

PROJECT: FOOD ORDER NEW DELHI DATASET CLEANING & VISUALIZATION

STEP1 : Import libraries

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

STEP2 : Load a dataset

```
1 df = pd.read_csv("/content/food_orders_new_delhi (1).csv")
```

STEP3 : Data cleaning

1. Read first 5 row

```
1 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
3	4	C6191	R2642	2024-01-16 22:52:49	2024-01-16 23:38:49	1463	50	Cash on Delivery	NaN	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

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

2. Read last 5 row

```
1 df.tail()
```

	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
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	NaN	110	42	0
997	998	C5926	R2837	2024-01-21 09:43:19	2024-01-21 10:44:19	553	20	Cash on Delivery	NaN	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

3. Find statistical information

```
1 df.describe()
```

	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


4. find all type information

```
1 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
```


5. Check duplicated values

```
1 df.duplicated().sum()
```

 np.int64(0)


6. Check duplicated value from order Id column

```
1 df["Order ID"].duplicated().sum()
```

 np.int64(0)

7. Check Null values

```
1 df.isna().sum()
```




	0
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	185
Commission Fee	0
Payment Processing Fee	0
Refunds/Chargebacks	0

dtype: int64

8. Handle Null values

```
1 df["Discounts and Offers"].fillna("None", inplace=True)
```


 <ipython-input-10-01f7341076c0>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the op

```
df["Discounts and Offers"].fillna("None", inplace=True)
```

9. Re-check Null values

```
1 df.isna().sum()
```



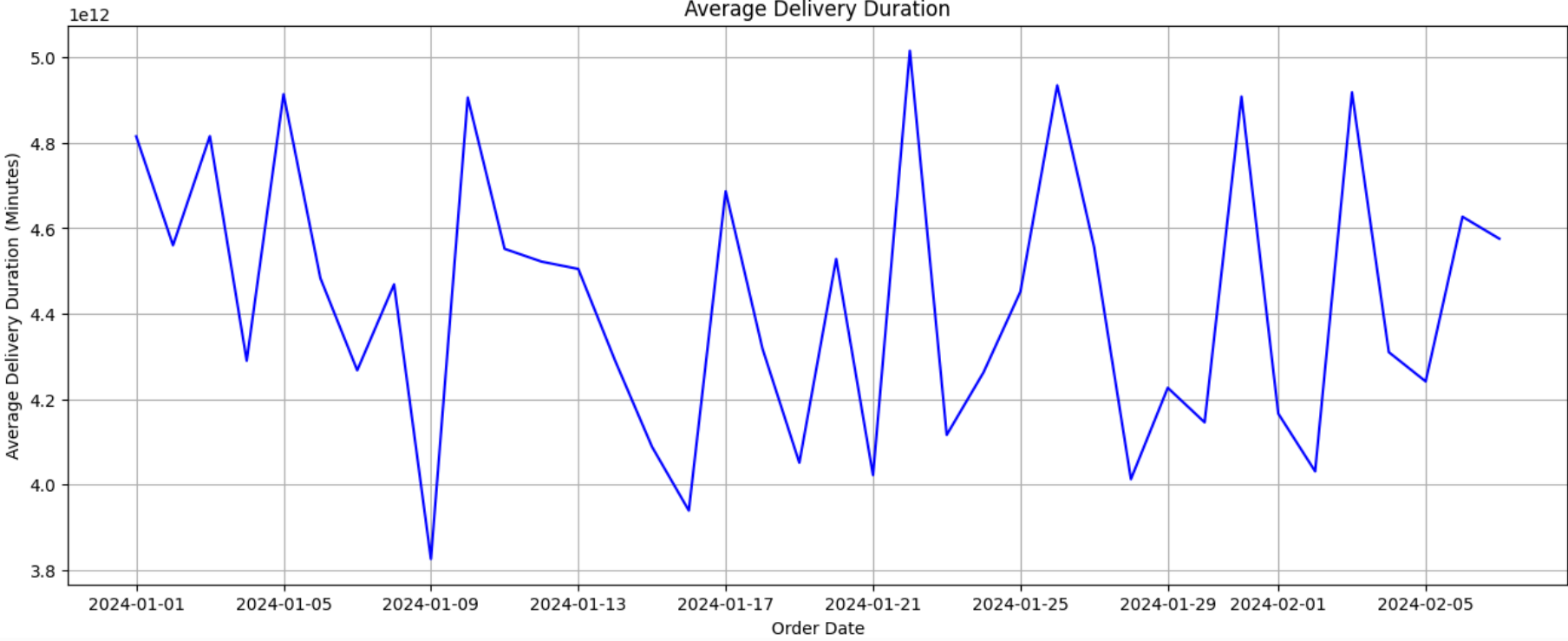
	0
Order ID	0
Customer ID	0
Restaurant ID	0
Order Date and Time	0
Delivery Date and Time	0
Order Value	0
Delivery Fee	0
Payment Method	0
Discounts and Offers	0
Commission Fee	0
Payment Processing Fee	0
Refunds/Chargebacks	0

dtype: int64

✓ STEP4 : Data visualization

QUE1 : How does the average delivery duration vary across the observed dates?

```
1 # convert date-time column into data-time format
2 df["Order Date and Time"] = pd.to_datetime(df["Order Date and Time"])
3 df["Delivery Date and Time"] = pd.to_datetime(df["Delivery Date and Time"])
4
5 # calculate delivery duration (in minutes)
6 df["Delivery Duration"] = (df["Delivery Date and Time"] - df["Order Date and Time"])
7
8 # Groupby day and calculate average delivery duration
9 df["Order Day"] = df["Order Date and Time"].dt.date
10 daily_avg = df.groupby("Order Day")["Delivery Duration"].mean()
11
12 # plot
13 plt.figure(figsize=(16, 6))
14 plt.plot(daily_avg.index, daily_avg.values, color='blue')
15 plt.title(label='Average Delivery Duration')
16 plt.xlabel('Order Date')
17 plt.ylabel('Average Delivery Duration (Minutes)')
18 plt.grid()
19 plt.show()
```



CONCLUSION : The chart depicts the fluctuating average delivery duration in minutes over the period from January 1, 2024, to February 5, 2024.

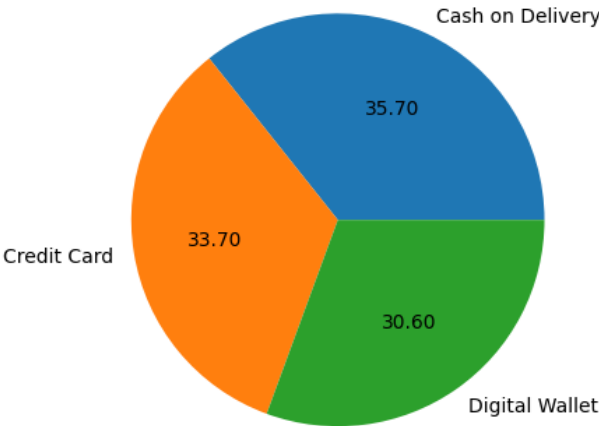
QUE2 : What are the percentage shares of each payment method depicted in the pie chart?

```
1 payment_dist = df.groupby("Payment Method")["Payment Method"].count()
2 print(payment_dist)
3 plt.pie(payment_dist, labels=payment_dist.index, autopct="%.2f", startangle=0)
4 plt.title("Distributions of Payment Methods")
5 plt.show()
```



```
Payment Method
Cash on Delivery    357
Credit Card        337
Digital Wallet      306
Name: Payment Method, dtype: int64
```

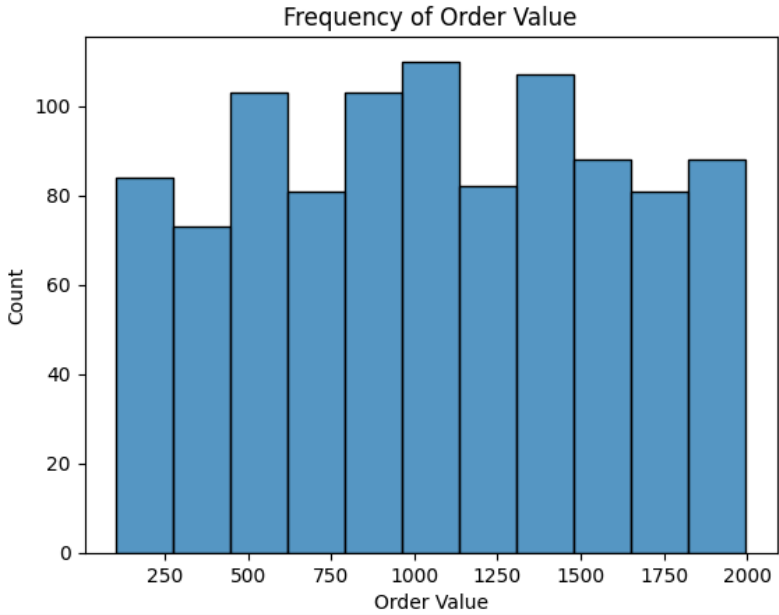
Distributions of Payment Methods



CONCLUSION : The pie chart shows the distribution of payment methods, with Cash on Delivery (35.70%), Credit Card (33.70%), and Digital Wallet (30.60%) as the three options.

QUE3 : Which order value range has the highest frequency in the chart?

```
1 sns.histplot(data=df, x="Order Value")
2 plt.title("Frequency of Order Value")
3 plt.show()
```



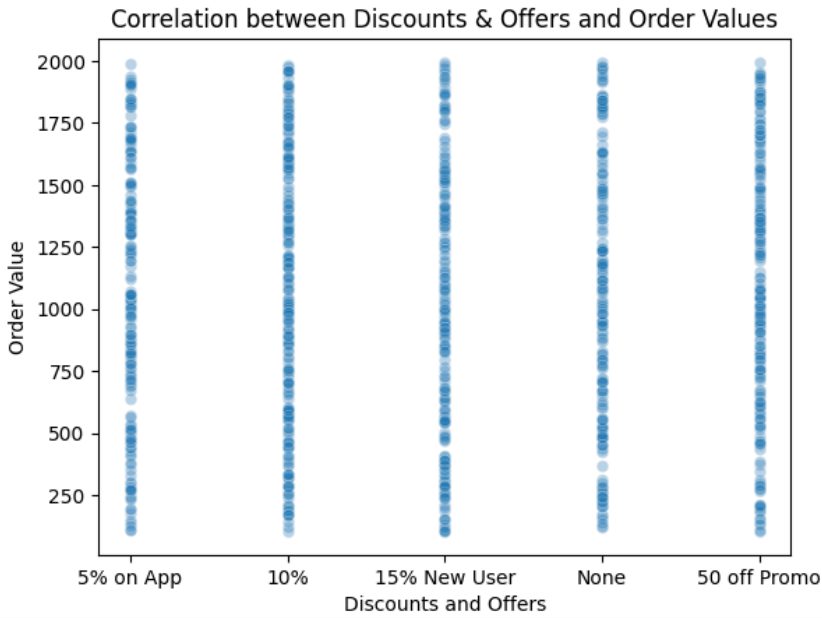
CONCLUSION : The chart illustrates the frequency of order values, showing that the 1000-value range has the highest frequency.

QUE4 : How do order values differ across various discount and offer categories represented in the chart?

```
1 sns.scatterplot(data=df, y="Order Value", x="Discounts and Offers",
2                 palette="cividis", alpha=0.3)
3 plt.title("Correlation between Discounts & Offers and Order Values")
4 plt.show()
```

```
plt.show()
```

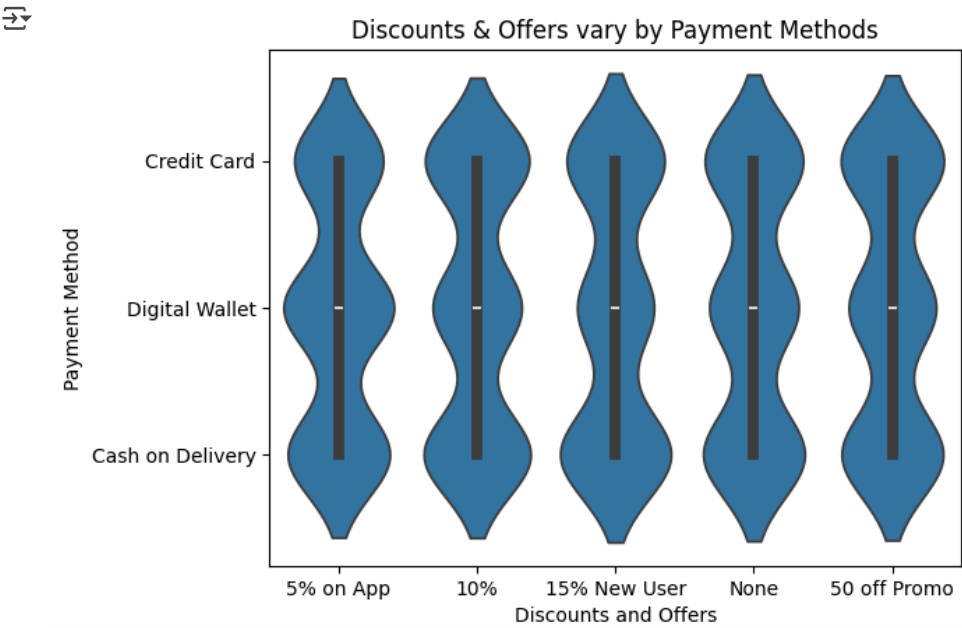
```
<ipython-input-18-a6827bd12c41>:1: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
sns.scatterplot(data=df, y="Order Value", x="Discounts and Offers", palette="cividis", alpha=0.3)
```



CONCLUSION : The chart demonstrates the relationship between discounts and offers and the corresponding order values, showing varied impacts across discount categories.

QUE5 : How do the discounts and offers vary across payment methods in the chart?

```
1 sns.violinplot(data=df, x="Discounts and Offers", y="Payment Method")
2 plt.title("Discounts & Offers vary by Payment Methods")
3 plt.show()
```



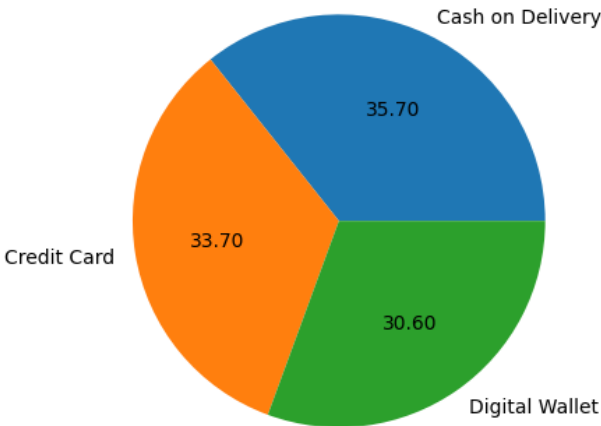
CONCLUSION : The chart shows how discounts and offers are distributed across different payment methods, such as Credit Card, Digital Wallet, and Cash on Delivery.

QUE6 : What is the percentage distribution of refunds/chargebacks among the payment methods represented in the chart?

```
1 refund = df.groupby("Payment Method")["Refunds/Chargebacks"].count()
2 print(refund)
3 plt.pie(refund, labels=refund.index, autopct="%.2f", startangle=0)
4 plt.title("Proportions of Refunds/Chargebacks by Payment Methods")
5 plt.show()
```

```
Payment Method
Cash on Delivery    357
Credit Card        337
Digital Wallet      306
Name: Refunds/Chargebacks, dtype: int64

Proportions of Refunds/Chargebacks by Payment Methods
```

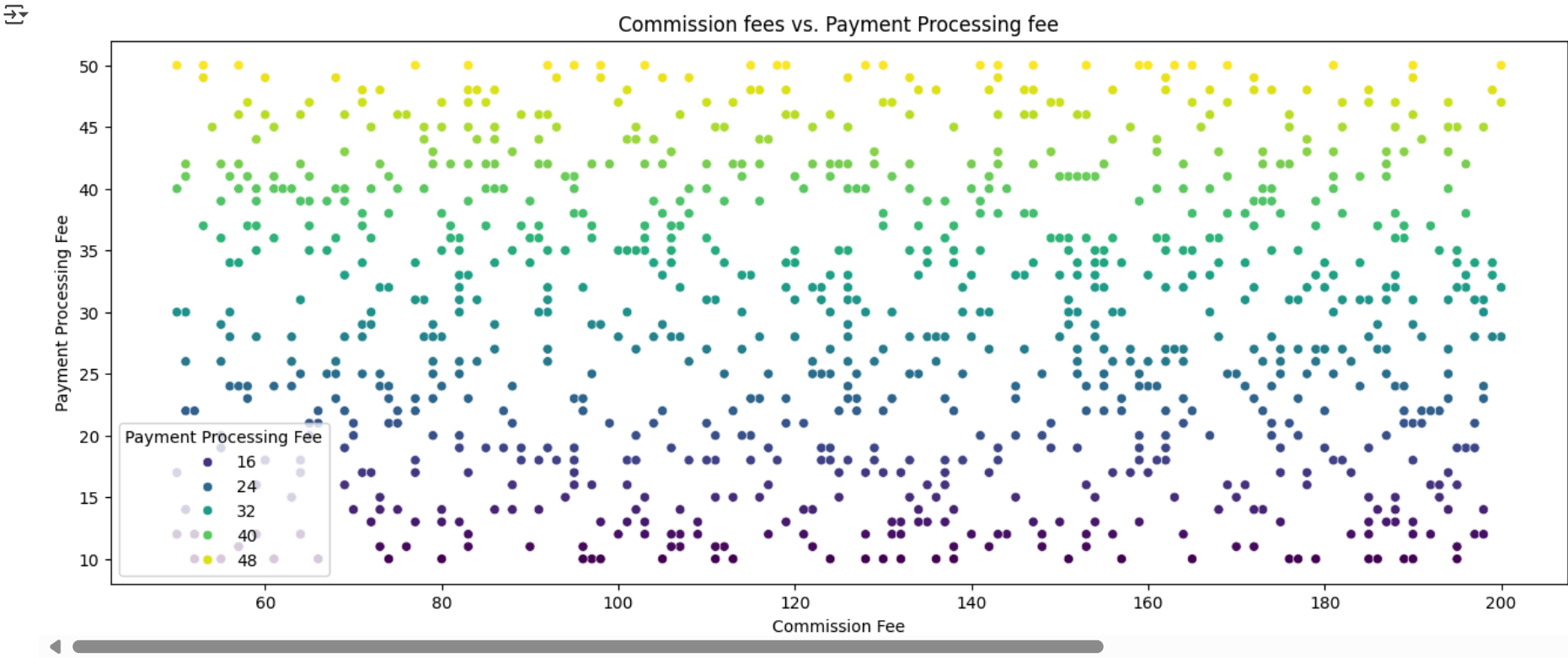


CONCLUSION : The chart highlights the proportions of refunds/chargebacks distributed across payment methods, showing Cash on Delivery at 35.70%, Credit Card at 33.70%, and Digital Wallet at 30.60%.

QUE7 : How does the scatter plot illustrate the correlation between commission fees and payment processing fees?

```
1 plt.figure(figsize=(16, 6))
2 sns.scatterplot(data=df, x="Commission Fee", y="Payment Processing Fee",
3   hue="Payment Processing Fee", palette="viridis")
4 plt.title("Commission fees vs. Payment Processing fee")
```

```
plt.show()
```



CONCUSION : The chart displays a scatter plot showing the relationship between commission fees and payment processing fees, with data points color-coded based on payment processing fee values.

QUE8 : Which discount type has the highest percentage in the chart?

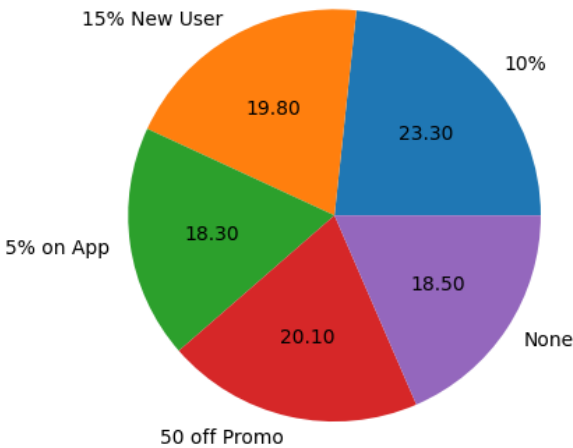
```
1 offer = df.groupby("Discounts and Offers")["Discounts and Offers"].count()
2 print(offer)
3 plt.pie(offer, labels=offer.index, autopct="%.2f", startangle=0)
4 plt.title("Proportions of Discounts and Offers")
5 plt.show()
```

Discounts and Offers

10%	233
15% New User	198
5% on App	183
50 off Promo	201
None	185

Name: Discounts and Offers, dtype: int64

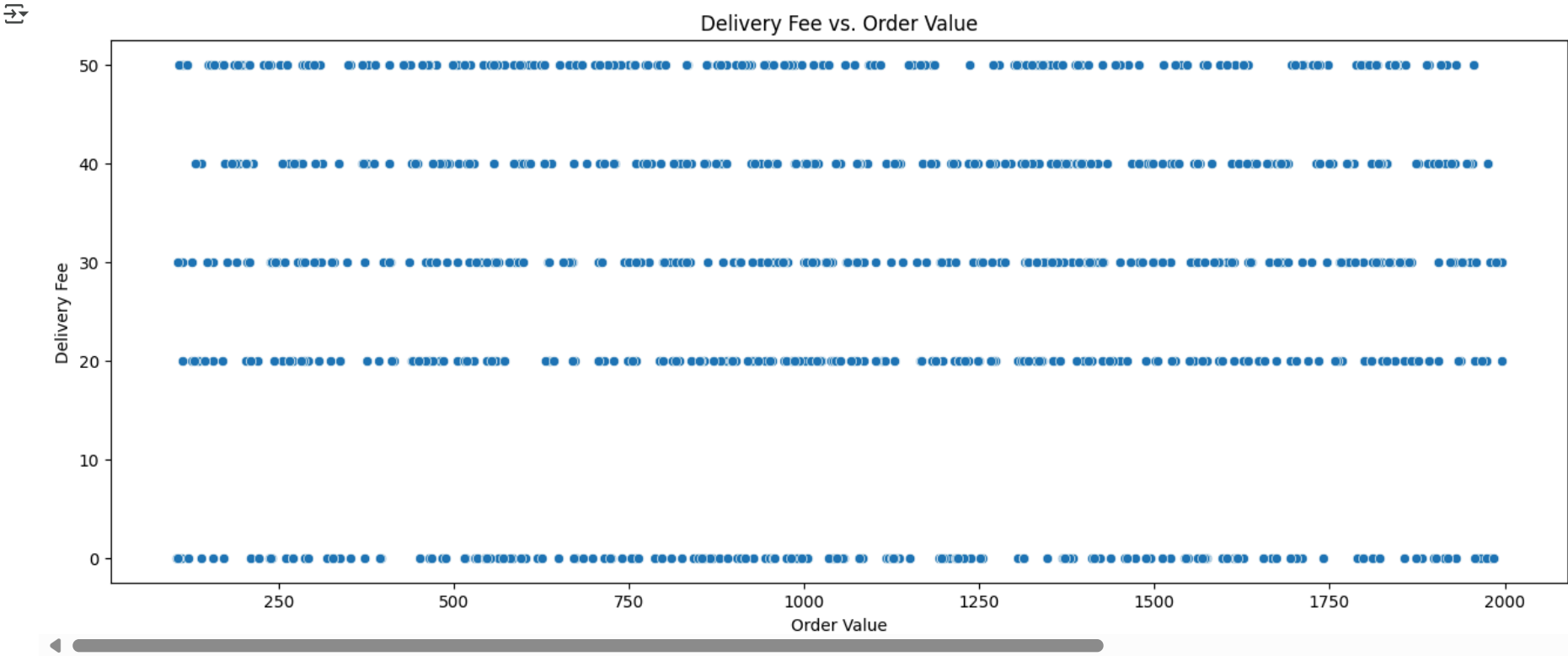
Proportions of Discounts and Offers



CONCLUSION : The chart represents proportions of different discount types, with the highest being 23.30% for a 10% discount and the lowest at 18.30% for a 5% discount on the app.

QUE9 : How does the scatter plot illustrate fixed delivery fees at specific values relative to order values?

```
1 plt.figure(figsize=(16, 6))
2 sns.scatterplot(data=df, x="Order Value", y="Delivery Fee")
3 plt.title("Delivery Fee vs. Order Value")
4 plt.show()
```



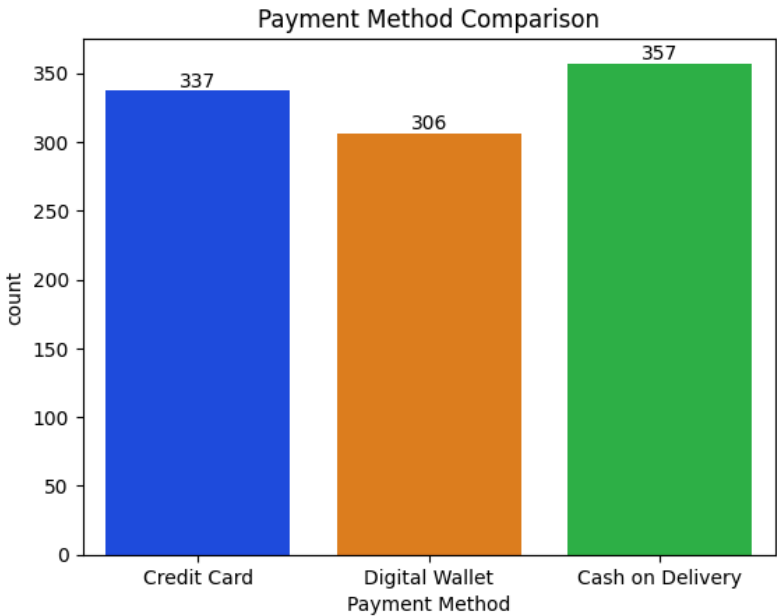
CONCLUSION : The scatter plot reveals fixed delivery fees (0, 20, 30, 40, and 50) distributed across varying order values.

QUE10 : Which payment method has the highest count in the bar chart?

```
1 payment = sns.countplot(data=df, x="Payment Method", palette="bright")
2 for i in range(0, 3):
3     payment.bar_label(payment.containers[i])
4 plt.title("Payment Method Comparison")
5 plt.show()
```

<ipython-input-29-61f597acb321>:1: FutureWarning: Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
payment = sns.countplot(data=df, x="Payment Method", palette="bright")
```



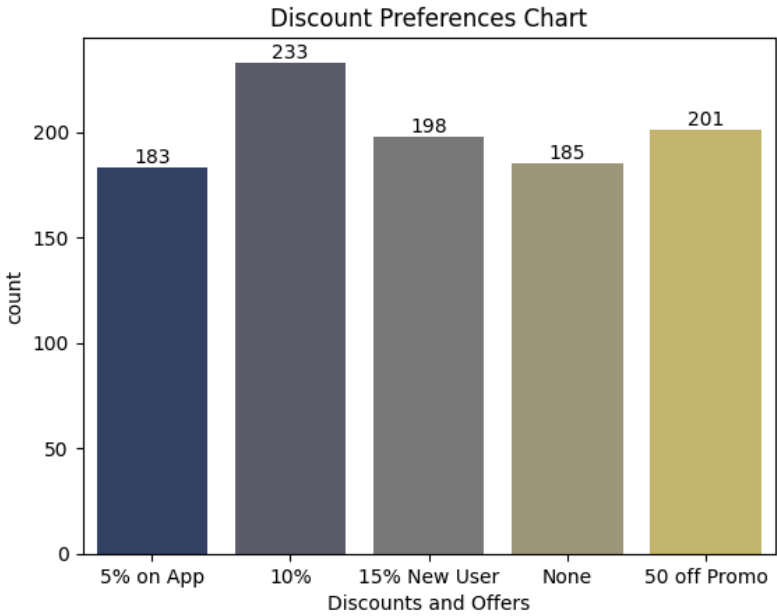
CONCLUSION : The bar chart shows the count of three payment methods: Cash on Delivery (357), Credit Card (337), and Digital Wallet (306), with Cash on Delivery being the most commonly used.

QUE11 : Which discount or offer has the highest count of users according to the chart?

```
1 payment = sns.countplot(data=df, x="Discounts and Offers", palette="cividis")
2 for i in range(0, 5):
3     payment.bar_label(payment.containers[i])
4 plt.title("Discount Preferences Chart")
5 plt.show()
```

<ipython-input-30-d2cef9e5d0c2>:1: FutureWarning: Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
payment = sns.countplot(data=df, x="Discounts and Offers", palette="cividis")
```



CONCLUSION : The bar chart shows the count of users availing different discounts/offers, with the highest count being 233 for the 10% discount.

STEP5 : Save clean data

```
1 df.to_csv("Food Order ND Cleaned Data.csv", index=False)
2 print("Data Cleaning & Visualized Completed...")
3 print("Food Order New Delhi data Cleaning & Visualized Project Done!...")
4 print()
```

Data Cleaning & Visualized Completed...
Food Order New Delhi data Cleaning & Visualized Project Done!...

Titanic Data Cleaning & Visualization Project Completed

