Food Delivery Cost and Profitability Analysis

Food Delivery Cost and Profitability Analysis involves examing all te direct and indirect costs associated with delivery of food orders A Food Delivery Service is facing issues in making their business profitable across its operations. With data of 1,000 orders the service seeks to understand the dynamics of their cost structure, profitability and to introduce new stratergies to increase the revenue and net profit.

The dataset contains comprehensive details on food orders, including Order ID, Customer ID, Restaurant ID, Order and Delivery Date and Time, Order Value, Delivery Fee, Payment Method, Discounts and Offers, Commission Fee, Payment Processing Fee, and Refunds/Chargebacks. This data provides a foundation for analyzing the cost structure and profitability of the food delivery service.

Now, let's get started with the task of Food Delivery Cost and Profitability Analysis by importing the necessary Python libraries and the dataset (https://statso.io/optimizing-cost-and-profitability-case-study/#google_vignette):

In [3]:
```
import pandas as pd

food_data = pd.read_csv("food_orders_new_delhi.csv")
print(food_data.head())
```

```
   Order ID Customer ID Restaurant ID  Order Date and Time  \
0         1       C8270         R2924  2024-02-01 01:11:52
1         2       C1860         R2054  2024-02-02 22:11:04
2         3       C6390         R2870  2024-01-31 05:54:35
3         4       C6191         R2642  2024-01-16 22:52:49
4         5       C6734         R2799  2024-01-29 01:19:30

  Delivery Date and Time  Order Value  Delivery Fee    Payment Method  \
0    2024-02-01 02:39:52         1914             0       Credit Card
1    2024-02-02 22:46:04          986            40    Digital Wallet
2    2024-01-31 06:52:35          937            30  Cash on Delivery
3    2024-01-16 23:38:49         1463            50  Cash on Delivery
4    2024-01-29 02:48:30         1992            30  Cash on Delivery

  Discounts and Offers  Commission Fee  Payment Processing Fee  \
0            5% on App             150                      47
1                  10%             198                      23
2         15% New User             195                      45
3                  NaN             146                      27
4         50 off Promo             130                      50

   Refunds/Chargebacks
0                    0
1                    0
2                    0
3                    0
4                    0
```

In [6]:
```
print(food_data.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
None
```

The dataset contains 1,000 entries and 12 columns, with no missing values in any of the fields. Now, we need to perform some data cleaning and preparation. Below are the necessary cleaning steps we need to perform:

Convert “Order Date and Time” and “Delivery Date and Time” to a datetime format.

Convert “Discounts and Offers” to a consistent numeric values to calculate the discount amounts.

In [9]:
```
from datetime import datetime

# converting date and time columns to datetime
food_data["Order Date and Time"] = pd.to_datetime(food_data["Order Date and Time"])
food_data["Delivery Date and Time"] = pd.to_datetime(food_data["Delivery Date and Time"])

#extracting the Discount percentage from discount column
food_data["Discount Percentage"] = food_data["Discounts and Offers"].str.extract('(\d+)').astype(float)

#filling 0 where no discount is applied
food_data["Discount Percentage"]=food_data["Discount Percentage"].fillna(0.0)
food_data["Discount Amount"]=(food_data["Order Value"]*food_data["Discount Percentage"])/100
print(food_data.head())
```

```
   Order ID Customer ID Restaurant ID Order Date and Time  \
0         1       C8270         R2924 2024-02-01 01:11:52
1         2       C1860         R2054 2024-02-02 22:11:04
2         3       C6390         R2870 2024-01-31 05:54:35
3         4       C6191         R2642 2024-01-16 22:52:49
4         5       C6734         R2799 2024-01-29 01:19:30

  Delivery Date and Time  Order Value  Delivery Fee    Payment Method  \
0    2024-02-01 02:39:52         1914             0       Credit Card
1    2024-02-02 22:46:04          986            40    Digital Wallet
2    2024-01-31 06:52:35          937            30  Cash on Delivery
3    2024-01-16 23:38:49         1463            50  Cash on Delivery
4    2024-01-29 02:48:30         1992            30  Cash on Delivery

  Discounts and Offers  Commission Fee  Payment Processing Fee  \
0            5% on App             150                      47
1                  10%             198                      23
2         15% New User             195                      45
3                  NaN             146                      27
4         50 off Promo             130                      50

   Refunds/Chargebacks  Discount Percentage  Discount Amount
0                    0                  5.0            95.70
1                    0                 10.0            98.60
2                    0                 15.0           140.55
3                    0                  0.0             0.00
4                    0                 50.0           996.00
```

Cost and Profitability Analysis

For the cost analysis, we'll consider the following direct/indirect costs associated with each order:

Delivery Fee : The fee charged for delivering the order.

Payment Processing Fee: The fee for processing the payment.

Discount Amount: The discount provided on the order.

Refunds/Chargebacks: Refunds incase of any complaints regarding the order.

We'll calculate the total cost per order and then aggregate it to understand the overall cost structure.

The 'Commission Fee' is the primary factor of the Revenue of the Business and then we'll find net profit by subtracting Total cost from Revenue generated.

In [10]:
```
#calculating total cost, revenue per order
food_data["Cost per order"] = (food_data["Delivery Fee"] + food_data["Payment Processing Fee"] +
                               food_data["Refunds/Chargebacks"] + food_data["Discount Amount"])

food_data["Revenue per order"] = food_data["Commission Fee"]

food_data["Profit per order"] = food_data["Revenue per order"] - food_data["Cost per order"]

#calculating total cost and profit for all the orders

print("Total number of Orders ",food_data.shape[0])
print("Total Cost",food_data["Cost per order"].sum())
print("Total Revenue",food_data["Revenue per order"].sum())
print("Total Profit",food_data["Profit per order"].sum())
```

```
Total number of Orders  1000
Total Cost 261009.84999999998
Total Revenue 126990
Total Profit -134019.85
```

Based on the analysis, here are the overall metrics for the food delivery operations:

Total Orders: 1,000

Total Revenue: 126,990 INR

Total Costs: 2,61,009.849 INR (including delivery fees, payment processing fees,discounts and refunds)

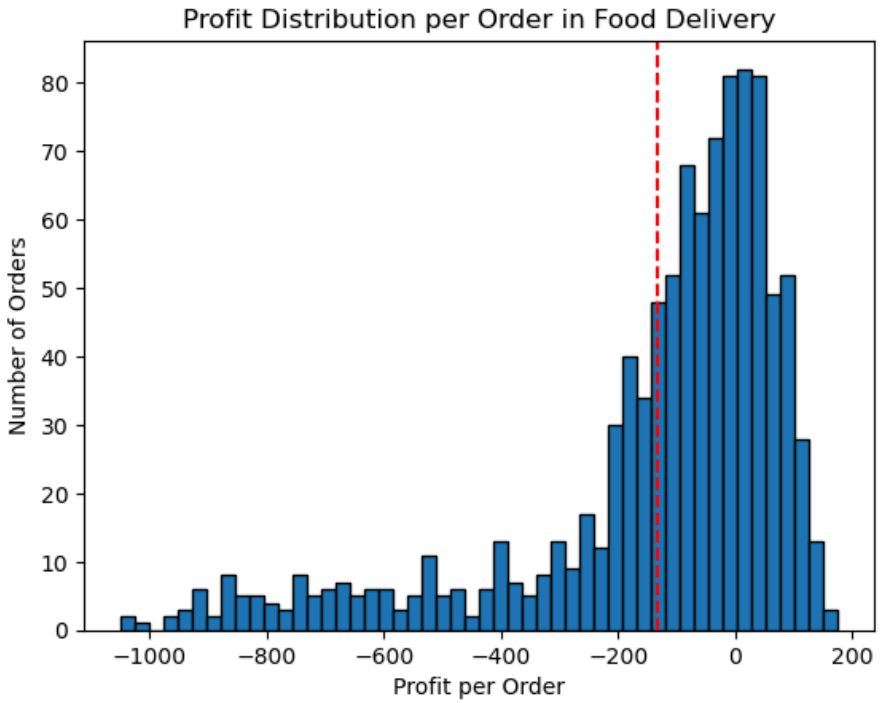
Total Profit: -1,34,019.85 INR

The analysis indicates that the total costs associated with the food delivery operations exceeds the total revenue generated from commission fees, resulting in a net loss. It confirms that the current commission rates, delivery fees, and discount strategies might not be sustainable for profitability.

To better understand the distribution of costs, revenue, and profit, let's plot:

1. A histogram of profits per order to visualize the distribution of profitable and unprofitable orders.
2. A pie chart to visualize the proportion of total costs (delivery fees, payment processing fees, and discounts).
3. A bar chart to compare total revenue, total costs, and total profit

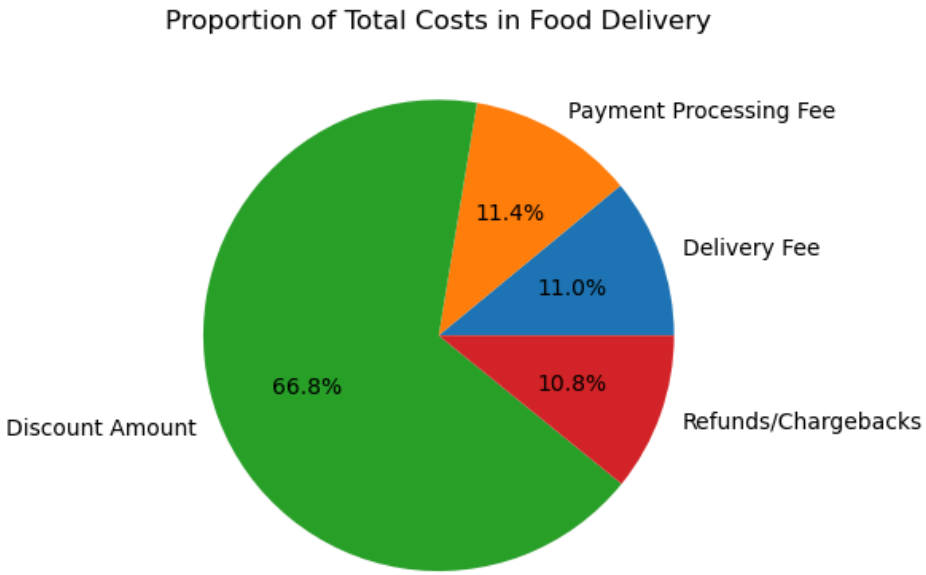
In [11]:
```
import matplotlib.pyplot as plt
plt.figure()
plt.hist(food_data['Profit per order'],bins=50,edgecolor="black")
plt.title('Profit Distribution per Order in Food Delivery')
plt.xlabel('Profit per Order')
plt.ylabel('Number of Orders')
plt.axvline(food_data["Profit per order"].mean(),color="red",linestyle="--")
plt.show()
```



The histogram shows a wide distribution of profit per order, with a noticeable number of orders resulting in a loss (on left side of 0). The red dashed line indicates the average profit, which is in the negative territory, highlighting the overall loss-making situation.

Now lets have look at proportion of total cost

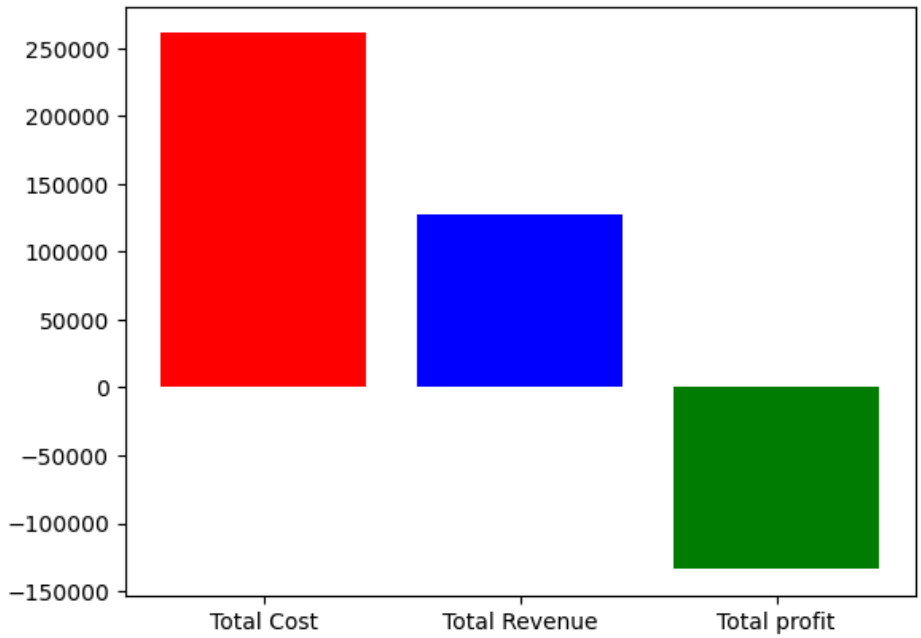
In [12]:
```
plt.figure()
costs_breakdown = food_data[['Delivery Fee', 'Payment Processing Fee', 'Discount Amount',"Refunds/Chargeb
plt.pie(costs_breakdown,labels=costs_breakdown.index,autopct='%1.1f%%')
plt.title('Proportion of Total Costs in Food Delivery')
plt.show()
```



The pie chart illustrates the breakdown of total costs into delivery fees, payment processing fees, and discount amounts. Discounts constitute a significant portion of the costs, suggesting that promotional and discount strategies might be heavily impacting overall profitability.

Now, let's compare total revenue, total costs, and total profit (net loss in our case):

In [13]:
```
Total=["Total Cost","Total Revenue","Total profit"]
Value=[(food_data["Cost per order"].sum()),(food_data["Revenue per order"].sum()),(food_data["Profit per
plt.figure()
plt.bar(Total,Value,color=["Red","Blue","Green"])
plt.show()
```



The bar chart compares the total revenue, total costs and total profit, and it also visually indicates a gap between revenue and cost and indicating that cost id surpassing the total revenue.

A New stratergy for Profit

From the Analysis we know that discounts are playing a major role in increasing the cost structure and resulting in a loss making situation, we need to find a safe spot for offering discounts and commissions. For analysing new profit statergies we need to observe the profit making orders more deeply for:

1. A new Commission rate based on profitable orders.
2. A new Discount rate based on profitable orders.

In [ ]:
```
#Calculating old discount and commission percentages
old_commission_percentage=(food_data["Commission Fee"]/food_data["Order Value"])*100
old_discount_percentage=(food_data["Discount Amount"]/food_data["Order Value"])*100
print("The old commission percentage ",old_commission_percentage.mean())
print("The old dicount percentage ",old_discount_percentage.mean())
```

```
The old commission percentage  19.750890168545332
The old dicount percentage  16.265
```

Now lets calculate new stratergy by considering only the profitable orders from dataset and then get the average from it

In [17]:
```
#Calculating new commission and discount percentages
#sorting onl the profitable orders
profitable_orders=food_data[food_data["Profit per order"]>0]
profitable_orders["Commission Percentage"]=(profitable_orders["Commission Fee"]/profitable_orders["Order 


print("new_commission_percentage",profitable_orders["Commission Percentage"].mean())

#directly using the discount percentage column for new average discount percentage
print("new_discount_percentage",profitable_orders["Discount Percentage"].mean())
```

```
new_commission_percentage 31.59734746146822
new_discount_percentage 5.7975460122699385
```

We can observe that the older/average commission rate is much lower as compared to the new commission rate and on the other hand the overall discount offered is on a much higher side when compared to new discount rate.

The new stratergy aims to apply commission rates around 31% and a discount rate of about 6% to make their business profitable

Now let's apply this suggested rates to the dataset and visualize the change in profitability.

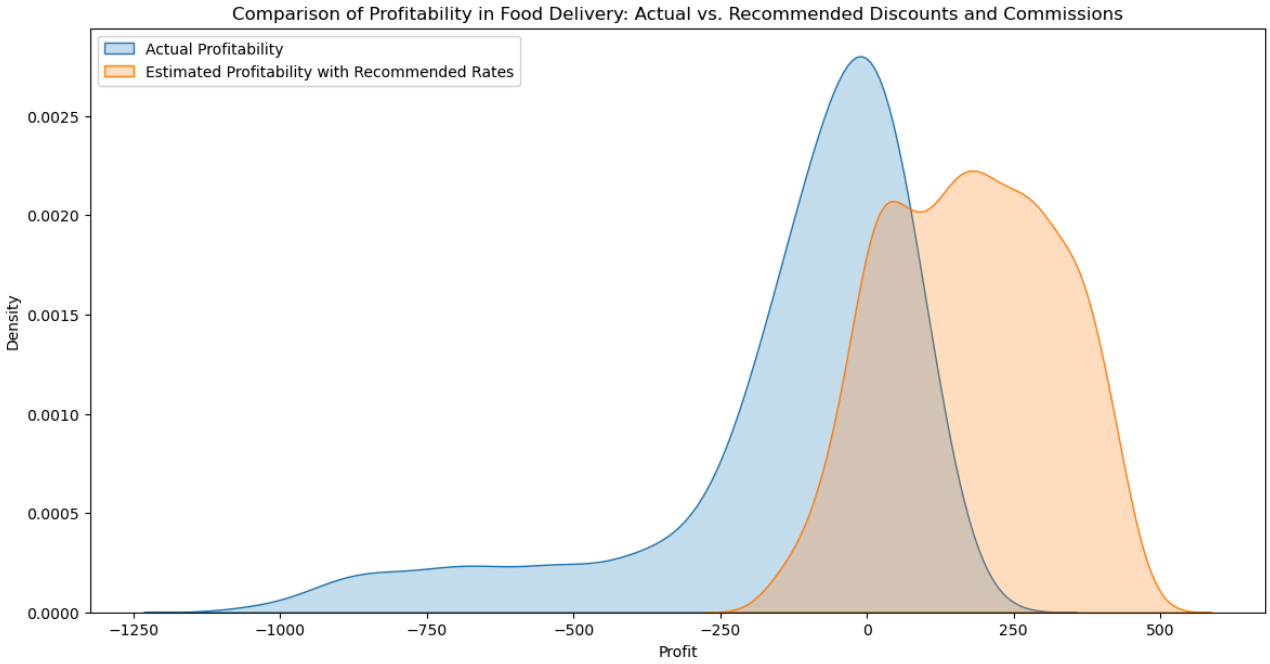
In [18]:
```
# simulate profitability with recommended discounts and commissions
recommended_commission_percentage = 31.0  # 31%
recommended_discount_percentage = 6.0    # 5.79% rounding off to 6

food_data['Predicted Commission Fee'] = food_data["Order Value"]*(recommended_commission_percentage/100)
food_data['Predicted Dicount Amount'] = food_data["Order Value"]*(recommended_discount_percentage/100)

#recalculating total costs and profit after applying predicted values
food_data["Predicted Total Cost"]=(food_data["Delivery Fee"]+food_data["Payment Processing Fee"]+
                                   food_data["Refunds/Chargebacks"]+food_data["Predicted Dicount Amount"]
food_data["Predicted Profit"]=food_data['Predicted Commission Fee']-food_data["Predicted Total Cost"]

import seaborn as sns
plt.figure(figsize=(14, 7))

sns.kdeplot(food_data["Profit per order"],fill=True,label='Actual Profitability')
sns.kdeplot(food_data["Predicted Profit"],fill=True,label='Estimated Profitability with Recommended Rates
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()
```



The visualization compares the distribution of profit per order using the old commission and discount rates versus the new applied rates.

The major part of the actual profitability was distributed on the negative side showing the loss making situation, while the predicted scenario suggests a shift of the overall profitability on a positive scale indicating that the suggested rates could lead towards the higher proportion of profitable orders.