```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
import pandas as pd
```

```
df = pd.read_csv("/content/food_orders_new_delhi.csv")
```

```
df.head()
```

	Order ID	Customer ID	Restaurant ID	Order Date and Time	Delivery Date and Time	Order Value	Delivery Fee	Payment Method	Discounts and Offers	Commission Fee	Payment Processing Fee	Refunds/Chargebacks
0	1	C8270	R2924	2024-02-01 01:11:52	2024-02-01 02:39:52	1914	0	Credit Card	5% on App	150	47	0
1	2	C1860	R2054	2024-02-02 22:11:04	2024-02-02 22:46:04	986	40	Digital Wallet	10%	198	23	0
2	3	C6390	R2870	2024-01-31 05:54:35	2024-01-31 06:52:35	937	30	Cash on Delivery	15% New User	195	45	0

Next steps: Generate code with df | New interactive sheet

```
df.sample(5)
```

	Order ID	Customer ID	Restaurant ID	Order Date and Time	Delivery Date and Time	Order Value	Delivery Fee	Payment Method	Discounts and Offers	Commission Fee	Payment Processing Fee	Refunds/Chargeback
937	938	C5662	R2561	2024-01-31 07:36:12	2024-01-31 08:33:12	1627	50	Credit Card	50 off Promo	174	40	
182	183	C9004	R2793	2024-01-28 14:27:13	2024-01-28 15:56:13	1241	30	Digital Wallet	NaN	95	17	
98	99	C3731	R2363	2024-02-06 03:29:11	2024-02-06 04:39:11	636	20	Credit Card	15% New User	88	24	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Order ID               1000 non-null   int64
 1   Customer ID            1000 non-null   object
 2   Restaurant ID          1000 non-null   object
 3   Order Date and Time    1000 non-null   object
 4   Delivery Date and Time 1000 non-null   object
 5   Order Value            1000 non-null   int64
 6   Delivery Fee           1000 non-null   int64
 7   Payment Method         1000 non-null   object
 8   Discounts and Offers   815 non-null    object
 9   Commission Fee         1000 non-null   int64
 10  Payment Processing Fee 1000 non-null   int64
 11  Refunds/Chargebacks    1000 non-null   int64
dtypes: int64(6), object(6)
memory usage: 93.9+ KB
```

```
from datetime import datetime

# convert date and time columns to datetime
df['Order Date and Time'] = pd.to_datetime(df['Order Date and Time'])
df['Delivery Date and Time'] = pd.to_datetime(df['Delivery Date and Time'])

# Function to extract numeric discount values safely
def extract_discount(discount_str):
    if pd.isna(discount_str):
        return 0.0
    discount_str = str(discount_str).strip().lower()
    if '%' in discount_str:
        # Percentage discount like "10% off"
        return float(discount_str.split('%')[0])
    elif 'off' in discount_str:
        # Fixed discount like "₹100 off"
        amount = ''.join([ch for ch in discount_str if ch.isdigit() or ch == '.'])
        return float(amount) if amount else 0.0
    else:
        return 0.0

# Extract discount values and type
df['Discount Percentage'] = df['Discounts and Offers'].apply(extract_discount)
df['Discount Type'] = df['Discounts and Offers'].apply(
    lambda x: 'percentage' if '%' in str(x) else 'fixed'
)

# Calculate discount amount based on type
df['Discount Amount'] = df.apply(
    lambda x: (x['Order Value'] * x['Discount Percentage'] / 100)
              if x['Discount Type'] == 'percentage'
              else x['Discount Percentage'],
    axis=1
)

# Display result and data types
print(df[['Order Value', 'Discounts and Offers', 'Discount Type', 'Discount Percentage', 'Discount Amount']].head())
print(df.dtypes)
```

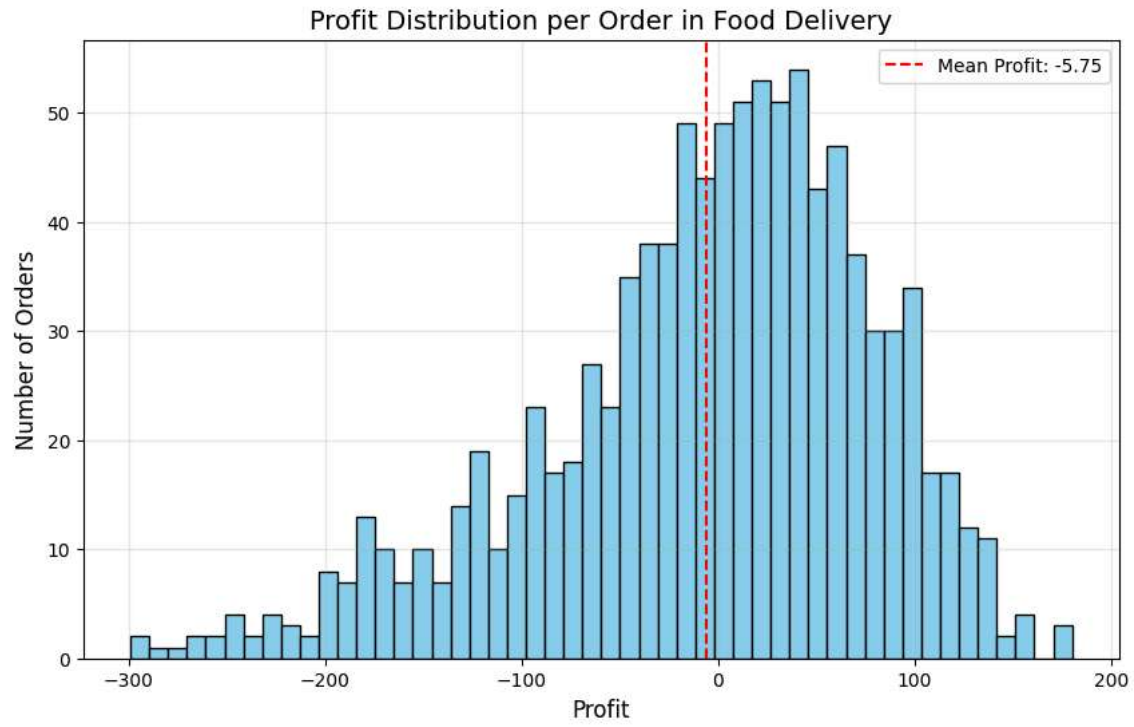
```
   Order Value Discounts and Offers Discount Type  Discount Percentage  \
0         1914            5% on App    percentage                  5.0
1          986                  10%    percentage                 10.0
2          937         15% New User    percentage                 15.0
3         1463                  NaN         fixed                  0.0
4         1992         50 off Promo         fixed                 50.0

   Discount Amount
0            95.70
1            98.60
2           140.55
3             0.00
4            50.00
Order ID                          int64
Customer ID                      object
Restaurant ID                    object
Order Date and Time      datetime64[ns]
Delivery Date and Time   datetime64[ns]
Order Value                       int64
Delivery Fee                      int64
Payment Method                   object
Discounts and Offers             object
Commission Fee                    int64
Payment Processing Fee            int64
Refunds/Chargebacks               int64
Discount Percentage             float64
Discount Type                    object
Discount Amount                 float64
dtype: object
```

```
# calculate total costs and revenue per order
df['Total Costs'] = df['Delivery Fee'] + df['Payment Processing Fee'] + df['Discount Amount']
df['Revenue'] = df['Commission Fee']
df['Profit'] = df['Revenue'] - df['Total Costs']

# aggregate data to get overall metrics
total_orders = df.shape[0]
total_revenue = df['Revenue'].sum()
total_costs = df['Total Costs'].sum()
```

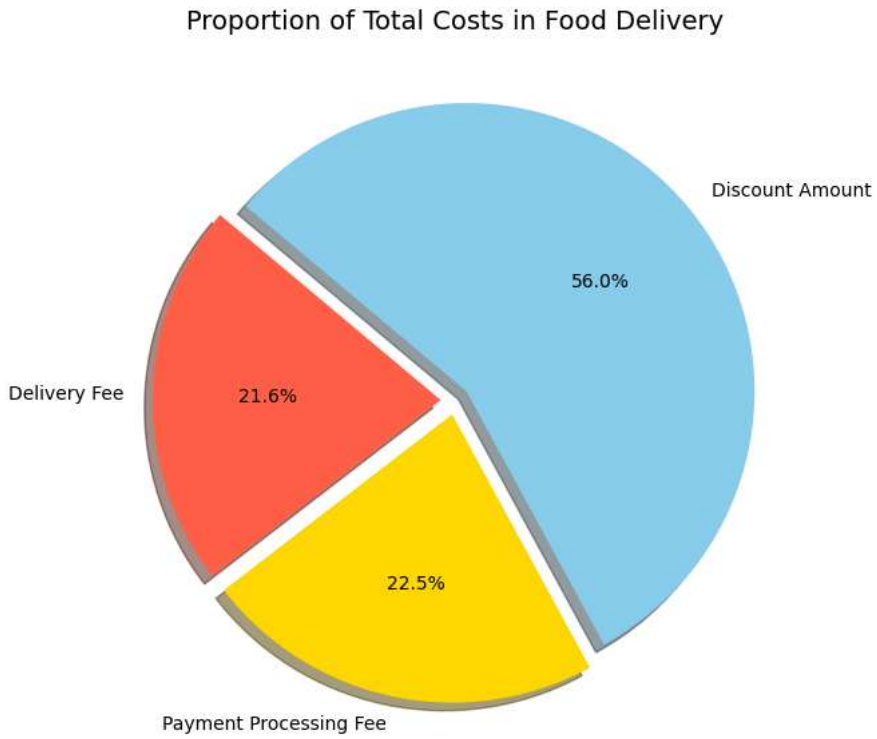
```
total_profit = df['Profit'].sum()

overall_metrics = {
    "Total Orders": total_orders,
    "Total Revenue": total_revenue,
    "Total Costs": total_costs,
    "Total Profit": total_profit
}

print(overall_metrics)
```

```
{'Total Orders': 1000, 'Total Revenue': np.int64(126990), 'Total Costs': np.float64(132741.84999999998), 'Total Profit': np.float64
```

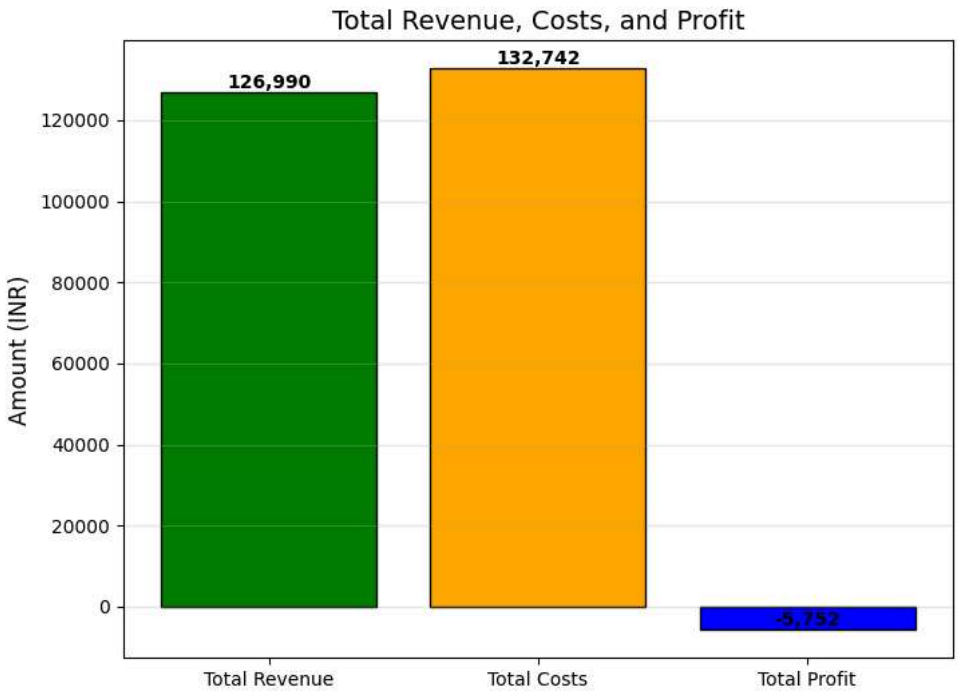
```
import matplotlib.pyplot as plt

# Histogram of profits per order
plt.figure(figsize=(10, 6))
plt.hist(df['Profit'], bins=50, color='skyblue', edgecolor='black')
plt.title('Profit Distribution per Order in Food Delivery', fontsize=14)
plt.xlabel('Profit', fontsize=12)
plt.ylabel('Number of Orders', fontsize=12)
plt.axvline(df['Profit'].mean(), color='red', linestyle='dashed', linewidth=1.5, label=f"Mean Profit: {df['Profit'].mean():.2f}")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```




```
import matplotlib.pyplot as plt

# Pie chart for the proportion of total costs
costs_breakdown = df[['Delivery Fee', 'Payment Processing Fee', 'Discount Amount']].sum()

plt.figure(figsize=(7, 7))
plt.pie(
    costs_breakdown,
    labels=costs_breakdown.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=['tomato', 'gold', 'skyblue'],
    explode=[0.05, 0.05, 0.05],  # slight separation for clarity
    shadow=True
)
plt.title('Proportion of Total Costs in Food Delivery', fontsize=14)
plt.show()
```




```
import matplotlib.pyplot as plt

# Bar chart comparing total revenue, costs, and profit
totals = ['Total Revenue', 'Total Costs', 'Total Profit']
values = [total_revenue, total_costs, total_profit]

plt.figure(figsize=(8, 6))
bars = plt.bar(totals, values, color=['green', 'orange', 'blue'], edgecolor='black')

# Add value labels on top of each bar
for bar in bars:
    plt.text(
        bar.get_x() + bar.get_width()/2,
        bar.get_height(),
        f"{bar.get_height():,.0f}",
        ha='center',
        va='bottom',
        fontsize=10,
        fontweight='bold'
    )

plt.title('Total Revenue, Costs, and Profit', fontsize=14)
plt.ylabel('Amount (INR)', fontsize=12)
plt.grid(axis='y', alpha=0.3)
plt.show()
```

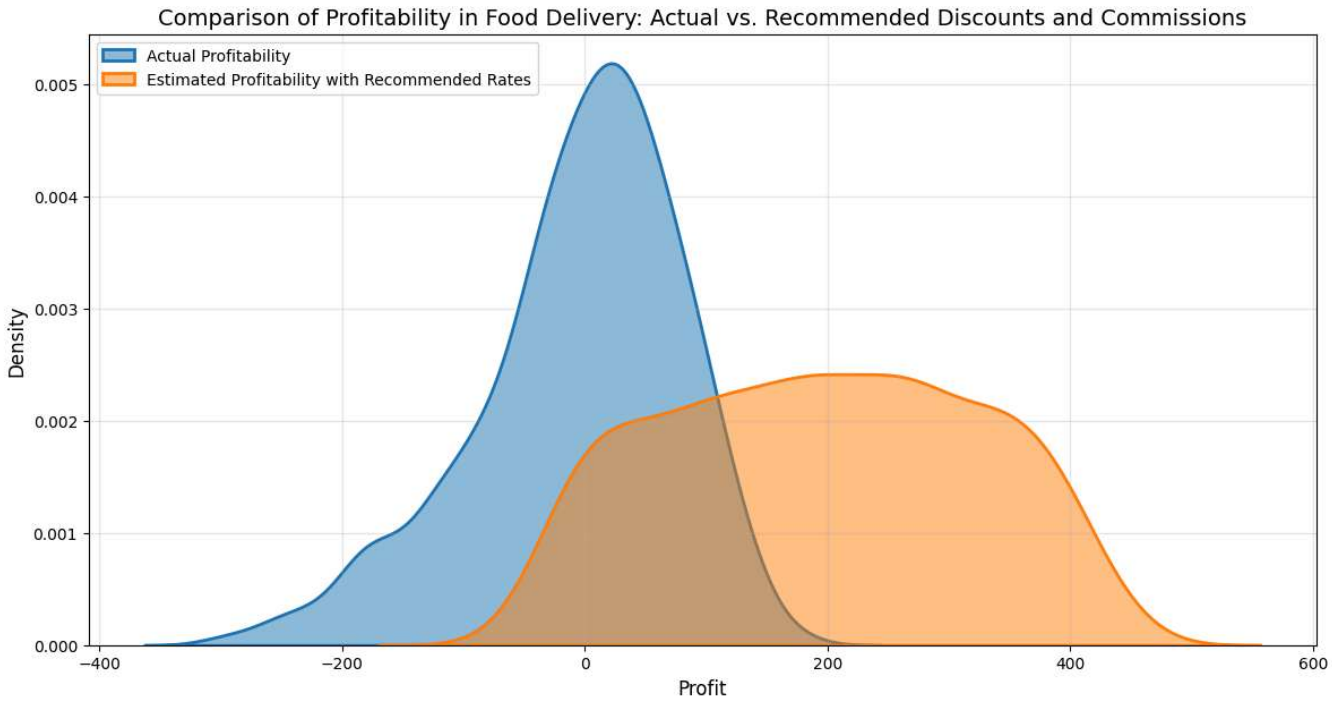



```
# Filter profitable orders (make a copy to avoid SettingWithCopyWarning)
profitable_orders = df[df['Profit'] > 0].copy()

# Calculate the average commission percentage for profitable orders
profitable_orders['Commission Percentage'] = (
    profitable_orders['Commission Fee'] / profitable_orders['Order Value']
) * 100

# Calculate the average discount percentage for profitable orders
profitable_orders['Effective Discount Percentage'] = (
    profitable_orders['Discount Amount'] / profitable_orders['Order Value']
) * 100

# Calculate new averages
new_avg_commission_percentage = profitable_orders['Commission Percentage'].mean()
new_avg_discount_percentage = profitable_orders['Effective Discount Percentage'].mean()

print("Average Commission %:", round(new_avg_commission_percentage, 2))
print("Average Effective Discount %:", round(new_avg_discount_percentage, 2))
```

```
Average Commission %: 27.71
Average Effective Discount %: 5.62
```

```
import seaborn as sns


# Recommended percentage values
recommended_commission_percentage = 30.0  # 30%
recommended_discount_percentage = 6.0     # 6%

# Calculate simulated commission fee and discount amount using recommended percentages
df['Simulated Commission Fee'] = df['Order Value'] * (recommended_commission_percentage / 100)
df['Simulated Discount Amount'] = df['Order Value'] * (recommended_discount_percentage / 100)

# Recalculate total costs and profit with simulated values
df['Simulated Total Costs'] = (
    df['Delivery Fee'] +
    df['Payment Processing Fee'] +
    df['Simulated Discount Amount']
)

df['Simulated Profit'] = (
    df['Simulated Commission Fee'] - df['Simulated Total Costs']
)

# Visualizing the comparison
```

```
plt.figure(figsize=(14, 7))

# Actual profitability
sns.kdeplot(df['Profit'], label='Actual Profitability', fill=True, alpha=0.5, linewidth=2)

# Simulated profitability
sns.kdeplot(df['Simulated Profit'], label='Estimated Profitability with Recommended Rates', fill=True, alpha=0.5, linewidth=2)

plt.title('Comparison of Profitability in Food Delivery: Actual vs. Recommended Discounts and Commissions', fontsize=14)
plt.xlabel('Profit', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.legend(loc='upper left', fontsize=10)
plt.grid(alpha=0.3)
plt.show()
```



Start coding or generate with AI.