In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

In [2]:
```
df  = pd.read_csv(r"C:\Users\tripa\OneDrive\Desktop\csv uber.csv")
```

Dataset Overview

In [3]:
```
#Overall columns we have got in our dataset
print(df.columns.tolist())
```

```
['Date', 'Time', 'Booking ID', 'Booking Status', 'Customer ID', 'Vehicle Type', 'Pick
up Location', 'Drop Location', 'Avg VTAT', 'Avg CTAT', 'Cancelled Rides by Customer',
'Reason for cancelling by Customer', 'Cancelled Rides by Driver', 'Driver Cancellatio
n Reason', 'Incomplete Rides', 'Incomplete Rides Reason', 'Booking Value', 'Ride Dist
ance', 'Driver Ratings', 'Customer Rating', 'Payment Method']
```

In [4]:
```
# Normalizing column names throughout dataset.
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

print(df.columns.tolist())
```

```
['date', 'time', 'booking_id', 'booking_status', 'customer_id', 'vehicle_type', 'pick
up_location', 'drop_location', 'avg_vtat', 'avg_ctat', 'cancelled_rides_by_customer',
'reason_for_cancelling_by_customer', 'cancelled_rides_by_driver', 'driver_cancellatio
n_reason', 'incomplete_rides', 'incomplete_rides_reason', 'booking_value', 'ride_dist
ance', 'driver_ratings', 'customer_rating', 'payment_method']
```

In [5]:
```
#Shape and column information
print(df.shape)
print(df.info())
df.describe(include="all")
```

```
(150000, 21)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 21 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   date                                150000 non-null  object
 1   time                                150000 non-null  object
 2   booking_id                          150000 non-null  object
 3   booking_status                      150000 non-null  object
 4   customer_id                         150000 non-null  object
 5   vehicle_type                        150000 non-null  object
 6   pickup_location                     150000 non-null  object
 7   drop_location                       150000 non-null  object
 8   avg_vtat                            139500 non-null  float64
 9   avg_ctat                            102000 non-null  float64
 10  cancelled_rides_by_customer         10500 non-null   float64
 11  reason_for_cancelling_by_customer   10500 non-null   object
 12  cancelled_rides_by_driver           27000 non-null   float64
 13  driver_cancellation_reason          27000 non-null   object
 14  incomplete_rides                    9000 non-null    float64
 15  incomplete_rides_reason             9000 non-null    object
 16  booking_value                       102000 non-null  float64
 17  ride_distance                       102000 non-null  float64
 18  driver_ratings                      93000 non-null   float64
 19  customer_rating                     93000 non-null   float64
 20  payment_method                      102000 non-null  object
dtypes: float64(9), object(12)
memory usage: 24.0+ MB
None
```

Out[5]:

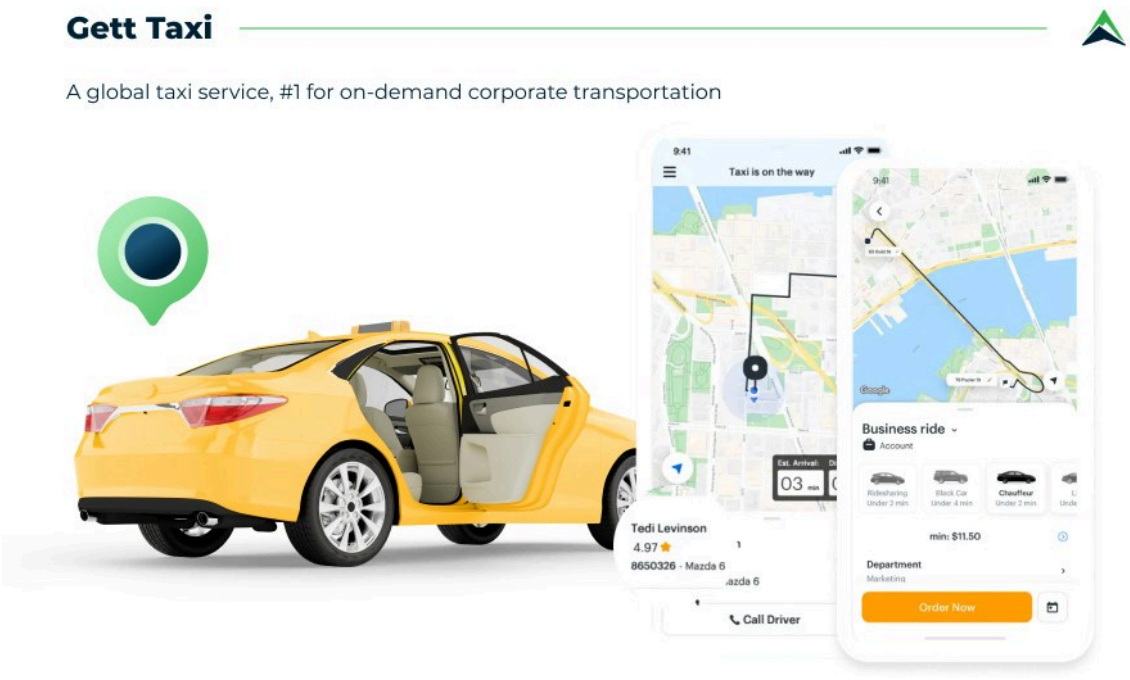
	date	time	booking_id	booking_status	customer_id	vehicle_type	pickup_locatic
count	150000	150000	150000	150000	150000	150000	1500(
unique	365	62910	148767	5	148788	7	1
top	11/16/2024	17:44:57	"CNR7908610"	Completed	"CID4523979"	Auto	Khand
freq	462	16	3	93000	3	37419	9
mean	NaN	NaN	NaN	NaN	NaN	NaN	Na
std	NaN	NaN	NaN	NaN	NaN	NaN	Na
min	NaN	NaN	NaN	NaN	NaN	NaN	Na
25%	NaN	NaN	NaN	NaN	NaN	NaN	Na
50%	NaN	NaN	NaN	NaN	NaN	NaN	Na
75%	NaN	NaN	NaN	NaN	NaN	NaN	Na
max	NaN	NaN	NaN	NaN	NaN	NaN	Na

11 rows × 21 columns

In [6]:
```
#Missing values
df.isna().mean().sort_values(ascending=False) * 100
```

Out[6]:
```
incomplete_rides_reason              94.0
incomplete_rides                     94.0
cancelled_rides_by_customer          93.0
reason_for_cancelling_by_customer    93.0
driver_cancellation_reason           82.0
cancelled_rides_by_driver            82.0
customer_rating                      38.0
driver_ratings                       38.0
ride_distance                        32.0
booking_value                        32.0
payment_method                       32.0
avg_ctat                             32.0
avg_vtat                              7.0
time                                  0.0
drop_location                         0.0
pickup_location                       0.0
vehicle_type                          0.0
customer_id                           0.0
booking_status                        0.0
booking_id                            0.0
date                                  0.0
dtype: float64
```

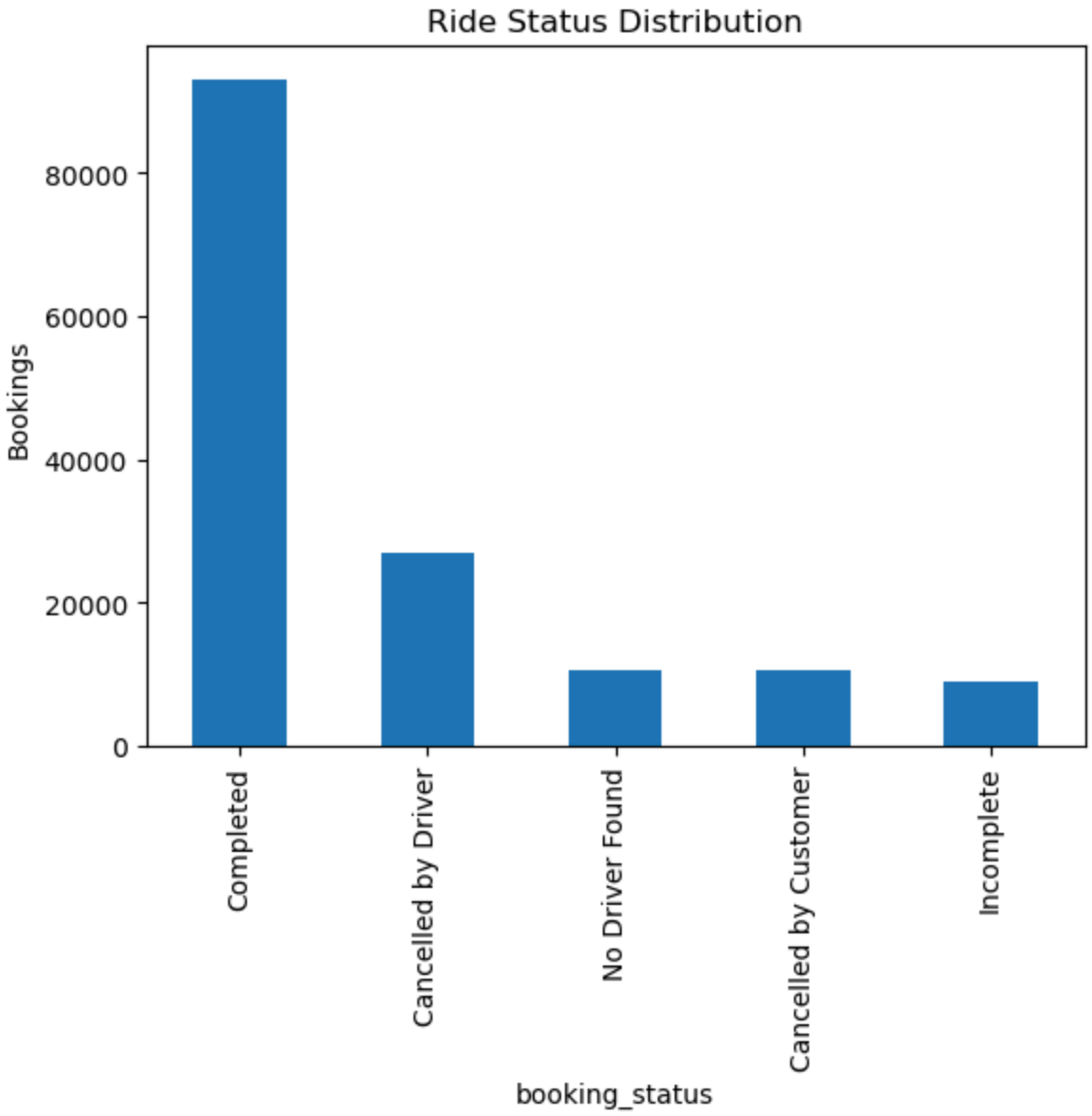
Status of ride distribution



In [7]:
```
# Total counts and percentage share
df['booking_status'].value_counts(normalize=True) * 100
```

Out[7]:
```
booking_status
Completed               62.0
Cancelled by Driver     18.0
No Driver Found          7.0
Cancelled by Customer    7.0
Incomplete               6.0
Name: proportion, dtype: float64
```

In [8]:
```
# Plot of ride status aspects .
df['booking_status'].value_counts().plot(kind='bar')
plt.title("Ride Status Distribution")
plt.ylabel("Bookings")
plt.show()
```

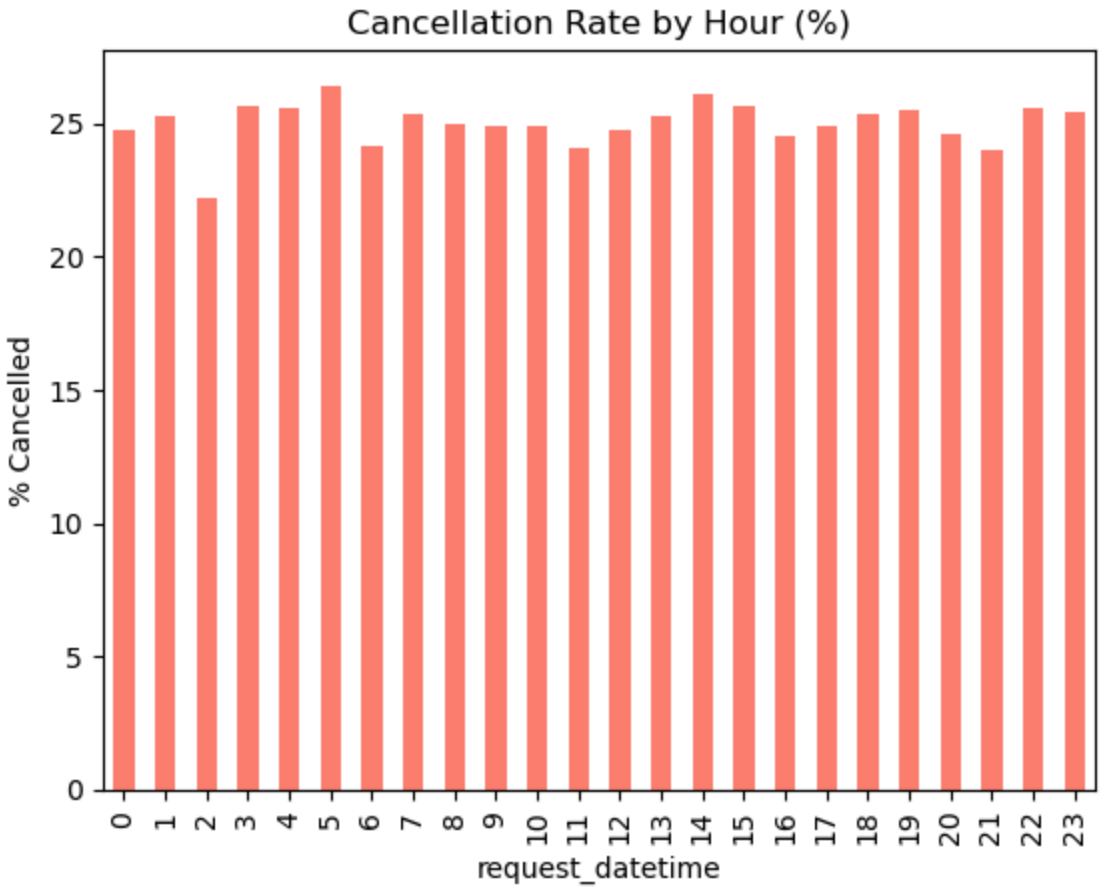


Cancellation Insights

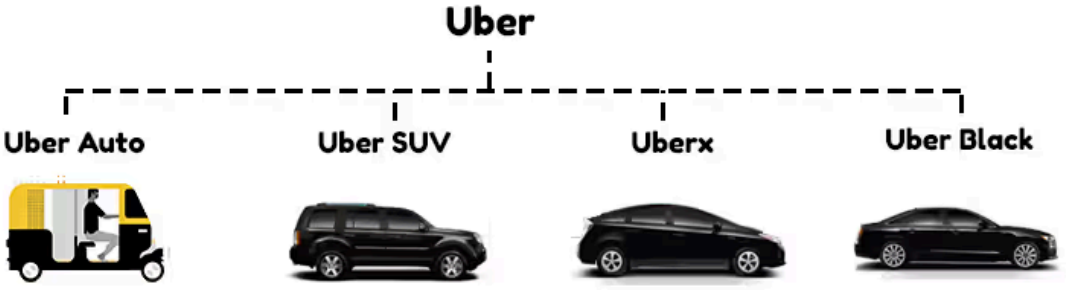
In [9]:
```
#Overall Cancellation rates across uber
cancel_rate = (df['booking_status'].str.contains("cancel", case=False)).mean() * 100
print(f"Cancellation Rate: {cancel_rate:.2f}%")
```

```
Cancellation Rate: 25.00%
```

In [10]:
```python
# Create datetime
df['request_datetime'] = pd.to_datetime(df['date'] + " " + df['time'])
#Graphically representation
df.groupby(df['request_datetime'].dt.hour)['booking_status'].apply(
    lambda x: (x.str.contains("cancel", case=False)).mean() * 100
).plot(kind="bar", color="salmon")
plt.title("Cancellation Rate by Hour (%)")
plt.ylabel("% Cancelled")
plt.show()
```



Vehicle Type and Performance



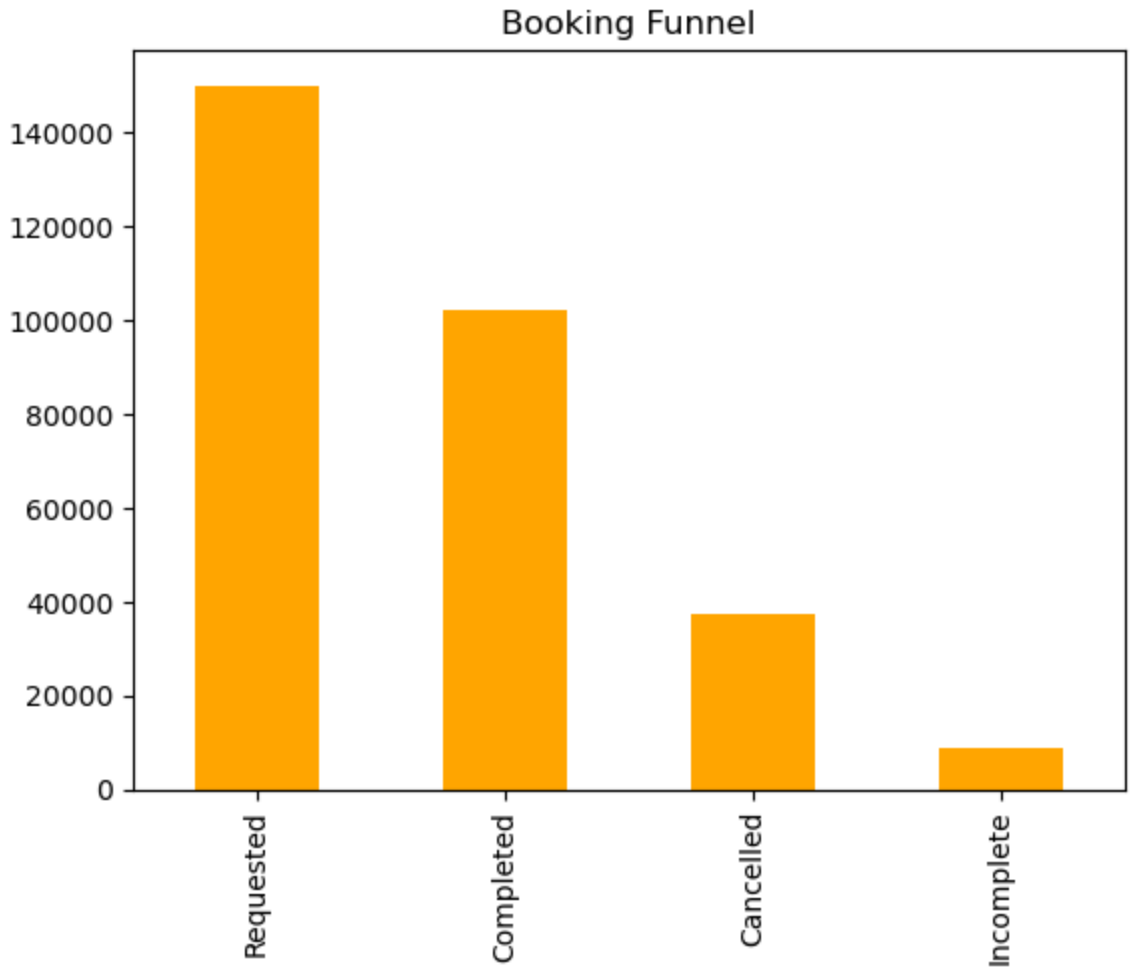
In [11]:
```
#Grouping vehicles and their types based on various aspects.
df.groupby('vehicle_type').agg(
    bookings=('booking_status','size'),
    completed=('booking_status', lambda x: (x.str.contains("complete", case=False)).su
    cancelled=('booking_status', lambda x: (x.str.contains("cancel", case=False)).sum(
    avg_fare=('booking_value','mean'),
    gbv=('booking_value','sum')
).sort_values('bookings', ascending=False)
```

Out[11]:

vehicle_type	bookings	completed	cancelled	avg_fare	gbv
Auto	37419	25415	9323	506.725241	12878422.0
Go Mini	29806	20364	7427	507.684934	10338496.0
Go Sedan	27141	18318	6863	511.503385	9369719.0
Bike	22517	15362	5652	510.200299	7837697.0
Premier Sedan	18111	12315	4516	509.568169	6275332.0
eBike	10557	7181	2630	503.897090	3618485.0
Uber XL	4449	3045	1089	501.816749	1528032.0

Booking Funnel (Flow of trip requests from 'Start' to 'Finish')

In [12]:
```
funnel = {
    "Requested": len(df),
    "Completed": (df['booking_status'].str.contains("complete", case=False)).sum(),
    "Cancelled": (df['booking_status'].str.contains("cancel", case=False)).sum(),
    "Incomplete": (df['booking_status'].str.contains("incomplete", case=False)).sum()
}
pd.Series(funnel).plot(kind="bar", color="orange", title="Booking Funnel")
plt.show()
```



In [13]:
```
# Overall Financial Metrics
print("Total GMV:", df['booking_value'].sum())
print("Average Fare:", df['booking_value'].mean())
print("Average Ride Distance:", df['ride_distance'].mean())
```

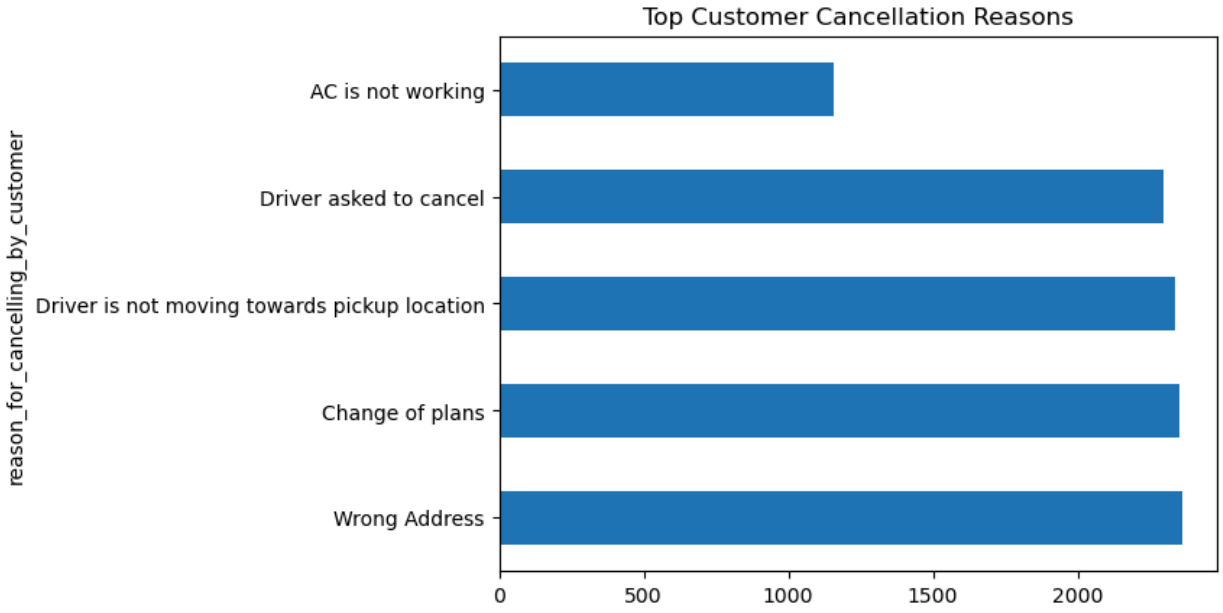
```
Total GMV: 51846183.0
Average Fare: 508.29591176470586
Average Ride Distance: 24.637011666666666
```

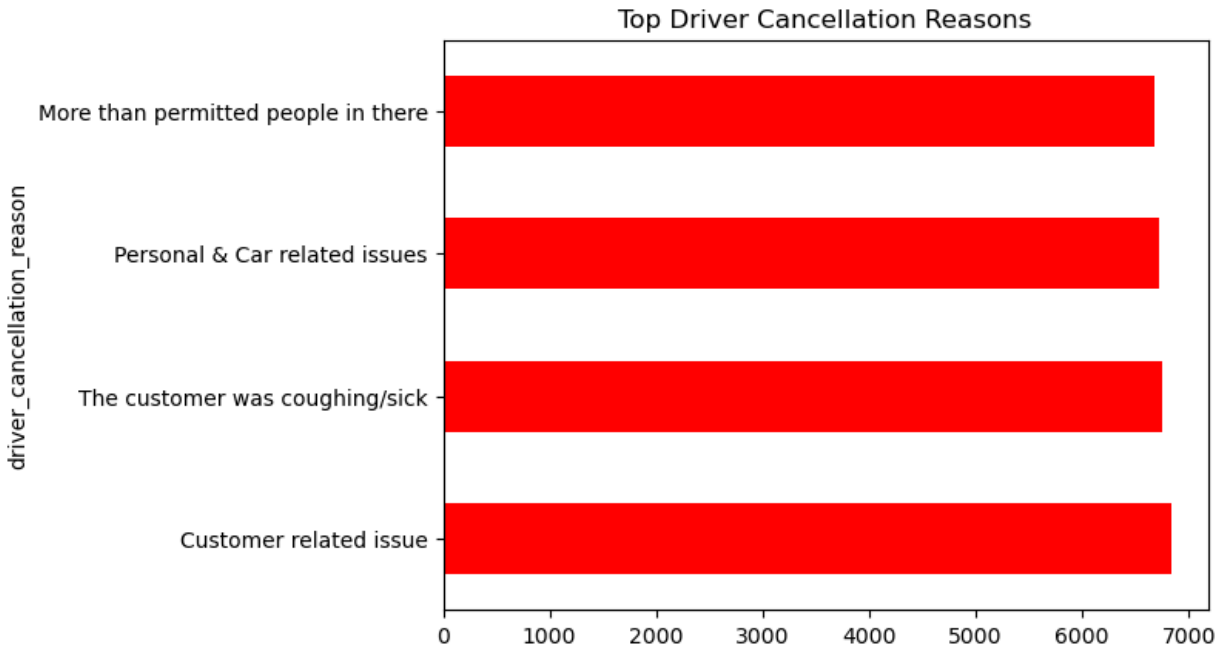
In [14]:
```
# Overall Cancellation Metrics by Customers & Driver
print("Cancelled by Customer:", df['cancelled_rides_by_customer'].sum())
print("Cancelled by Driver:", df['cancelled_rides_by_driver'].sum())
```

```
Cancelled by Customer: 10500.0
Cancelled by Driver: 27000.0
```

In [15]:
```
# Reasons for Cancellation
df['reason_for_cancelling_by_customer'].value_counts().head(10).plot(kind="barh")
plt.title("Top Customer Cancellation Reasons")
plt.show()

df['driver_cancellation_reason'].value_counts().head(10).plot(kind="barh", color="red"
plt.title("Top Driver Cancellation Reasons")
plt.show()
```



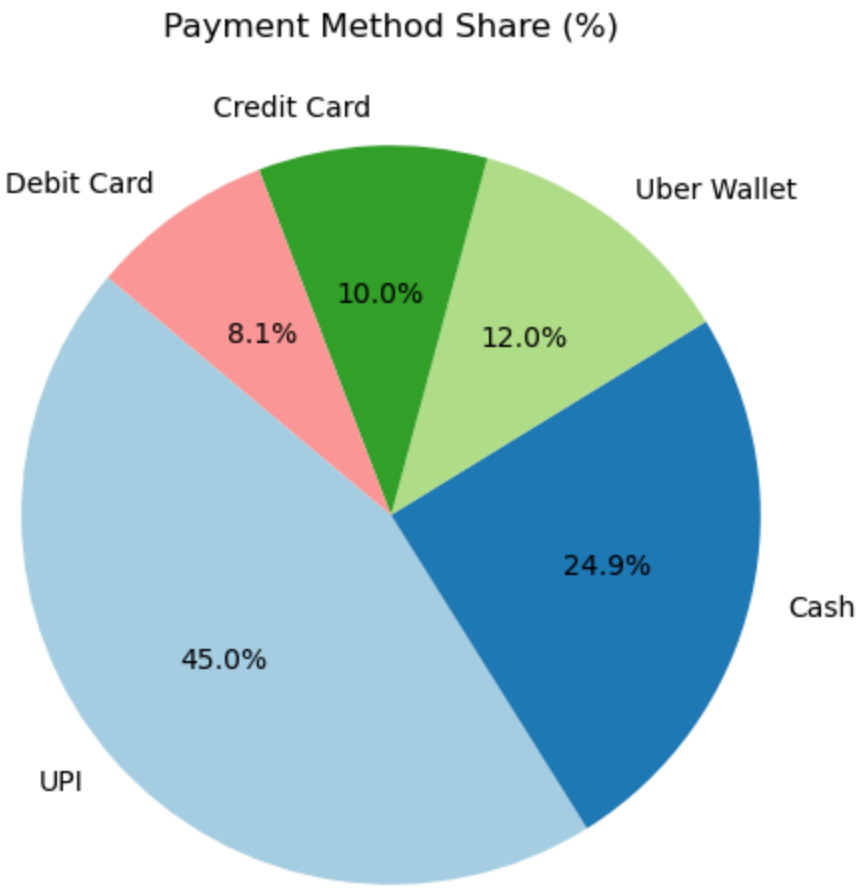


In [16]:
```
#Payment Methods share
df['payment_method'].value_counts(normalize=True) * 100
```

Out[16]:
```
payment_method
UPI            45.008824
Cash           24.869608
Uber Wallet    12.035294
Credit Card    10.008824
Debit Card      8.077451
Name: proportion, dtype: float64
```

In [17]:
```
# Representation
payment_share = df['payment_method'].value_counts(normalize=True) * 100

plt.figure(figsize=(6,6))
plt.pie(
    payment_share,
    labels=payment_share.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=plt.cm.Paired.colors
)
plt.title("Payment Method Share (%)")
plt.show()
```



In [ ]: