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

In [2]:
```python
df = pd.read_csv(r"C:\Users\Ajith\Downloads\food_orders_new_delhi.csv")
```

In [3]:
```python
df
```

Out[3]:

	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
3	4	C6191	R2642	2024-01-16 22:52:49	2024-01-16 23:38:49	1463	50	Cash on Delivery	None	146	27	0
4	5	C6734	R2799	2024-01-29 01:19:30	2024-01-29 02:48:30	1992	30	Cash on Delivery	50 off Promo	130	50	0
...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	C6232	R2129	2024-01-14 05:57:00	2024-01-14 06:39:00	825	0	Digital Wallet	5% on App	165	47	50
996	997	C6797	R2742	2024-01-28 08:50:43	2024-01-28 10:10:43	1627	50	Cash on Delivery	None	110	42	0
997	998	C5926	R2837	2024-01-21 09:43:19	2024-01-21 10:44:19	553	20	Cash on Delivery	None	64	31	0
998	999	C7016	R2144	2024-01-30 22:23:38	2024-01-31 00:07:38	1414	0	Cash on Delivery	15% New User	199	34	0
999	1000	C4335	R2890	2024-01-08 14:46:43	2024-01-08 15:39:43	1657	20	Digital Wallet	15% New User	180	27	100

1000 rows × 12 columns

In [4]:
```python
df.head()
```

Out[4]:

	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
3	4	C6191	R2642	2024-01-16 22:52:49	2024-01-16 23:38:49	1463	50	Cash on Delivery	None	146	27	0
4	5	C6734	R2799	2024-01-29 01:19:30	2024-01-29 02:48:30	1992	30	Cash on Delivery	50 off Promo	130	50	0

In [5]:
```python
df.describe()
```

Out[5]:

	Order ID	Order Value	Delivery Fee	Commission Fee	Payment Processing Fee	Refunds/Chargebacks
count	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000
mean	500.500000	1053.969000	28.620000	126.99000	29.832000	28.300000
std	288.819436	530.975339	16.958278	43.06405	11.627165	49.614228
min	1.000000	104.000000	0.000000	50.00000	10.000000	0.000000
25%	250.750000	597.750000	20.000000	90.00000	20.000000	0.000000
50%	500.500000	1038.500000	30.000000	127.00000	30.000000	0.000000
75%	750.250000	1494.000000	40.000000	164.00000	40.000000	50.000000
max	1000.000000	1995.000000	50.000000	200.00000	50.000000	150.000000

In [6]:
```python
df.shape
```

Out[6]:
```
(1000, 12)
```

In [7]:
```python
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    1000 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
```

In [8]:
```python
df.isnull().sum()
```

Out[8]:
```
Order ID                  0
Customer ID               0
Restaurant ID             0
Order Date and Time       0
Delivery Date and Time    0
Order Value               0
Delivery Fee              0
Payment Method            0
Discounts and Offers      0
Commission Fee            0
Payment Processing Fee    0
Refunds/Chargebacks       0
dtype: int64
```

In [9]:
```python
df.columns
```

Out[9]:
```
Index(['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'],
      dtype='object')
```

In [10]:
```python
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'])
```

In [11]:
```python
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   datetime64[ns]
 4   Delivery Date and Time  1000 non-null   datetime64[ns]
 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    1000 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: datetime64[ns](2), int64(6), object(4)
memory usage: 93.9+ KB
```

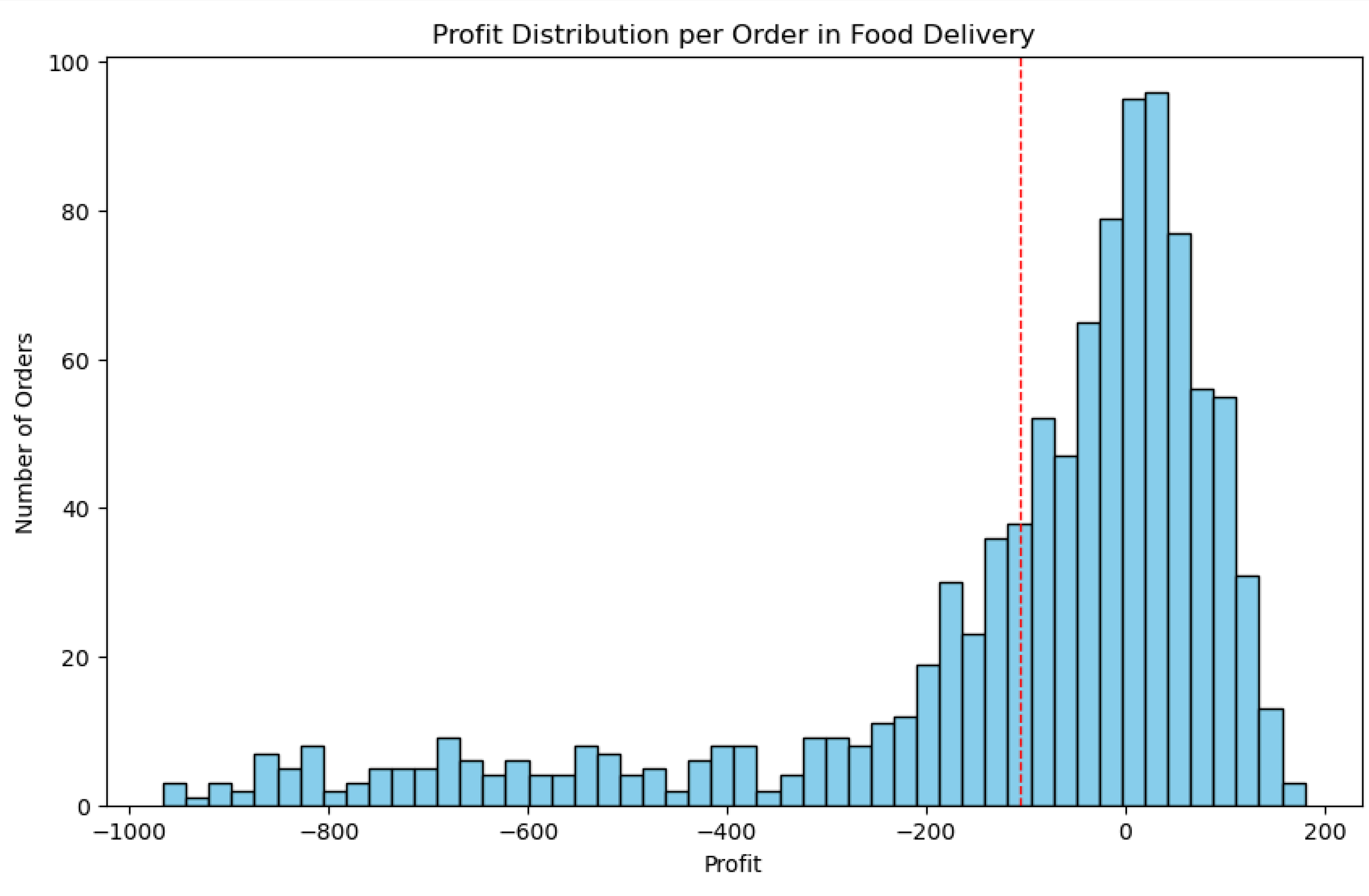
In [12]:
```python
# first, let's create a function to extract numeric values from the 'Discounts and Offers' string
def extract_discount(discount_str):
    if 'off' in discount_str:
        # Fixed amount off
        return float(discount_str.split(' ')[0])
    elif '%' in discount_str:
        # Percentage off
        return float(discount_str.split('%')[0])
    else:
        # No discount
        return 0.0

# apply the function to create a new 'Discount Value' column
df['Discount Percentage'] = df['Discounts and Offers'].apply(lambda x: extract_discount(x))

# for percentage discounts, calculate the discount amount based on the order value
df['Discount Amount'] = df.apply(lambda x: (x['Order Value'] * x['Discount Percentage'] / 100)
                                                   if x['Discount Percentage'] > 1
                                                   else x['Discount Percentage'], axis=1)

# adjust 'Discount Amount' for fixed discounts directly specified in the 'Discounts and Offers' column
df['Discount Amount'] = df.apply(lambda x: x['Discount Amount'] if x['Discount Percentage'] <= 1
                                                   else x['Order Value'] * x['Discount Percentage'] / 100, axis=1)

print(df[['Order Value', 'Discounts and Offers', 'Discount Percentage', 'Discount Amount']].head(), df.dtypes)
```

```
   Order Value Discounts and Offers  Discount Percentage  Discount Amount
0         1914            5% on App                  5.0            95.70
1          986                  10%                 10.0            98.60
2          937         15% New User                 15.0           140.55
3         1463                 None                  0.0             0.00
4         1992         50 off Promo                 50.0           996.00 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 Amount                  float64
dtype: object
```

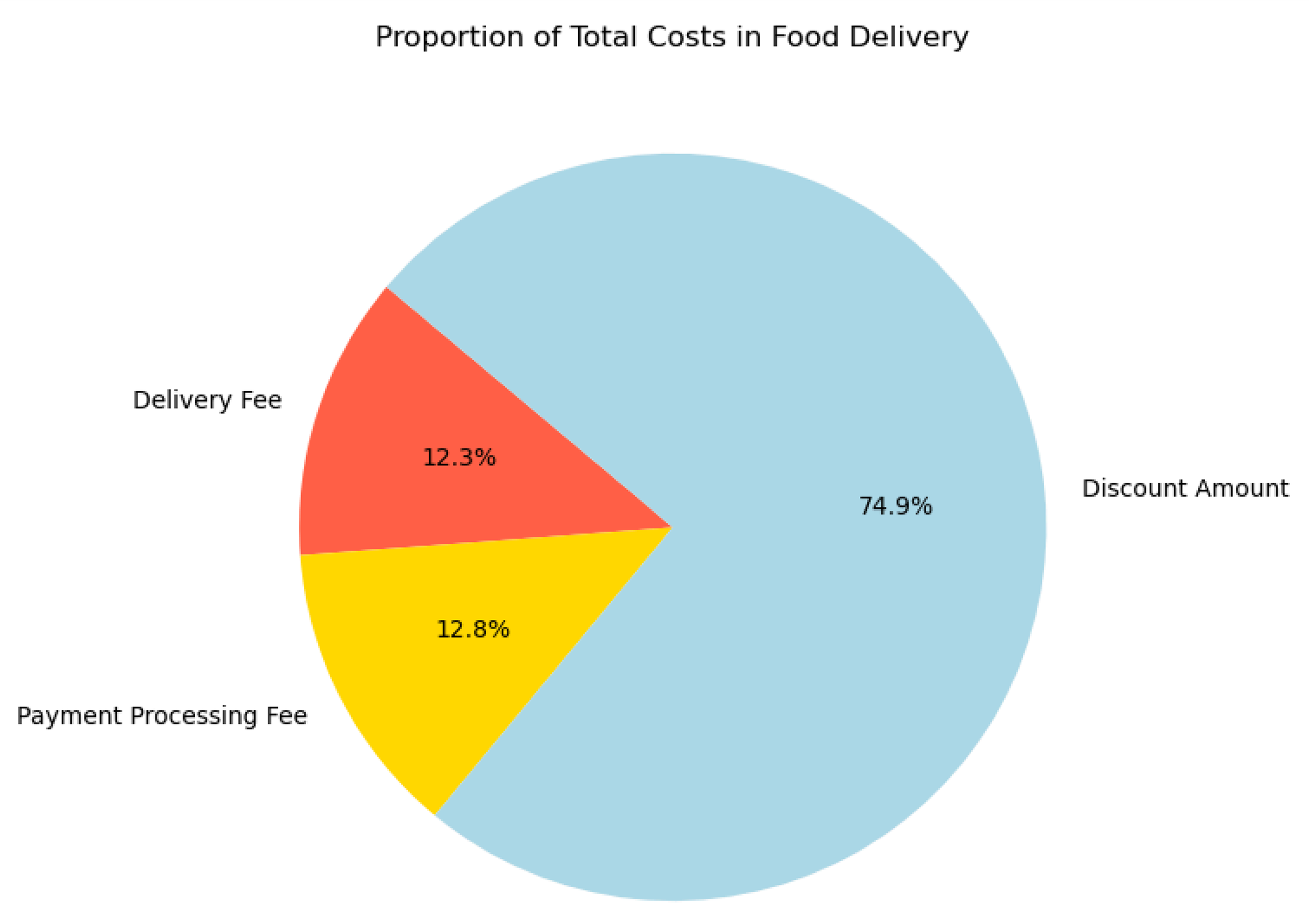
In [13]:
```python
# calculate total costs and revenue per order
df['Total Costs'] = df['Delivery Fee'] + df['Payment Processing Fee'] + df['Discount Amount']
df['Revenue'] = df['Commission Fee']
df['Profit'] = df['Revenue'] - df['Total Costs']

# 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': 126990, 'Total Costs': 232709.85, 'Total Profit': -105719.85}
```

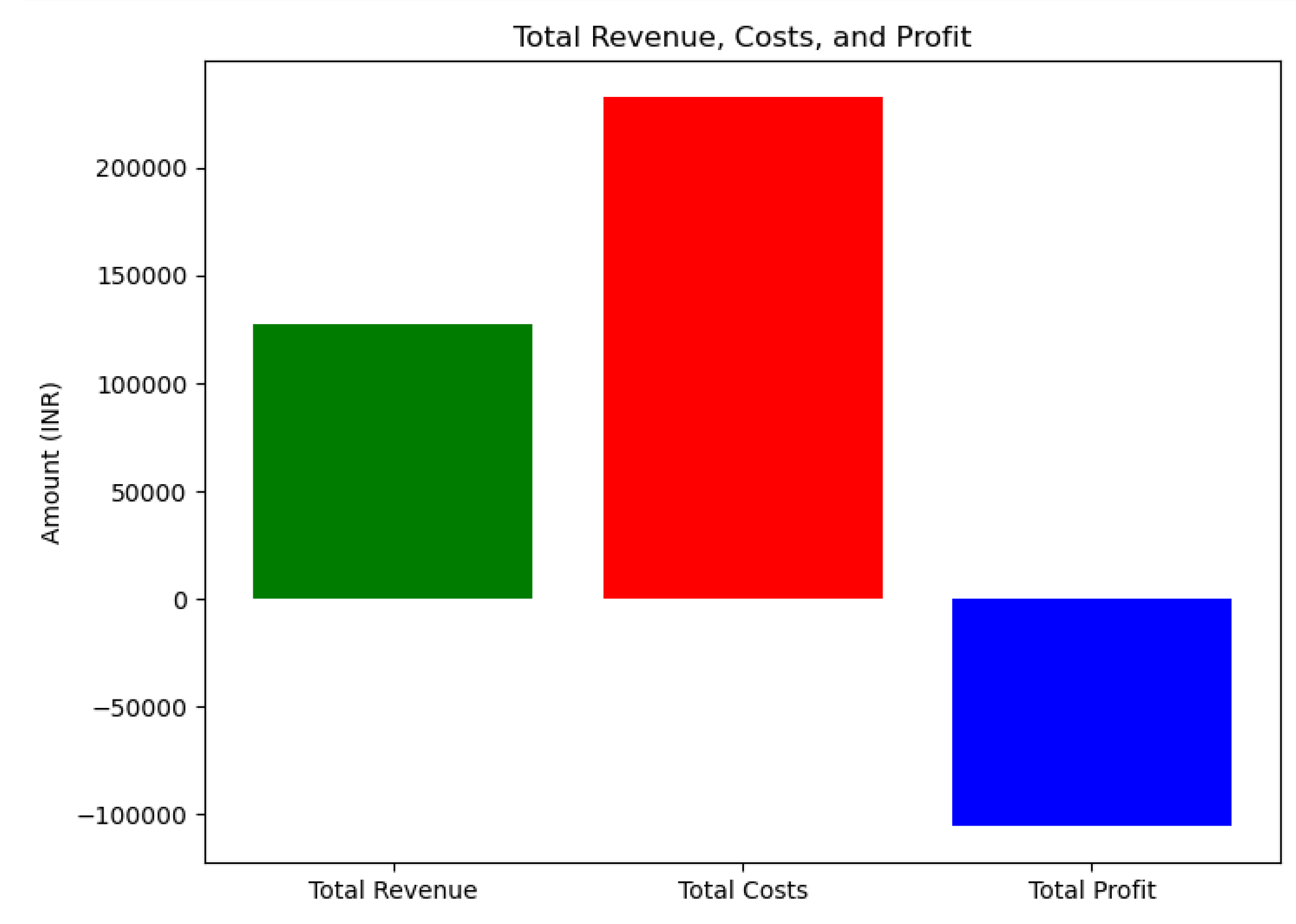
In [14]:
```python
# histogram of profits per order
plt.figure(figsize=(10, 6))
plt.hist(df['Profit'], bins=50, color='skyblue', edgecolor='black')
plt.title('Profit Distribution per Order in Food Delivery')
plt.xlabel('Profit')
plt.ylabel('Number of Orders')
plt.axvline(df['Profit'].mean(), color='red', linestyle='dashed', linewidth=1)
plt.show()
```



In [14]:
```python
# 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', 'lightblue'])
plt.title('Proportion of Total Costs in Food Delivery')
plt.show()
```



In [15]:
```python
# bar chart for total revenue, costs, and profit

totals = ['Total Revenue', 'Total Costs', 'Total Profit']
values = [total_revenue, total_costs, total_profit]
plt.figure(figsize=(8, 6))
plt.bar(totals, values, color=['green', 'red', 'blue'])
plt.title('Total Revenue, Costs, and Profit')
plt.ylabel('Amount (INR)')
plt.show()
```

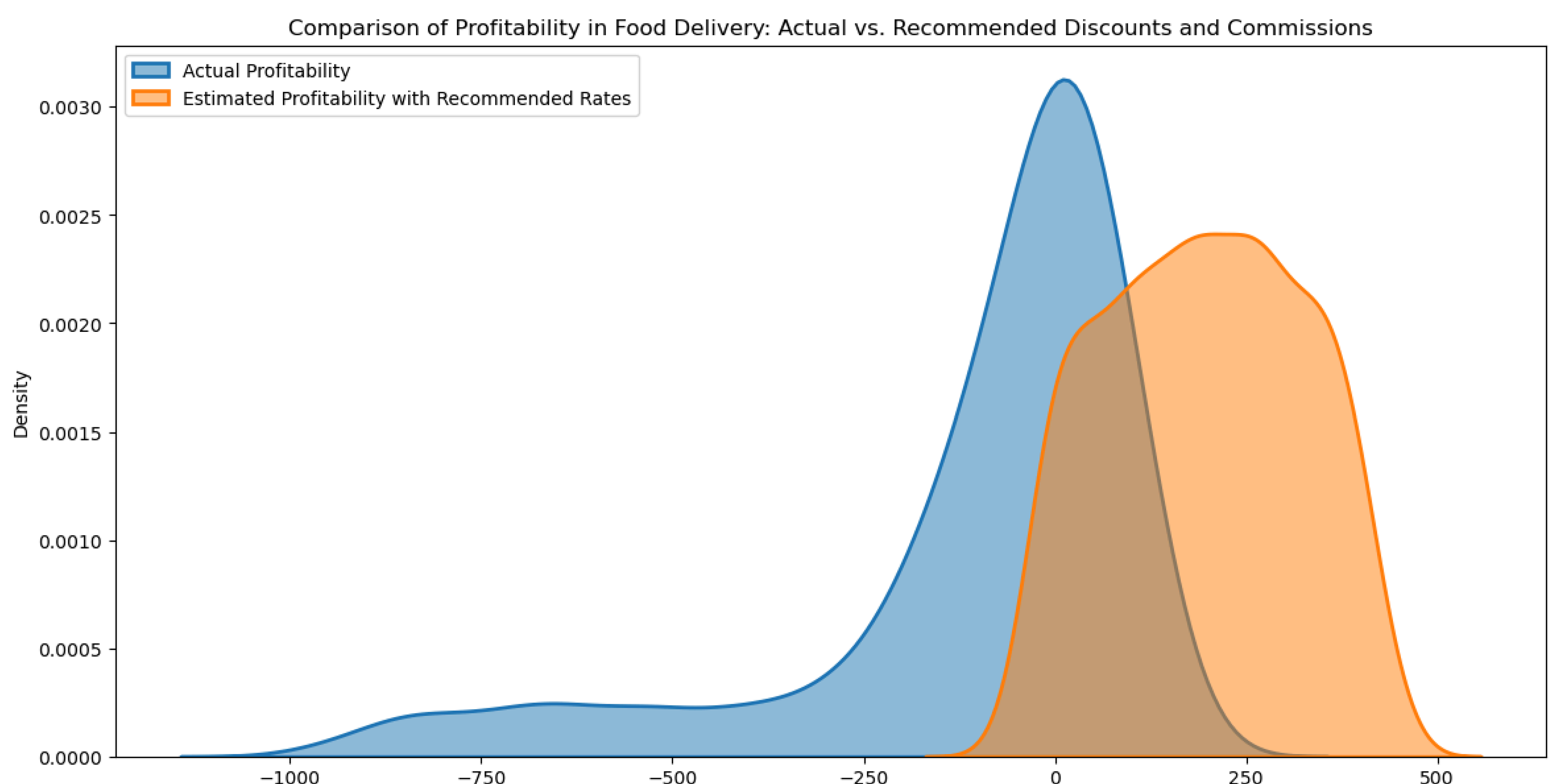


In [16]:
```python
# filter the dataset for profitable orders
profitable_orders = df[df['Profit'] > 0]

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

print(new_avg_commission_percentage, new_avg_discount_percentage)
```

```
30.508436145149435 5.867469879518072
```

```
C:\Users\Ajith\AppData\Local\Temp\ipykernel_9584\1524930468.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  profitable_orders['Commission Percentage'] = (profitable_orders['Commission Fee'] / profitable_orders['Order Value']) * 100
C:\Users\Ajith\AppData\Local\Temp\ipykernel_9584\1524930468.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  profitable_orders['Effective Discount Percentage'] = (profitable_orders['Discount Amount'] / profitable_orders['Order Value']) * 100
```

In [17]:
```python
# simulate profitability with recommended discounts and commissions
recommended_commission_percentage = 30.0  # 30%
recommended_discount_percentage = 6.0    # 6%

# calculate the 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'])


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')
plt.xlabel('Profit')
plt.ylabel('Density')
plt.legend(loc='upper left')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: