In [28]:
```
print("What's up data nerds!")
```

```
What's up data nerds!
```

In [1]:
```
import pandas as pd
```

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

In [3]:
```
df.head ()
```

Out[3]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Drop Location
0	2024-03-23	12:29:38	"CNR5884300"	No Driver Found	"CID1982111"	eBike	Palam Vihar	Jhilmil
1	2024-11-29	18:01:39	"CNR1326809"	Incomplete	"CID4604802"	Go Sedan	Shastri Nagar	Gurgaon Sector 56
2	2024-08-23	08:56:10	"CNR8494506"	Completed	"CID9202816"	Auto	Khandsa	Malviya Nagar
3	2024-10-21	17:17:25	"CNR8906825"	Completed	"CID2610914"	Premier Sedan	Central Secretariat	Inderlok
4	2024-09-16	22:08:00	"CNR1950162"	Completed	"CID9933542"	Bike	Ghitorni Village	Khan Market

5 rows × 21 columns

In [4]:
```
df.isnull ().sum()   # Nulls per column
```

Out[4]:
```
Date                                  0
Time                                  0
Booking ID                            0
Booking Status                        0
Customer ID                           0
Vehicle Type                          0
Pickup Location                       0
Drop Location                         0
Avg VTAT                          10500
Avg CTAT                          48000
Cancelled Rides by Customer      139500
Reason for cancelling by Customer    139500
Cancelled Rides by Driver        123000
Driver Cancellation Reason       123000
Incomplete Rides                 141000
Incomplete Rides Reason          141000
Booking Value                     48000
Ride Distance                     48000
Driver Ratings                    57000
Customer Rating                   57000
Payment Method                    48000
dtype: int64
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Date', 'Time', 'Booking ID', 'Booking Status', 'Customer ID',
       'Vehicle Type', 'Pickup Location', 'Drop Location', 'Avg VTAT',
       'Avg CTAT', 'Cancelled Rides by Customer',
       'Reason for cancelling by Customer', 'Cancelled Rides by Driver',
       'Driver Cancellation Reason', 'Incomplete Rides',
       'Incomplete Rides Reason', 'Booking Value', 'Ride Distance',
       'Driver Ratings', 'Customer Rating', 'Payment Method'],
      dtype='object')
```

In [7]:
```
import pandas as pd

df = pd.read_csv("C:\\Users\\abdir\\Desktop\\ncr_ride_bookings.csv")
   # <-- this creates df
df.head(10)   # now it works
```

Out[7]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Dro Locatio
0	2024-03-23	12:29:38	"CNR5884300"	No Driver Found	"CID1982111"	eBike	Palam Vihar	Jhilm
1	2024-11-29	18:01:39	"CNR1326809"	Incomplete	"CID4604802"	Go Sedan	Shastri Nagar	Gurgao Sector 5
2	2024-08-23	08:56:10	"CNR8494506"	Completed	"CID9202816"	Auto	Khandsa	Malviy Naga
3	2024-10-21	17:17:25	"CNR8906825"	Completed	"CID2610914"	Premier Sedan	Central Secretariat	Inderlo
4	2024-09-16	22:08:00	"CNR1950162"	Completed	"CID9933542"	Bike	Ghitorni Village	Kha Marke
5	2024-02-06	09:44:56	"CNR4096693"	Completed	"CID4670564"	Auto	AIIMS	Narsinghpu
6	2024-06-17	15:45:58	"CNR2002539"	Completed	"CID6800553"	Go Mini	Vaishali	Punjab Bag
7	2024-03-19	17:37:37	"CNR6568000"	Completed	"CID8610436"	Auto	Mayur Vihar	Cyber Hu
8	2024-09-14	12:49:09	"CNR4510807"	No Driver Found	"CID7873618"	Go Sedan	Noida Sector 62	Noid Sector 1
9	2024-12-16	19:06:48	"CNR7721892"	Incomplete	"CID5214275"	Auto	Rohini	Adars Naga

10 rows × 21 columns

In [8]:
```
# Convert Date and Time into one datetime column
df["datetime"] = pd.to_datetime(df["Date"] + " " + df["Time"], errors="coerce")

# Extract useful features
df["year"] = df["datetime"].dt.year
df["month"] = df["datetime"].dt.month
df["day"] = df["datetime"].dt.day
df["day_of_week"] = df["datetime"].dt.day_name()
df["hour"] = df["datetime"].dt.hour
```

In [10]:
```
# Remove extra quotes
df["Booking ID"] = df["Booking ID"].str.replace('"', '')
df["Customer ID"] = df["Customer ID"].str.replace('"', '')
```

In [11]:
```
# Drop high-null columns
drop_cols = [
    "Cancelled Rides by Customer", "Reason for cancelling by Customer",
```

```
    "Cancelled Rides by Driver", "Driver Cancellation Reason",
    "Incomplete Rides", "Incomplete Rides Reason"
]
df.drop(columns=drop_cols, inplace=True)

# Fill numeric columns with median
for col in ["Avg VTAT", "Avg CTAT", "Booking Value", "Ride Distance", "Driver Ratin
    df[col].fillna(df[col].median(), inplace=True)

# Fill categorical columns with mode
for col in ["Payment Method", "Vehicle Type", "Pickup Location", "Drop Location"]:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

```
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
```

```
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:11: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].median(), inplace=True)
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\2665887149.py:15: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].mode()[0], inplace=True)
```

In [12]:
```
df.drop_duplicates(subset="Booking ID", inplace=True)
```

In [13]:
```
df.info()
df.isnull().sum()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 148767 entries, 0 to 149999
Data columns (total 21 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Date             148767 non-null  object
 1   Time             148767 non-null  object
 2   Booking ID       148767 non-null  object
 3   Booking Status   148767 non-null  object
 4   Customer ID      148767 non-null  object
 5   Vehicle Type     148767 non-null  object
 6   Pickup Location  148767 non-null  object
 7   Drop Location    148767 non-null  object
 8   Avg VTAT         148767 non-null  float64
 9   Avg CTAT         148767 non-null  float64
 10  Booking Value    148767 non-null  float64
 11  Ride Distance    148767 non-null  float64
 12  Driver Ratings   148767 non-null  float64
 13  Customer Rating  148767 non-null  float64
 14  Payment Method   148767 non-null  object
 15  datetime         148767 non-null  datetime64[ns]
 16  year             148767 non-null  int32
 17  month            148767 non-null  int32
 18  day              148767 non-null  int32
 19  day_of_week      148767 non-null  object
 20  hour             148767 non-null  int32
dtypes: datetime64[ns](1), float64(6), int32(4), object(10)
memory usage: 22.7+ MB
```

Out[13]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Drop Location	Av VTA
0	2024-03-23	12:29:38	CNR5884300	No Driver Found	CID1982111	eBike	Palam Vihar	Jhilmil	8
1	2024-11-29	18:01:39	CNR1326809	Incomplete	CID4604802	Go Sedan	Shastri Nagar	Gurgaon Sector 56	4
2	2024-08-23	08:56:10	CNR8494506	Completed	CID9202816	Auto	Khandsa	Malviya Nagar	13
3	2024-10-21	17:17:25	CNR8906825	Completed	CID2610914	Premier Sedan	Central Secretariat	Inderlok	13
4	2024-09-16	22:08:00	CNR1950162	Completed	CID9933542	Bike	Ghitorni Village	Khan Market	5

5 rows × 21 columns

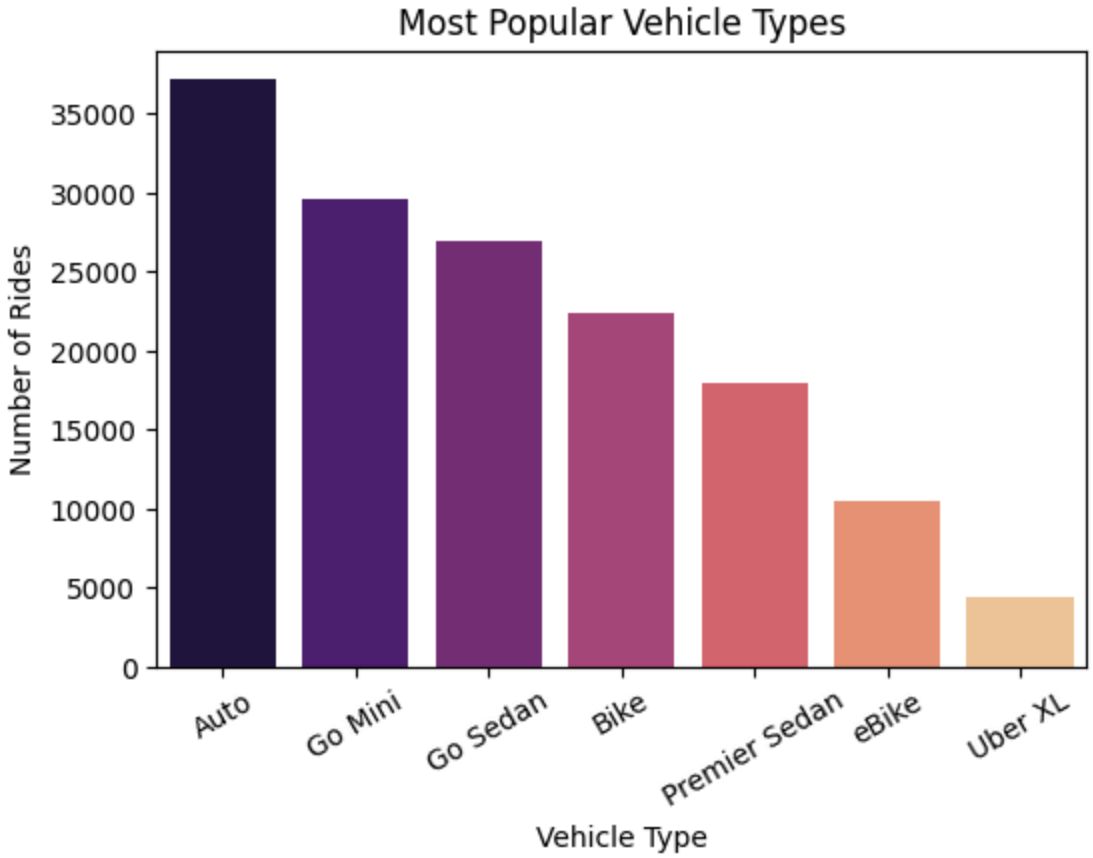
In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Booking status distribution
status_counts = df["Booking Status"].value_counts()

plt.figure(figsize=(6,4))
sns.barplot(x=status_counts.index, y=status_counts.values, palette="viridis")
plt.title("Booking Status Distribution")
plt.ylabel("Number of Bookings")
plt.xticks(rotation=30)
plt.show()
```

```
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\302591104.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=status_counts.index, y=status_counts.values, palette="viridis")
```

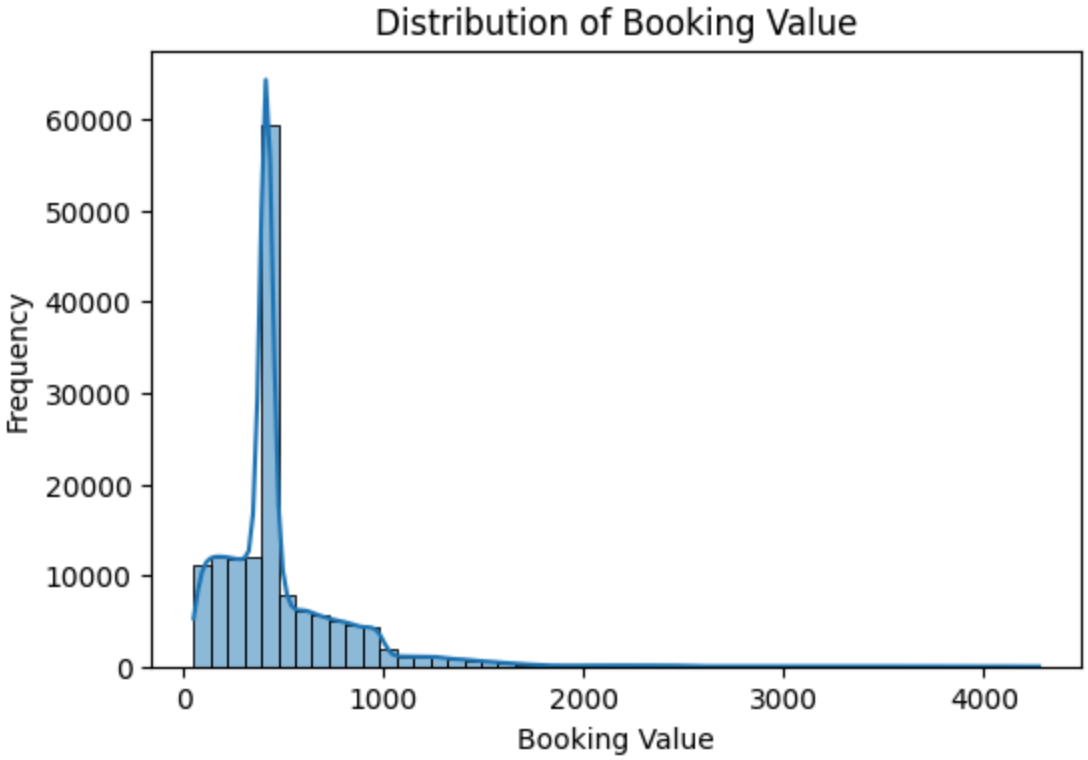
In [15]:
```
vehicle_counts = df["Vehicle Type"].value_counts()

plt.figure(figsize=(6,4))
sns.barplot(x=vehicle_counts.index, y=vehicle_counts.values, palette="magma")
plt.title("Most Popular Vehicle Types")
plt.ylabel("Number of Rides")
plt.xticks(rotation=30)
plt.show()
```

```
C:\Users\abdir\AppData\Local\Temp\ipykernel_7248\665332025.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=vehicle_counts.index, y=vehicle_counts.values, palette="magma")
```

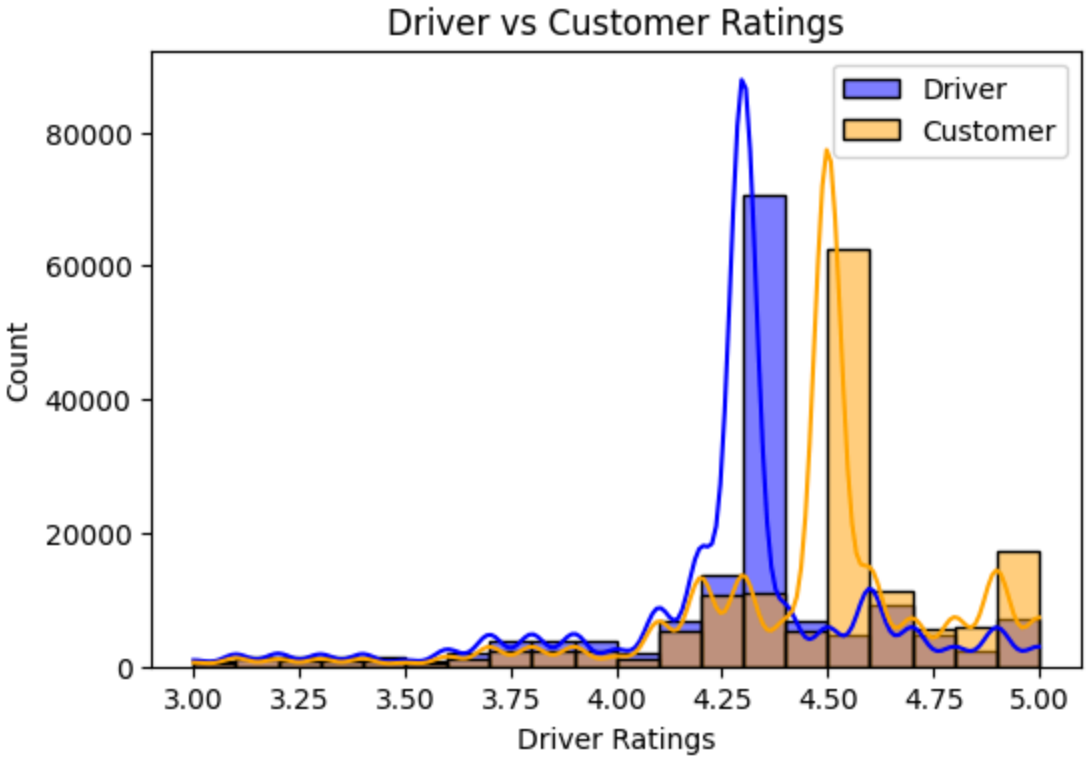


In [17]:
```
print("Total Revenue:", df["Booking Value"].sum())
print("Average Booking Value:", df["Booking Value"].mean())

plt.figure(figsize=(6,4))
sns.histplot(df["Booking Value"], bins=50, kde=True)
plt.title("Distribution of Booking Value")
plt.xlabel("Booking Value")
plt.ylabel("Frequency")
plt.show()
```

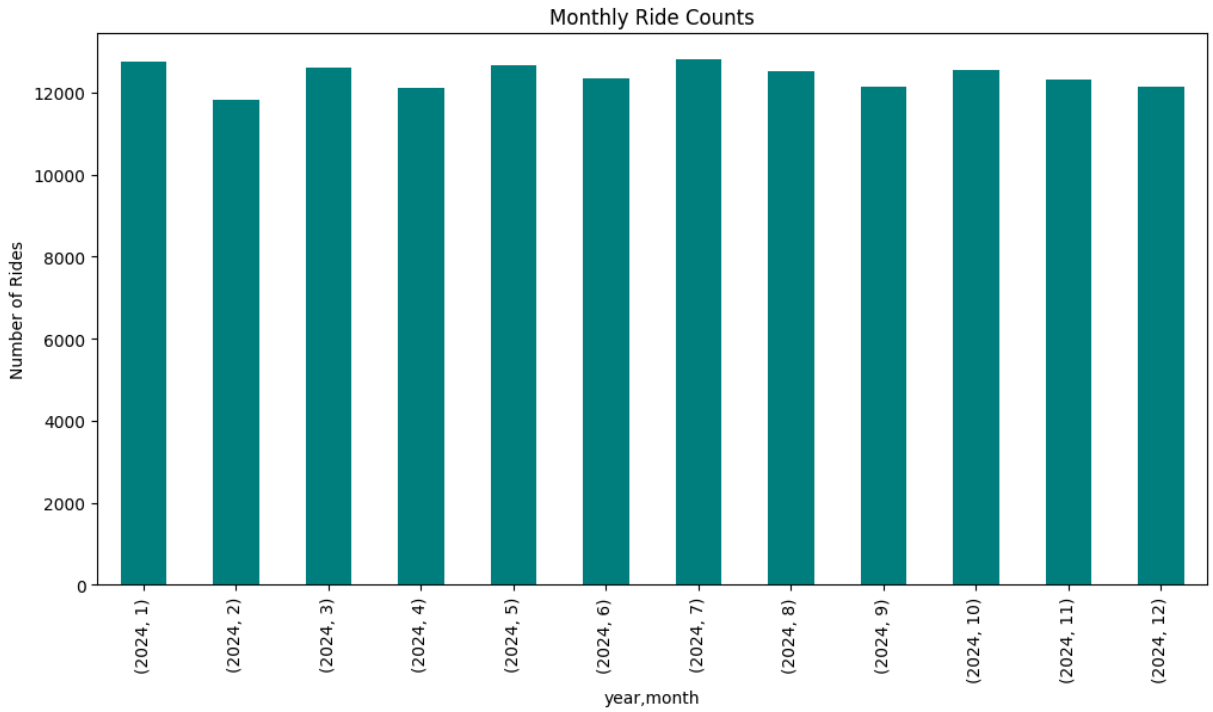
```
Total Revenue: 71129352.0
Average Booking Value: 478.1258746899514
```



In [18]:
```
plt.figure(figsize=(6,4))
sns.histplot(df["Driver Ratings"], bins=20, kde=True, color="blue", label="Driver")
sns.histplot(df["Customer Rating"], bins=20, kde=True, color="orange", label="Custo
plt.title("Driver vs Customer Ratings")
plt.legend()
plt.show()
```



In [20]:
```
# Monthly ride counts
monthly_rides = df.groupby(["year","month"])["Booking ID"].count()

monthly_rides.plot(kind="bar", figsize=(12,6), color="teal")
plt.title("Monthly Ride Counts")
plt.ylabel("Number of Rides")
plt.show()
```



In [21]:
```
df.head(10)
```

Out[21]:

	Date	Time	Booking ID	Booking Status	Customer ID	Vehicle Type	Pickup Location	Drop Location
0	2024-03-23	12:29:38	CNR5884300	No Driver Found	CID1982111	eBike	Palam Vihar	Jhilmil
1	2024-11-29	18:01:39	CNR1326809	Incomplete	CID4604802	Go Sedan	Shastri Nagar	Gurgaon Sector 56
2	2024-08-23	08:56:10	CNR8494506	Completed	CID9202816	Auto	Khandsa	Malviya Nagar
3	2024-10-21	17:17:25	CNR8906825	Completed	CID2610914	Premier Sedan	Central Secretariat	Inderlok
4	2024-09-16	22:08:00	CNR1950162	Completed	CID9933542	Bike	Ghitorni Village	Khan Market
5	2024-02-06	09:44:56	CNR4096693	Completed	CID4670564	Auto	AIIMS	Narsinghpur
6	2024-06-17	15:45:58	CNR2002539	Completed	CID6800553	Go Mini	Vaishali	Punjabi Bagh
7	2024-03-19	17:37:37	CNR6568000	Completed	CID8610436	Auto	Mayur Vihar	Cyber Hub
8	2024-09-14	12:49:09	CNR4510807	No Driver Found	CID7873618	Go Sedan	Noida Sector 62	Noida Sector 18
9	2024-12-16	19:06:48	CNR7721892	Incomplete	CID5214275	Auto	Rohini	Adarsh Nagar

10 rows × 21 columns

In [ ]: