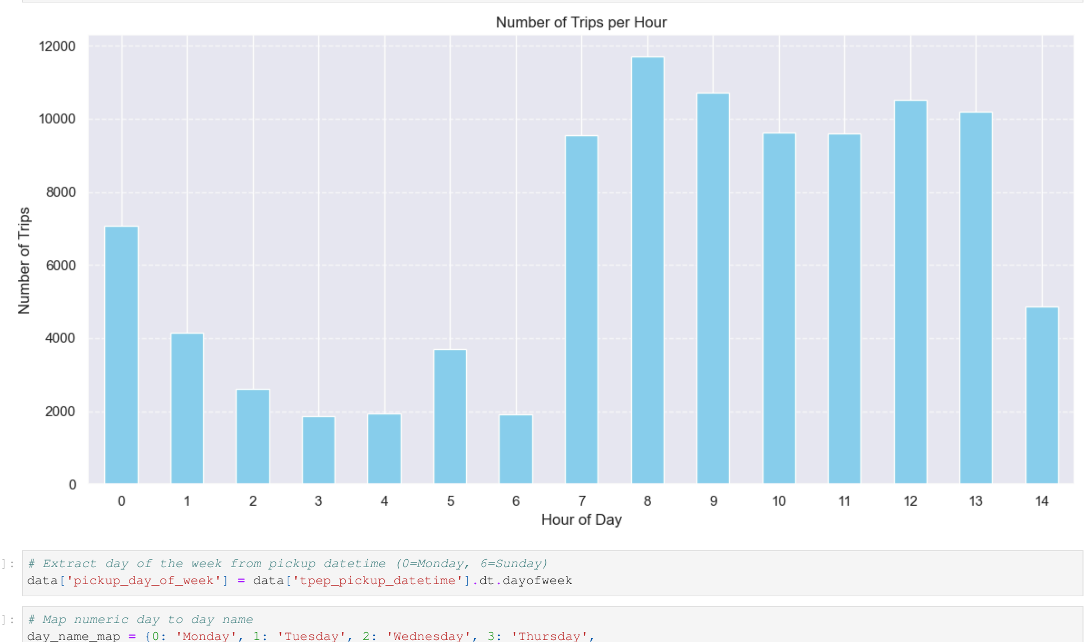
In [13]:
```
import matplotlib.pyplot as plt
```

In [14]:
```
# Convert pickup datetime column to datetime type
data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
```

In [15]:
```
# Extract hour from pickup datetime
data['pickup_hour'] = data['tpep_pickup_datetime'].dt.hour
```

In [16]:
```
# Count number of trips per hour
trips_per_hour = data['pickup_hour'].value_counts().sort_index()
```

In [17]:
```
# Plotting
plt.figure(figsize=(12, 6))
trips_per_hour.plot(kind='bar', color='skyblue')
plt.title('Number of Trips per Hour')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Trips')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

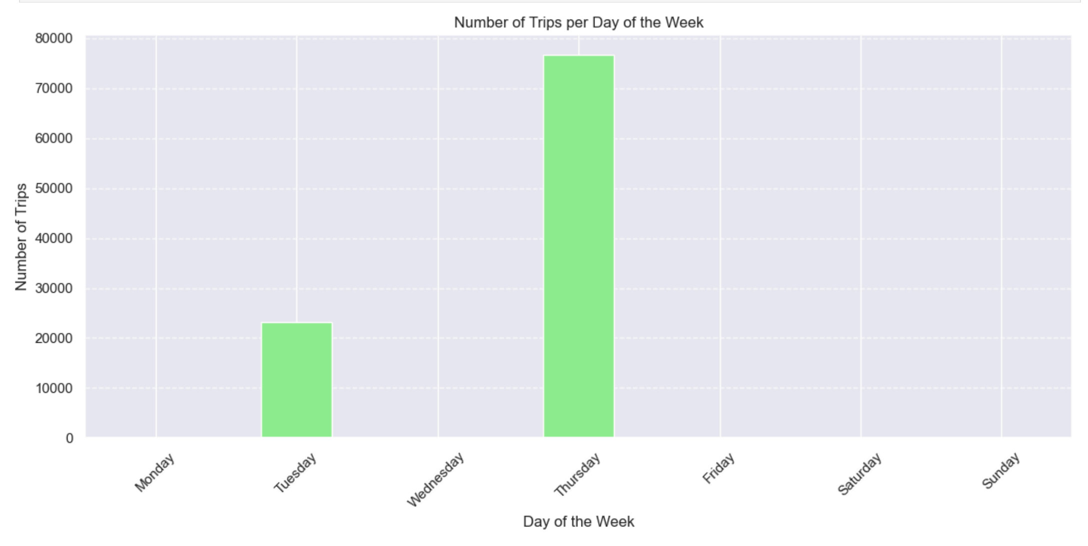


In [18]:
```
# Extract day of the week from pickup datetime (0=Monday, 6=Sunday)
data['pickup_day_of_week'] = data['tpep_pickup_datetime'].dt.dayofweek
```

In [19]:
```
# Map numeric day to day name
day_name_map = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday',
                4: 'Friday', 5: 'Saturday', 6: 'Sunday'}
data['pickup_day_name'] = data['pickup_day_of_week'].map(day_name_map)
```

In [20]:
```
# Count number of trips per day of the week
trips_per_day = data['pickup_day_name'].value_counts().reindex(
    ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
)
```

In [21]:
```
# Plotting
plt.figure(figsize=(12, 6))
trips_per_day.plot(kind='bar', color='lightgreen')
plt.title('Number of Trips per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Trips')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Feature Engineering

In [27]:
```
# Create dummy variables for categorical features
categorical_cols = ['VendorID', 'RatecodeID', 'store_and_fwd_flag', 'payment_type']
data_with_dummies = pd.get_dummies(data, columns=categorical_cols, drop_first=True)

# Display the first few rows of the modified dataframe
data_with_dummies.head()
```

Out[27]:

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	fare_amount	extra
0	2016-03-01	2016-03-01 00:07:55	1	2.50	-73.976746	40.765152	-74.004265	40.746128	9.0	0.5
1	2016-03-01	2016-03-01 00:11:06	1	2.90	-73.983482	40.767925	-74.005943	40.733166	11.0	0.5
2	2016-03-01	2016-03-01 00:31:06	2	19.98	-73.782021	40.644810	-73.974541	40.675770	54.5	0.5
3	2016-03-01	2016-03-01 00:00:00	3	10.78	-73.863419	40.769814	-73.969650	40.757767	31.5	0.0
4	2016-03-01	2016-03-01 00:00:00	5	30.43	-73.971741	40.792183	-74.177170	40.695053	98.0	0.0

5 rows × 31 columns

In [38]:
```
# Drop columns not used as features
columns_to_exclude = [
    'tpep_pickup_datetime', 'tpep_dropoff_datetime',
    'pickup_longitude', 'pickup_latitude',
    'dropoff_longitude', 'dropoff_latitude',
    'total_amount'  # Target variable
]

# Define features (X) and target (y)
X = data_with_dummies.drop(columns=columns_to_exclude)
y = data_with_dummies['total_amount']

# Display the shapes of features and target
X.shape, y.shape
```

Out[38]: ((100000, 24), (100000,))

Define features and target variable

In [60]:
```
features = [
    'VendorID', 'passenger_count', 'trip_distance', 'pickup_longitude',
    'pickup_latitude', 'RatecodeID', 'store_and_fwd_flag', 'dropoff_longitude',
    'dropoff_latitude', 'payment_type', 'fare_amount', 'extra', 'mta_tax',
    'tip_amount', 'tolls_amount', 'improvement_surcharge'
]
target = 'total_amount'

X = data[features]
y = data[target]
```

In [64]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [66]:
```
# Preprocessing: encode categorical features
categorical_features = ['VendorID', 'RatecodeID', 'store_and_fwd_flag', 'payment_type']
numeric_features = list(set(features) - set(categorical_features))

preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ]
)
```

In [68]:
```
# Create the model pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])
```

Split the data into training and testing sets

In [70]:
```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Train a Random Forest Regressor

In [72]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[72]:

```
Pipeline
  preprocessor: ColumnTransformer
    num                 cat
    passthrough         OneHotEncoder
  RandomForestRegressor
```

Predict on the test set

In [73]:
```
# Predictions and evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

Out[73]: (0.28509262500602883, 0.998591839653688)

Visualization of Predictions

In [76]:
```
# Plotting true vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.3, color='blue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Total Amount')
plt.ylabel('Predicted Total Amount')
plt.title('Actual vs Predicted Total Amount')
plt.grid(True)
plt.tight_layout()
plt.show()
```

conclusion

1.(datetime.date(2016, 3, 10), 76780) is the busiest day 2.Late evenings are the popular pickup times. 3.Mean Squared Error (MSE): ~0.107 R² Score: ~0.999 This indicates the model predicts base fares with high accuracy.

In [ ]: