



# FOOD DELIVERY COST & PROFITABILITY ANALYSIS

# PURPOSE OF THIS ANALYSIS

This project aims to evaluate the financial aspects of food delivery services by comparing costs incurred with revenue generated, to assess the overall profitability per order.

## ● Key Focus Areas

- Cost components (Commission, Delivery, Processing fees, Refunds)
- Revenue from orders
- Profit margins per order





# KEY METRICS ANALYZED

- **Order Value, Delivery Fee, Commission Fee, Refunds**
- **Total Costs, Revenue per Order, Profit per Order**

## DATASET

DataSet was imported from kaggle

### TOOLS USED

- SQL for data extraction
- Python for deeper analysis

## Q.1 TOTAL ORDERS AND REVENUE

```
1  -- Q1 Write a query to calculate total orders and revenue
2
3  • SELECT COUNT(*) AS Total_Orders, SUM(`Order Value`) AS Total_Revenue
4     FROM food_orders;
5
```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	Total_Orders	Total_Revenue
▶	1000	1053969

## Q.2 EXTRACT TOP 10 RESTAURANTS BY ORDER COUNT



```
1  -- Q2 Write a query to extract top 10 Restaurants by order count
2
3  • select `Restaurant ID`,count(*) as Order_count
4     from food_orders
5     group by `Restaurant ID`
6     order by order_count desc
7     limit 10
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	Restaurant ID	Order_count		
▶	R2317	6		
	R2016	5		
	R2804	5		
	R2726	5		
	R2523	5		
	R2083	5		
	R2028	4		
	R2348	4		
	R2933	4		
	R2520	4		







## Q.3 AVERAGE DELIVERY TIME AND REVENUE BY MONTH

```
1  -- Q3 Write a query to calculate average delivery time and revenue by month
2
3  • select date_format(`Order Date and Time`, '%Y-%m') as month,
4         avg(timestampdiff(minute, `Order Date and Time`, `Delivery Date and Time`)) as avg_delivery_time,
5         sum(`Order Value`) as Monthly_revenue
6  from food_orders
7  group by month
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	month	avg_delivery_time	Monthly_revenue
▶	2024-02	73.5590	207291
	2024-01	73.5789	846678

## Q.4 TOP 3 RESTAURANTS BY PROFIT MARGIN





```
1  -- Q4 Write a query to get top 3 Restaurants by profit margin
2
3  • select `Restaurant ID`,
4         sum(`Order Value`) as Total_Revenue,
5         sum(`Order Value` - (`Commission Fee` + `Delivery Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`)) as Total_Profit,
6         (sum(`Order Value` - (`Commission Fee` + `Delivery Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`))*100.0)/sum(`Order Value`) as Profit_Percentage
7  from food_orders
8  group by `Restaurant ID`
9  order by profit_percentage desc
10 limit 3
```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content:  | Fetch rows: 

	Restaurant ID	Total_Revenue	Total_Profit	Profit_Percentage
▶	R2198	1487	1412	94.95629
	R2584	1981	1875	94.64917
	R2882	1543	1452	94.10240

## Q.5 MOST COMMON PAYMENT METHOD

```
1  -- Q5 Write a query for the most common payment method
2
3  • select `Payment Method`, count(*) as total_payments
4     from food_orders
5     group by `Payment Method`
6     order by total_payments desc
7     limit 1
```




Result Grid |   Filter Rows:  | Export:  | Wrap Cell Content:  | Fetch rows: 

	Payment Method	total_payments
▶	Cash on Delivery	357



## Q.6 COMMISSION TO PROFIT RATIO PER RESTAURANTS

```
1  -- Q6 Write a query to find commission to profit ratio per restaurants
2
3  • select `Restaurant ID`,
4         sum(`Commission Fee`) as Total_commission,
5         sum(`Order Value` - (`Commission Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`)) as total_profit,
6         ( sum(`Commission Fee`)*100)/sum(`Order Value` - (`Commission Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`)) as commission_profit_ratio
7  from food_orders
8  group by `Restaurant ID`
```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	Restaurant ID	Total_commission	total_profit	commission_profit_ratio
▶	R2924	150	1717	8.7362
	R2054	416	3043	13.6707
	R2870	392	1189	32.9689
	R2642	368	3192	11.5288
	R2799	461	3740	12.3262
	R2777	179	1462	12.2435
	R2457	144	97	148.4536
	R2978	55	186	29.5699
	R2877	116	1499	7.7385
	R2161	357	1646	21.6889
	R2379	339	1778	19.0664
	R2992	61	1703	3.5819
	R2086	309	1850	16.7027
	R2475	250	1082	23.1054
	R2177	130	336	38.6905
	R2390	306	999	30.6306



## Q.7 MONTHLY GROWTH OF ORDERS AND PROFIT

```
1  -- Q7 Write a query for monthly growth of orders and profit
2
3  • with monthly as (
4      select date_format(`Order Date and Time`, '%Y-%m') as month,
5              count(*) as order_count,
6              sum(`Order Value` - (`Commission Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`)) as profit
7      from food_orders
8      group by month
9  )
10 select month,
11        order_count, (order_count - lag(order_count) over (order by month)) / lag(order_count) over (order by month) * 100 AS Growth_Rate_order,
12        profit, (profit - lag(profit) over (order by month)) / lag(profit) over (order by month) * 100 as Growth_rate_profit
13 from monthly
```

Result Grid   Filter Rows:   Export:   Wrap Cell Content:					
	month	order_count	Growth_Rate_order	profit	Growth_rate_profit
▶	2024-01	805	NULL	697368	NULL
	2024-02	195	-75.7764	171479	-75.4105

## Q.8 TIME TO DELIVERY PERFORMANCE

```
1  -- Q8 write a query for time to delivery performance
2
3  • select hour(`Order Date and Time`) as order_hour,
4         avg(timestampdiff(minute,`Order Date and Time`,`Delivery Date and Time`)) as average_delivery_time
5  from food_orders
6  group by order_hour
7  order by order_hour
```

Result Grid |  Filter Rows:  | Export:  | Wrap Cell Content: 

	order_hour	average_delivery_time
▶	0	75.4571
	1	76.2286
	2	74.0526
	3	68.7209
	4	75.6809
	5	72.3542
	6	67.0417
	7	72.8182
	8	72.2826
	9	69.3778
	10	68.0667
	11	76.3333
	12	81.2903
	13	72.8333
	14	75.8974
	15	78.4750



## Q.9 RETENTION ANALYSIS : RETURNING VS NEW CUSTOMERS

```
1  -- Q9 Customer Retention Analysis : Returning Vs New Customers
2
3  with customer_order as (
4      select `Customer ID`,
5      min(`Order Date and time`) as first_order
6      from food_orders
7      group by `Customer ID`
8  ), monthly_orders as(
9      select `Customer ID`,
10     date_format(`Order Date and Time`, '%Y-%m') as order_month
11     from food_orders)
12     select mo.order_month,
13     sum(case when mo.order_month = date_format(co.first_order, '%Y-%m') then 1 else 0 end) as new_customer,
14     sum(case when mo.order_month > date_format(co.first_order, '%Y-%m') then 1 else 0 end) as returning_customer
15     from monthly_orders mo
16     join customer_order co on mo.`Customer ID` = co.`Customer ID`
17     group by mo.order_month
18     order by mo.order_month
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	order_month	new_customer	returning_customer
▶	2024-01	0	805
	2024-02	0	195



# Q.10 SLIDING 7-DAY WINDOW METRICS

1

-- Q10 Write query for Sliding 7-Day Window Metrics

2

3

WITH Daily\_Orders AS (

4

select DATE(`Order Date and Time`) AS Order\_Date,

5

sum(`Order Value`) AS Daily\_Revenue,

6

sum(`Order Value` - (`Commission Fee` + `Delivery Fee` + `Payment Processing Fee` + `Refunds/Chargebacks`)) AS Daily\_Profit,

7

count(\*) AS Order\_Count

8

FROM food\_orders

9

GROUP BY Order\_Date

10

)

11

SELECT

12

d1.Order\_Date,

13

sum(d2.Daily\_Revenue) AS Revenue\_Last\_7\_Days,

14

sum(d2.Daily\_Profit) AS Profit\_Last\_7\_Days,

15

sum(d2.Order\_Count) AS Orders\_Last\_7\_Days

16

FROM Daily\_Orders d1

17

JOIN Daily\_Orders d2

18

ON d2.Order\_Date BETWEEN DATE\_SUB(d1.Order\_Date, INTERVAL 6 DAY) AND d1.Order\_Date

19

GROUP BY d1.Order\_Date

20

ORDER BY d1.Order\_Date;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Order_Date	Revenue_Last_7_Days	Profit_Last_7_Days	Orders_Last_7_Days
▶	2024-01-01	32380	26201	28
	2024-01-02	59648	48512	52
	2024-01-03	90370	73099	79
	2024-01-04	108952	87621	99
	2024-01-05	129681	104994	118
	2024-01-06	156081	125660	143
	2024-01-07	178166	142520	167

# PYTHON ANALYSIS

```
import numpy as np
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
```

[1]

```
food_order = pd.read_csv("food_orders.csv")
```

[2]

▷ ~

```
print(food_order.head())
```

[3]

...

	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

# PYTHON ANALYSIS

```
food_order.info()
```

```
[4]
```

```
... <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
```

```
food_order.describe()
```

```
[5]
```

```
...

```

	Order ID	Order Value	Delivery Fee	Commission Fee	Payment Processing Fee	Refunds/Chargebacks
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	1053.969000	28.620000	126.990000	29.832000	28.300000
std	288.819436	530.975339	16.958278	43.06405	11.627165	49.614228
min	1.000000	104.000000	0.000000	50.000000	10.000000	0.000000
25%	250.750000	597.750000	20.000000	90.000000	20.000000	0.000000
50%	500.500000	1038.500000	30.000000	127.000000	30.000000	0.000000
75%	750.250000	1494.000000	40.000000	164.000000	40.000000	50.000000
max	1000.000000	1995.000000	50.000000	200.000000	50.000000	150.000000



# PYTHON ANALYSIS

```
food_order['Order Date and Time'] = pd.to_datetime(food_order['Order Date and Time'])
food_order['Delivery Date and Time'] = pd.to_datetime(food_order['Delivery Date and Time'])

def extract_discount(discount_str) :
    if isinstance(discount_str,float):
        return float(discount_str)
    elif 'off' in discount_str:
        return float(discount_str.split(' ')[0])
    elif '%' in discount_str:
        return float(discount_str.split('%')[0])
    else:
        return 0.0

food_order['Discount Percentage'] = food_order['Discounts and Offers'].apply(lambda x : extract_discount(x))

food_order['Discount Amount'] = food_order.apply(lambda x: (x['Order Value'] * x['Discount Percentage'] / 100)
                                                    if x['Discount Percentage'] > 1
                                                    else x['Discount Percentage'], axis=1)

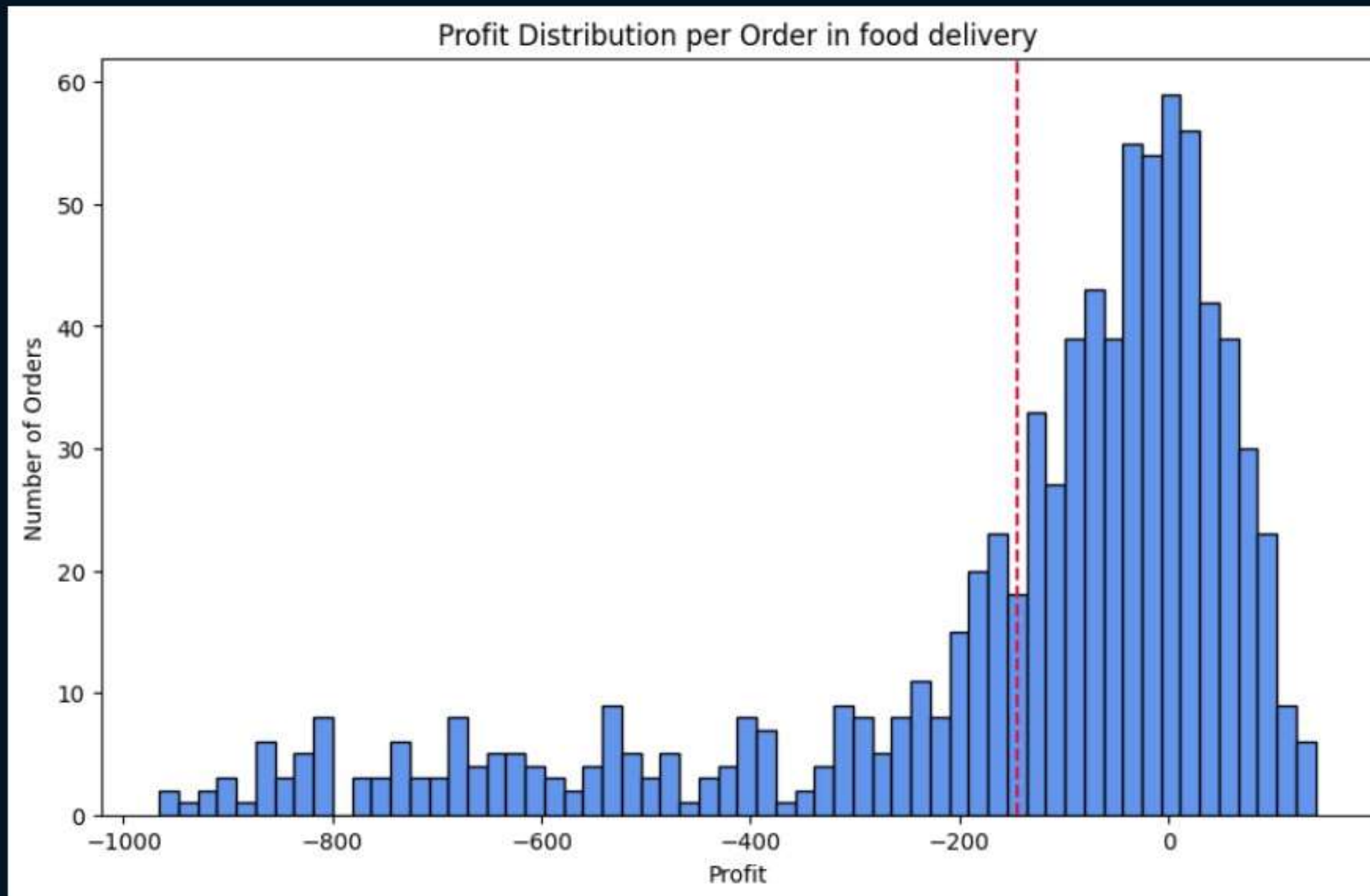
food_order['Discount Amount'] = food_order.apply(lambda x: x['Discount Amount'] if x['Discount Percentage'] <= 1
                                                    else x['Order Value'] * x['Discount Percentage'] / 100, axis=1)

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



# PYTHON ANALYSIS

```
plt.figure(figsize=(10,6))
plt.hist(food_order['Profit'],bins = 60,color='cornflowerblue',edgecolor = 'black')
plt.title('Profit Distribution per Order in food delivery')
plt.xlabel('Profit')
plt.ylabel('Number of Orders')
plt.axvline(food_order['Profit'].mean(),color='crimson',linestyle = 'dashed',linewidth='1.5')
plt.show()
```

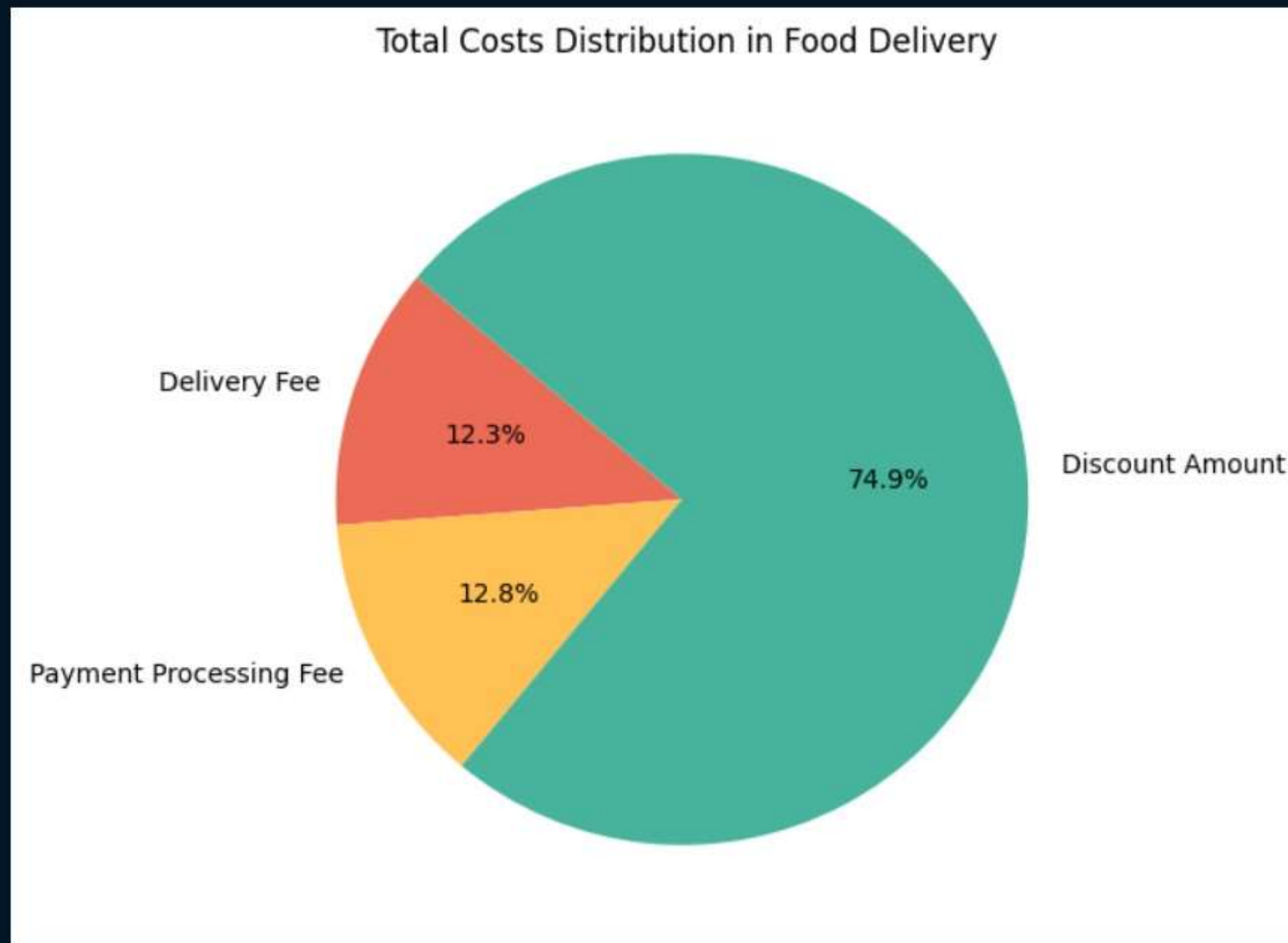


# PYTHON ANALYSIS

```
cost_distribution = food_order[['Delivery Fee','Payment Processing Fee','Discount Amount']].sum()
plt.figure(figsize=(6, 6))
plt.pie(cost_distribution,labels = cost_distribution.index,autopct='%1.1f%%',startangle=140,colors=['#EC6B56','#FFC154','#47B39C'])
plt.title(' Total Costs Distribution in Food Delivery')
plt.show()
```

[10]

...



# PYTHON ANALYSIS

```
totals = ['Total Cost', 'Total Revenue', 'Total Profit']  
value = [totalCost, totalRevenue, totalProfit]
```

```
plt.figure(figsize=(8,6))  
plt.bar(totals,value,color=['blue','green','red'])  
plt.title("Total Cost, Revenue and Profit")  
plt.ylabel('Amount (INR)')  
plt.show()
```

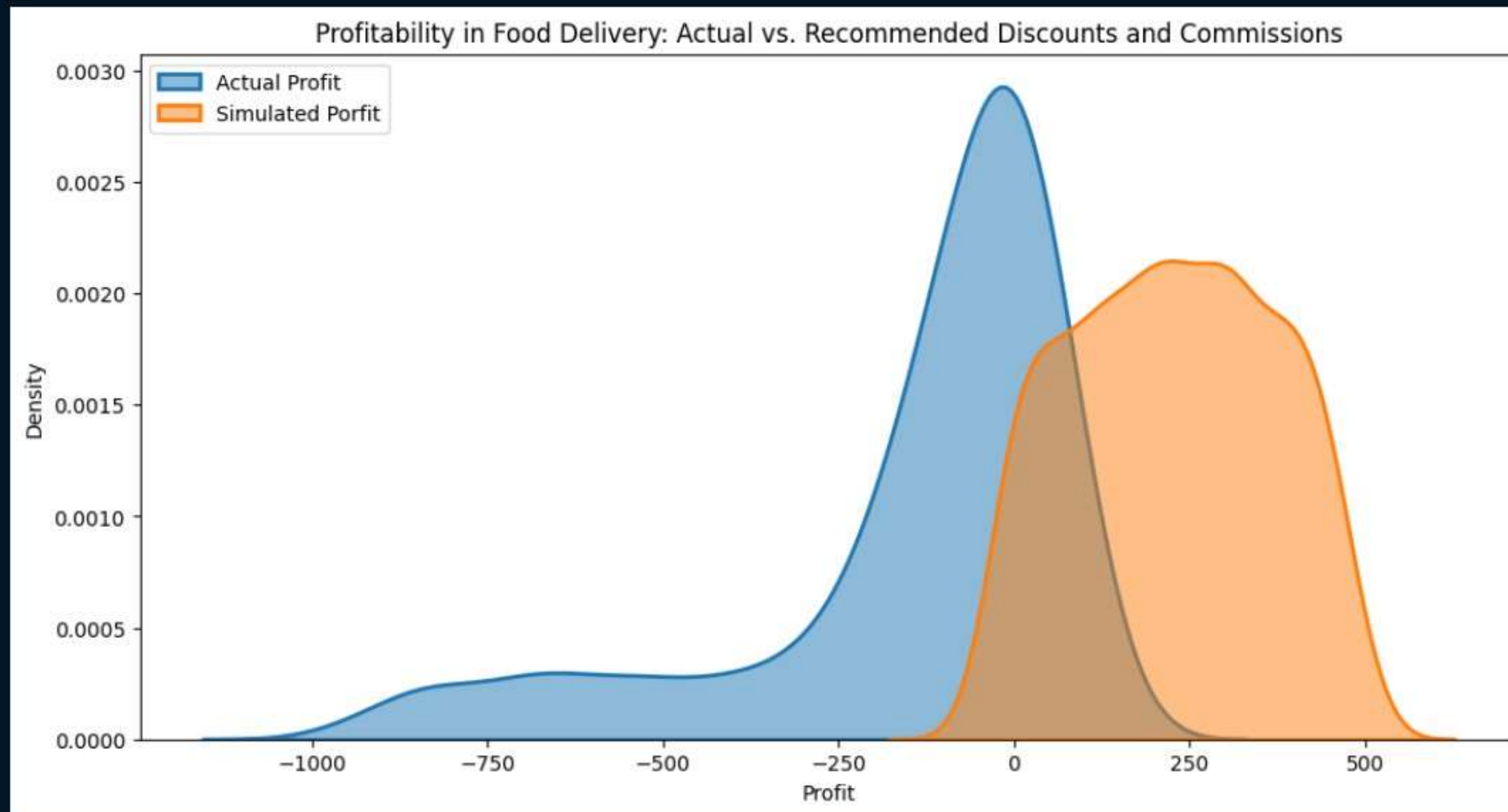


# PYTHON ANALYSIS

```
plt.figure(figsize=(12, 6))
sns.kdeplot(food_order['Profit'],label='Actual Profit',fill=True,alpha=0.5,linewidth = 2)
sns.kdeplot(food_order['Simulated Profit'],label='Simulated Porfit',fill=True,alpha=0.5,linewidth=2)

plt.title('Profitability in Food Delivery: Actual vs. Recommended Discounts and Commissions')
plt.xlabel('Profit')
plt.ylabel('Density')
plt.legend(loc='upper left')
plt.show()
```

✓ 0.2s







THANK

to

be

and

to

the

you

