Swiggy Sales Analysis

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Import data

In [2]:
```
df=pd.read_excel("swiggy_data.xlsx")
```

EDA

In [3]:
```
# Check First 5 rows

df.head()
```

Out[3]:

	State	City	Order Date	Restaurant Name	Location	Category	Dish Name	Price (INR)	Rating
0	Karnataka	Bengaluru	2025-06-29	Anand Sweets & Savouries	Rajarajeshwari Nagar	Snack	Butter Murukku-200gm	133.9	4.0
1	Karnataka	Bengaluru	2025-04-03	Srinidhi Sagar Deluxe	Kengeri	Recommended	Badam Milk	52.0	4.5
2	Karnataka	Bengaluru	2025-01-15	Srinidhi Sagar Deluxe	Kengeri	Recommended	Chow Chow Bath	117.0	4.7
3	Karnataka	Bengaluru	2025-04-17	Srinidhi Sagar Deluxe	Kengeri	Recommended	Kesari Bath	65.0	4.6
4	Karnataka	Bengaluru	2025-03-13	Srinidhi Sagar Deluxe	Kengeri	Recommended	Mix Raitha	130.0	4.0

In [4]:
```
# Check First 5 rows

df.tail()
```

Out[4]:

	State	City	Order Date	Restaurant Name	Location	Category	Dish Name	Price (INR)	Rating	Rating Count
197425	Sikkim	Gangtok	2025-01-25	Mama's Kitchen	Gangtok	Momos	Soya cheese chilli momo ...	112.0	4.4	0
197426	Sikkim	Gangtok	2025-07-02	Mama's Kitchen	Gangtok	Momos	Kurkure momo fried ...	140.0	4.4	0
197427	Sikkim	Gangtok	2025-03-25	Mama's Kitchen	Gangtok	Momos	Chilli cheese momo	126.0	4.4	0
197428	Sikkim	Gangtok	2025-03-26	Mama's Kitchen	Gangtok	Momos	Veg Momos (8 Pc)	85.0	4.4	0
197429	Sikkim	Gangtok	2025-03-27	Mama's Kitchen	Gangtok	Momos	Soya Momo	100.0	4.4	0

In [53]:
```
# Check overall info of data like data types, non-null count

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197430 entries, 0 to 197429
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   State            197430 non-null  object
 1   City             197430 non-null  object
 2   Order Date       197430 non-null  datetime64[ns]
 3   Restaurant Name  197430 non-null  object
 4   Location         197430 non-null  object
 5   Category         197430 non-null  object
 6   Dish Name        197430 non-null  object
 7   Price (INR)      197430 non-null  float64
 8   Rating           197430 non-null  float64
 9   Rating Count     197430 non-null  int64
 10  YearMonth        197430 non-null  object
 11  DayNames         197430 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(8)
memory usage: 18.1+ MB
```

In [6]:
```
# Check stastical Data

df.describe()
```

Out[6]:

	Price (INR)	Rating	Rating Count
count	197430.000000	197430.000000	197430.000000
mean	268.512920	4.341582	28.321805
std	219.338363	0.422585	87.542593
min	0.950000	1.500000	0.000000
25%	139.000000	4.300000	0.000000
50%	229.000000	4.400000	2.000000
75%	329.000000	4.500000	15.000000
max	8000.000000	5.000000	999.000000

In [7]:
```
#Check No of rows and columns

print("No of Rows and columns :",df.shape)
```

```
No of Rows and columns : (197430, 10)
```

In [100…
```
# Check columns names in data

df.columns
```

Out[100]:
```
Index(['State', 'City', 'Order Date', 'Restaurant Name', 'Location',
       'Category', 'Dish Name', 'Price (INR)', 'Rating', 'Rating Count',
       'YearMonth', 'DayNames', 'Food Category', 'Order_Date', 'Quater',
       'Quarter', 'Weekly_trend', 'Week'],
      dtype='object')
```

In [9]:
```
# Check data types of data

df.dtypes
```

Out[9]:
```
State                     object
City                      object
Order Date         datetime64[ns]
Restaurant Name           object
Location                  object
Category                  object
Dish Name                 object
Price (INR)              float64
Rating                   float64
Rating Count               int64
dtype: object
```

In [10]:
```
# Check is there any null values present in columns

df.isnull().sum()
```

Out[10]:
```
State              0
City               0
Order Date         0
Restaurant Name    0
Location           0
Category           0
Dish Name          0
Price (INR)        0
Rating             0
Rating Count       0
dtype: int64
```

Data Cleaning

In [102…
```
#Check if there is unique names are there and not repeated

df['State'].unique()
```

Out[102]:
```
array(['Karnataka', 'Maharashtra', 'Tamil Nadu', 'Delhi', 'Telangana',
       'West Bengal', 'Gujarat', 'Rajasthan', 'Uttar Pradesh', 'Punjab',
       'Madhya Pradesh', 'Bihar', 'Odisha', 'Haryana', 'Kerala',
       'Jharkhand', 'Assam', 'Chhattisgarh', 'Uttarakhand',
       'Himachal Pradesh', 'Goa', 'Jammu and Kashmir', 'Tripura',
       'Manipur', 'Meghalaya', 'Nagaland', 'Mizoram', 'Sikkim'],
      dtype=object)
```

In [12]:
```
df['Category'].unique()
```

Out[12]:
```
array(['Snack', 'Recommended', 'North Indian Gravy', ..., 'Chicken Momo',
       'Chicken Rice', 'make your own combo'], dtype=object)
```

KPI's

- Total Sales
- Average Orader Value
- Total Orders
- Average Rating
- Rating Count

In [13]:
```
# Total Sales
Total_Sales=df['Price (INR)'].sum()
```

In [14]:
```
# Average Ratings
Avg_Rating=df['Rating'].mean()
```

In [15]:
```
# Average Order Value
Avg_order_val=df['Price (INR)'].mean()
```

In [16]:
```
# Rating Count
Rating_count=df['Rating Count'].sum()
```

In [17]:
```
# Total Orders
Total_orders=len(df)
```

In [18]:
```
# DISPLAYS KPI's

print('Total Sales         :',round(Total_Sales,2))
print('Average Ratings     :',round(Avg_Rating,2))
print('Average Order Value :',round(Avg_order_val,2))
print('Rating Count        :',Rating_count)
print('Total Orders        :',Total_orders)
```

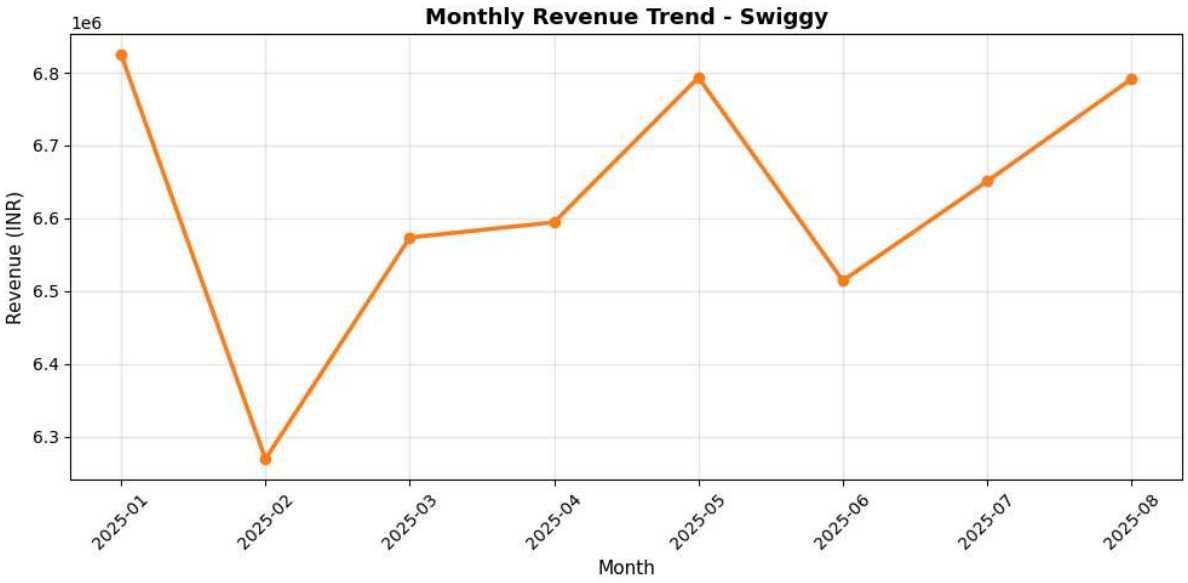
```
Total Sales         : 53012505.77
Average Ratings     : 4.34
Average Order Value : 268.51
Rating Count        : 5591574
Total Orders        : 197430
```

Charts Requirenment

Monthly Revenue Trend

In [115…

```
# Step 1: Convert 'Order Date' column into datetime format
# This allows us to extract month, year, week, etc. from the date
df['Order Date'] = pd.to_datetime(df['Order Date'])

# Step 2: Create a new column 'YearMonth' from Order Date
# to_period("M") groups dates into monthly periods (example: 2024-01, 2024-02)
df['YearMonth'] = df['Order Date'].dt.to_period("M").astype(str)

# Step 3: Group data by YearMonth and calculate total revenue for each month
monthly_revenue = (
    df.groupby('YearMonth')['Price (INR)']
    .sum()                 # Add all sales amounts for each month
    .reset_index()        # Convert result into a DataFrame
)

# Step 4: Create a figure for plotting
plt.figure(figsize=(10,5))

# Step 5: Plot Monthly Revenue Trend as a line chart
plt.plot(
    monthly_revenue["YearMonth"],     # X-axis → Months (2024-01, 2024-02, ...)
    monthly_revenue['Price (INR)'],   # Y-axis → Total Revenue per month
    color='#FC8019',                  # Swiggy orange color
    marker='o',                       # Show data points with circles
    linewidth=2.5                    # Thickness of the line
)

# Step 6: Improve chart readability
plt.xticks(rotation=45)              # Rotate month labels
plt.xlabel("Month", fontsize=11)     # X-axis label
plt.ylabel("Revenue (INR)", fontsize=11)  # Y-axis label
plt.title("Monthly Revenue Trend - Swiggy", fontsize=13, fontweight='bold')

# Step 7: Add light grid for easy reading
plt.grid(alpha=0.3)

# Step 8: Adjust layout to avoid overlapping text
plt.tight_layout()

# Step 9: Display the chart
plt.show()
```

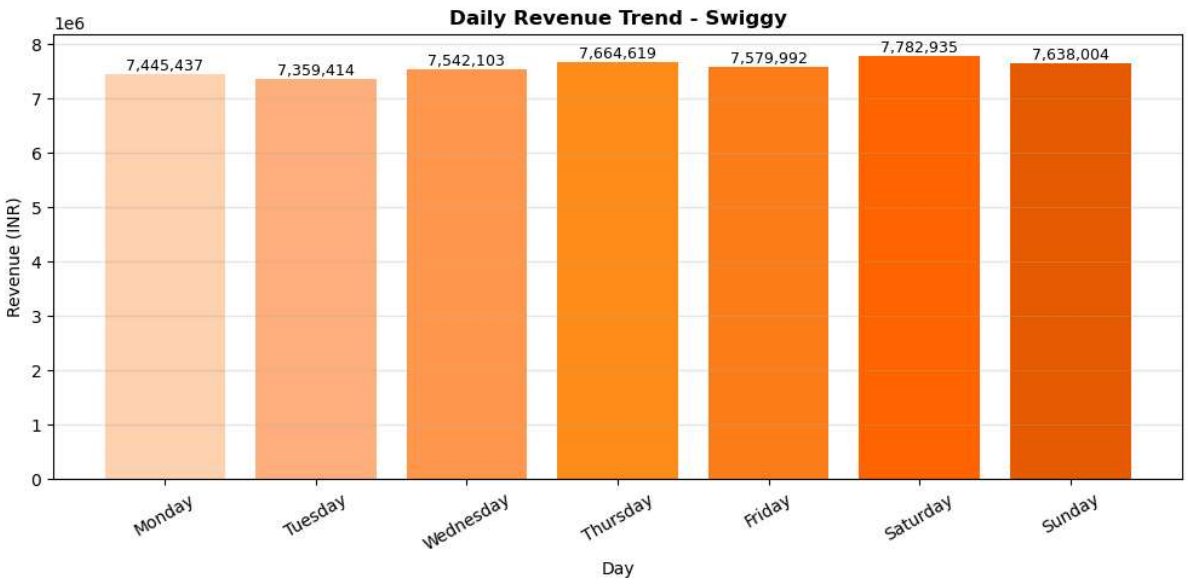


Daily Revenue Trend

In [106…

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Convert Order Date column to datetime and extract day names (Monday, Tues
df['DayNames'] = pd.to_datetime(df['Order Date']).dt.day_name()

# Step 2: Group data by DayNames and calculate total revenue for each day
daily_revenue = (
    df.groupby('DayNames')['Price (INR)']
    .sum()   # Sum of revenue for each day
    # Reorder days in correct weekly order instead of alphabetical order
    .reindex(["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday
)

# Step 3: Define Swiggy-style color gradient for bars
colors = ['#FFD2B3','#FFB380','#FF9A4D','#FF8C1A','#FC8019','#FF6600','#E65C00']

# Step 4: Create bar chart
plt.figure(figsize=(10,5))
bars = plt.bar(daily_revenue.index, daily_revenue.values, color=colors)

# Step 5: Add chart labels and title
plt.title("Daily Revenue Trend - Swiggy", fontweight='bold')
plt.xlabel("Day")
plt.ylabel("Revenue (INR)")
plt.xticks(rotation=30)

# Step 6: Add grid only on Y-axis for better readability
plt.grid(axis='y', alpha=0.3)

# Step 7: Display revenue values on top of each bar
for bar in bars:
    yval = bar.get_height()   # Get height (revenue) of each bar
    plt.text(
        bar.get_x() + bar.get_width()/2,  # X position
        yval,                             # Y position
        f'{int(yval):,}',                 # Format number with commas
        ha='center',
        va='bottom',
        fontsize=9
    )
```

```
# Step 8: Adjust layout so labels don't overlap
plt.tight_layout()

# Step 9: Show the chart
plt.show()
```



Total Sales by Food Type (Veg-Nonveg)

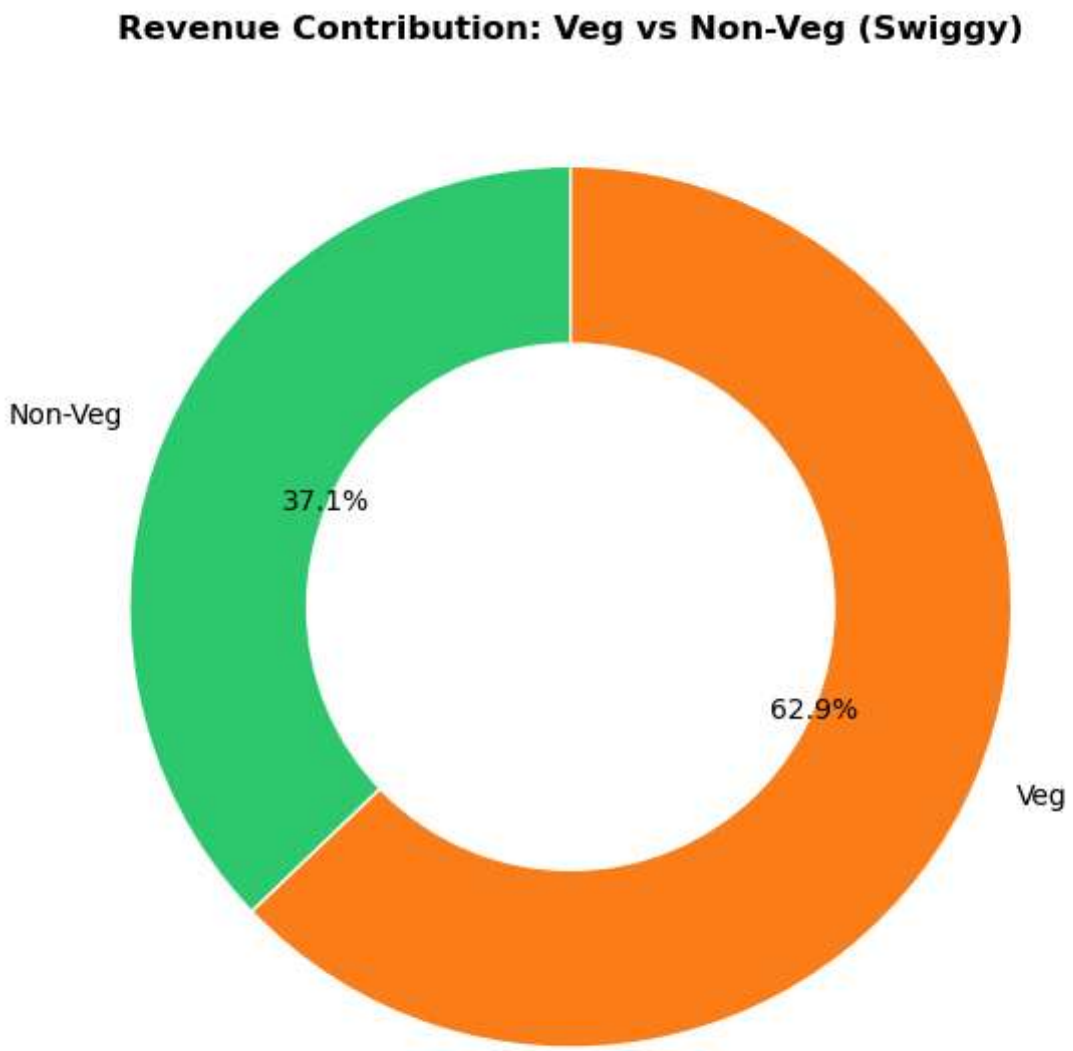
In [117…

```
# Step 1: Create a list of keywords that indicate Non-Veg food items
nonveg_keywords = ['chicken','egg','fish','biryani','prawn','mutton','kabab','kebab

# Step 2: Create a new column "Food Category" based on Dish Name
# Logic:
# If Dish Name contains any word from nonveg_keywords → label as "Non-Veg"
# Otherwise → label as "Veg"
df['Food Category'] = np.where(
    df['Dish Name'].str.lower().str.contains('|'.join(nonveg_keywords), na=False),
    "Non-Veg",
    "Veg"
)
```

In [125…

```
# Group data by Food Category and calculate total revenue
food_revenue = (
    df.groupby('Food Category')['Price (INR)']
      .sum()
)

# Define Swiggy brand colors
colors = ["#2ECC71", "#FC8019"]  # Veg = Green, Non-Veg = Swiggy Orange

plt.figure(figsize=(6,6))

# Create pie chart
wedges, texts, autotexts = plt.pie(
    food_revenue,
    labels=food_revenue.index,
    autopct='%1.1f%%',          # Show percentage on chart
    startangle=90,              # Rotate chart for better look
    colors=colors,
    wedgeprops={'edgecolor':'white'}
)
```

```
# Create donut effect by adding white circle at center
centre_circle = plt.Circle((0,0), 0.60, fc='white')
plt.gca().add_artist(centre_circle)

# Title
plt.title("Revenue Contribution: Veg vs Non-Veg (Swiggy)", fontweight='bold')

plt.tight_layout()
plt.show()
```

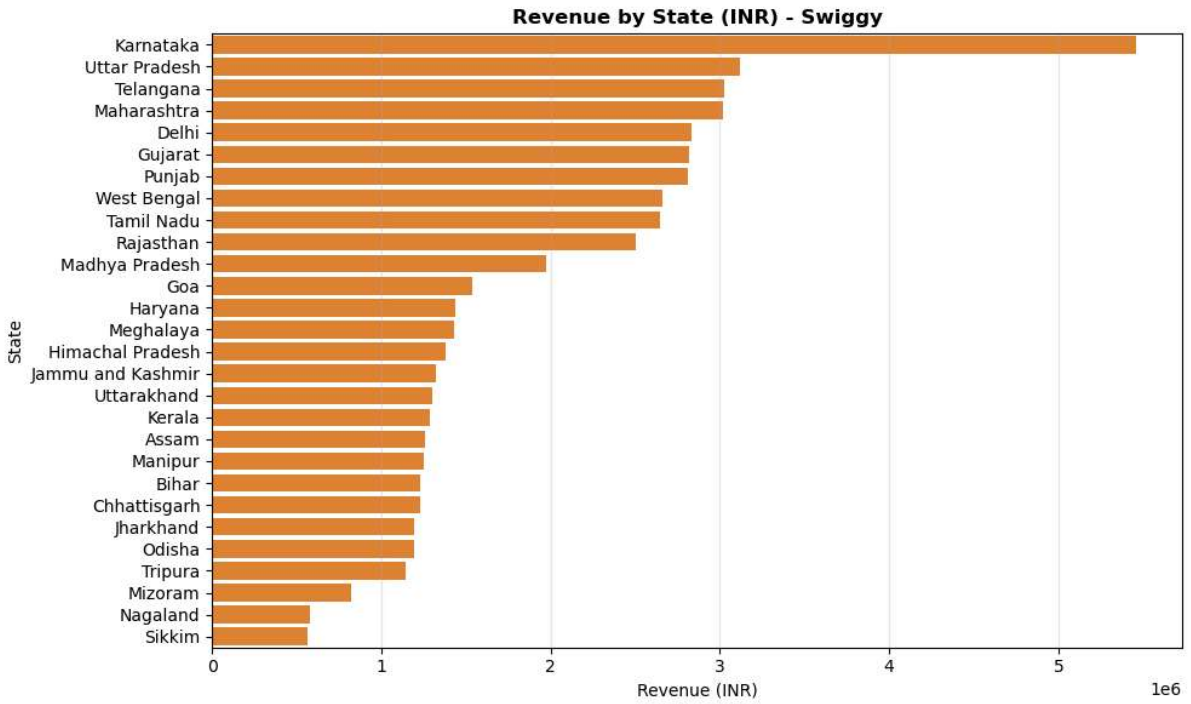


Total Sales by State (Map Visualization)

In [123…

```
import seaborn as sns
import matplotlib.pyplot as plt

state_revenue = (
    df.groupby("State")["Price (INR)"]
      .sum()
      .sort_values(ascending=False)
      .reset_index()
)

plt.figure(figsize=(10,6))

sns.barplot(
    data=state_revenue,
```

```
    x="Price (INR)",
    y="State",
    color="#FC8019"
)

plt.title("Revenue by State (INR) - Swiggy", fontweight='bold')
plt.xlabel("Revenue (INR)")
plt.ylabel("State")
plt.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()
```



Quterly performance Summary

In [111…

```
# Convert the 'Order Date' column into datetime format
# This allows us to extract time-related information like year, month, quarter, etc
df["Order_Date"] = pd.to_datetime(df["Order Date"])


# Extract Quarter from the Order_Date column
# to_period("Q") converts each date into a quarter format like: 2024Q1, 2024Q2
# astype(str) converts it into string so it looks clean in the table
df["Quarter"] = df["Order_Date"].dt.to_period("Q").astype(str)


# Create a quarterly summary table using groupby and aggregation
Quarterly_summary = (

    # Group the dataset by Quarter
    df.groupby("Quarter", as_index=False)

    # Calculate important business metrics for each quarter
    .agg(
        Total_Sales=("Price (INR)", "sum"),      # Total revenue generated in each q
        Avg_Rating=("Rating", "mean"),           # Average customer rating in each q
        Total_Orders=("Order Date", "count")     # Total number of orders placed in
    )

    # Sort the quarters in chronological order (Q1, Q2, Q3, Q4)
```

```
    .sort_values("Quarter")
)

# Display the final quarterly summary table
Quarterly_summary
```

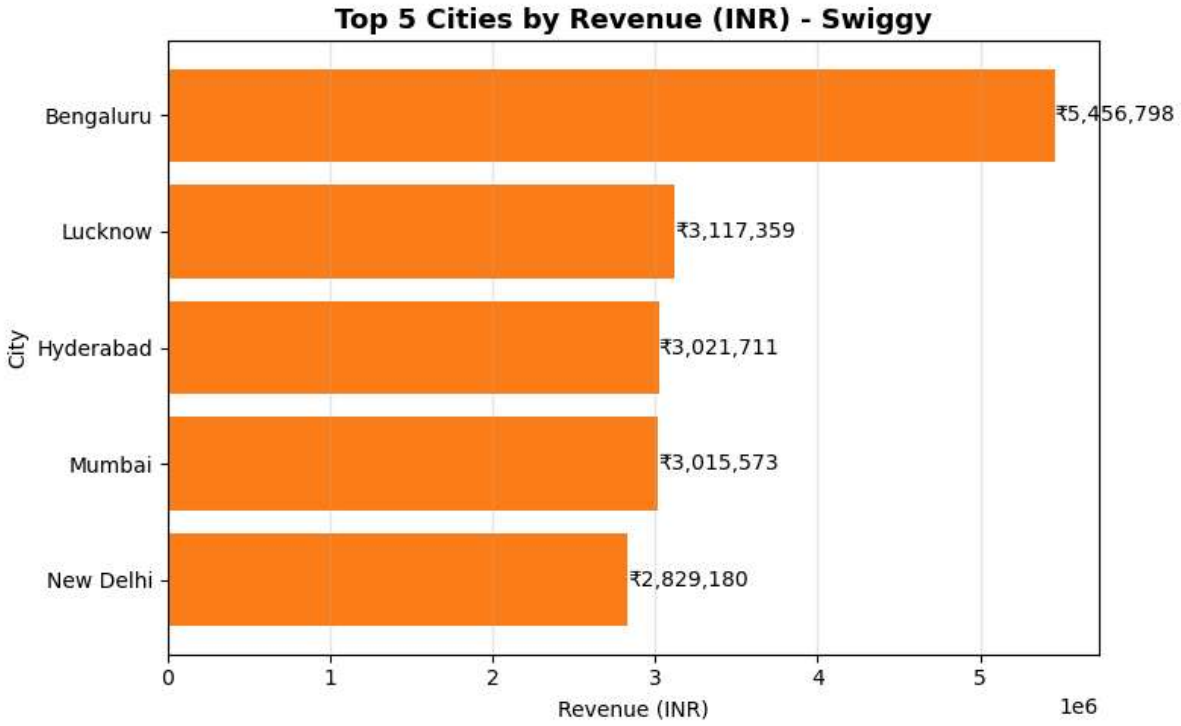
Out[111]:

	Quarter	Total_Sales	Avg_Rating	Total_Orders
0	2025Q1	19667821.77	4.342643	73096
1	2025Q2	19902256.59	4.340011	74163
2	2025Q3	13442427.41	4.342359	50171

Top 5 Cities by Sales

In [126…

```
import matplotlib.pyplot as plt

# Group data by City and calculate total revenue for each city
top_5_cities = (
    df.groupby("City")["Price (INR)"]   # Group rows by City and select Price colum
      .sum()                            # Calculate total revenue per city
      .nlargest(5)                      # Select top 5 cities with highest revenue
      .sort_values(ascending=True)      # Sort so highest value appears at the top
)

plt.figure(figsize=(8,5))

# Create horizontal bar chart
bars = plt.barh(
    top_5_cities.index,                 # Y-axis: City names
    top_5_cities.values,                # X-axis: Revenue values
    color="#FC8019"                     # Swiggy brand orange color
)

# Add chart title and axis labels
plt.title("Top 5 Cities by Revenue (INR) - Swiggy", fontweight='bold', fontsize=13)
plt.xlabel("Revenue (INR)")
plt.ylabel("City")

# Add grid lines for better readability
plt.grid(axis='x', alpha=0.3)

# Add revenue labels on each bar
for bar in bars:
    width = bar.get_width()
    plt.text(
        width,                          # Position text at end of bar
        bar.get_y() + bar.get_height()/2,
        f"₹{int(width):,}",             # Format number with commas
        va='center'
    )

plt.tight_layout()
plt.show()
```



Weekly trend analysis

In [113…

```
# Group data by Week and calculate total number of orders in each week
weekly_orders = (
    df.groupby('Week')['Order Date']          # Group rows based on Week and select
    .count()                                  # Count number of orders in each week
    .reset_index(name='Total_Orders')         # Convert result into DataFrame and r
    .sort_values('Week')                      # Sort weeks in chronological order
)

# Create a new figure for plotting
plt.figure(figsize=(10,5))

# Plot a line chart for weekly order trend
plt.plot(
    weekly_orders['Week'],        # X-axis: Week
    weekly_orders['Total_Orders'],# Y-axis: Total orders per week
    color='#2ECC71',              # Green color (represents order volume)
    marker='o',                   # Add circular marker on each data point
    linewidth=2.5                 # Thickness of the line
)

# Add labels to axes
plt.xlabel("Week", fontsize=11)         # Label for X-axis
plt.ylabel("Total Orders", fontsize=11) # Label for Y-axis

# Add title to the chart
plt.title("Weekly Order Volume Trend - Swiggy", fontsize=13, fontweight='bold')

# Rotate X-axis labels for better readability
plt.xticks(rotation=45)

# Add light grid for easier comparison
plt.grid(alpha=0.3)

# Adjust layout so labels and title do not overlap
plt.tight_layout()
```

```
# Display the chart
plt.show()
```

Conclusion

The Swiggy Sales Analysis project provided insights into revenue trends, customer preferences, and regional performance. The study revealed higher revenue contribution from Non-Veg items, increased order volume on weekends, and strong performance from top cities and states. Time-based analysis highlighted steady growth and seasonal patterns, supporting data-driven business decision making.