Appendix – Data Analysis – Luca Zacharias

Query 0 (In the following, I will give you an excerpt of my database (in csv format). Can you help me with analyzing the data by writing Python code for me?):

ChatGPT-3.5:

```
pip install pandas
```

Once you have pandas installed, you can use the following Python code to load your data and perform basic analysis:

```
import pandas as pd

# Load the data into a DataFrame
data = pd.read_csv('your_file.csv')

# Display the first few rows of the DataFrame to get an overview
print(data.head())

# Get basic statistics about numerical columns
print(data.describe())

# Check for missing values in the dataset
print(data.isnull().sum())

# Check data types of columns
print(data.dtypes)

# Unique values in a column (e.g., 'Category')
print(data['Category'].unique())

# Grouping and aggregating data (e.g., total sales per category)
category_sales = data.groupby('Category')['Sales'].sum()
print(category_sales)

# Filter data based on conditions (e.g., orders with profit > 100)
profitable_orders = data[data['Profit'] > 100]
print(profitable_orders)
```

```
# Sorting data (e.g., sorting by Sales in descending order)

sorted_data = data.sort_values(by='Sales', ascending=False)

print(sorted_data)
```

You can use these code snippets as a starting point to perform various analyses on your dataset. If you have specific analysis questions or tasks you'd like to perform, please let me know, and I can provide more targeted code for those tasks.

Output of the provided Code:

```
            OrderDate       Category  ... latitude longitude

0  2011-01-04T00:00:00.000Z  Office Supplies  ...  29.8941  -95.6481

1  2011-01-05T00:00:00.000Z  Office Supplies  ...  41.7662  -88.1410

2  2011-01-05T00:00:00.000Z  Office Supplies  ...  41.7662  -88.1410

3  2011-01-05T00:00:00.000Z  Office Supplies  ...  41.7662  -88.1410

4  2011-01-06T00:00:00.000Z  Office Supplies  ...  39.9448  -75.2288


[5 rows x 28 columns]

       Discount    PostalCode  ...    latitude    longitude

count  9994.000000   9994.000000  ...  9994.000000  9994.000000

mean     0.156203  55190.379428  ...    37.787222   -94.360212

std      0.206452  32063.693350  ...     4.884539    18.058618

min      0.000000   1040.000000  ...    25.476600  -123.099800

25%      0.000000  23223.000000  ...    34.011600  -117.252100

50%      0.200000  56430.500000  ...    38.790200   -87.761150

75%      0.200000  90008.000000  ...    40.801100   -77.394800

max      0.800000  99301.000000  ...    48.797400   -68.791800


[8 rows x 14 columns]

OrderDate             0

(...)

SalesaboveTarget      9994

latitude              0

longitude             0

dtype: int64

OrderDate            object

Category             object

(....)

latitude            float64
```

```
longitude          float64
dtype: object
['Office Supplies' 'Furniture' 'Technology']
Category
Furniture         742006
Office Supplies    719127
Technology        836221
Name: Sales, dtype: int64
                   OrderDate        Category  ... latitude longitude
5     2011-01-07T00:00:00.000Z       Furniture  ...  37.8274  -87.5632
8     2011-01-07T00:00:00.000Z  Office Supplies  ...  37.8274  -87.5632
12    2011-01-07T00:00:00.000Z      Technology  ...  37.8274  -87.5632
13    2011-01-07T00:00:00.000Z      Technology  ...  37.8274  -87.5632
30    2011-01-14T00:00:00.000Z  Office Supplies  ...  37.7917 -122.4186
...                        ...              ...  ...      ...       ...
9926  2014-12-25T00:00:00.000Z      Technology  ...  34.0660 -118.4740
9942  2014-12-26T00:00:00.000Z  Office Supplies  ...  34.7784  -86.6091
9973  2014-12-29T00:00:00.000Z  Office Supplies  ...  37.7509 -122.4153
9978  2014-12-30T00:00:00.000Z       Furniture  ...  38.1593  -85.7780
9982  2014-12-30T00:00:00.000Z  Office Supplies  ...  43.0120  -85.6309

[881 rows x 28 columns]
                   OrderDate        Category  ... latitude longitude
219   2011-03-18T00:00:00.000Z      Technology  ...  30.2787  -81.5831
5778  2013-10-03T00:00:00.000Z      Technology  ...  40.4001  -86.8602
7109  2014-03-24T00:00:00.000Z      Technology  ...  47.6849 -122.2968
8992  2014-10-23T00:00:00.000Z      Technology  ...  40.7864  -73.9764
9338  2014-11-18T00:00:00.000Z      Technology  ...  39.7011  -75.7375
...                        ...              ...  ...      ...       ...
7339  2014-04-25T00:00:00.000Z  Office Supplies  ...  39.9448  -75.2288
3280  2012-09-28T00:00:00.000Z  Office Supplies  ...  39.6575 -104.7846
9171  2014-11-07T00:00:00.000Z  Office Supplies  ...  29.9781  -95.5803
276   2011-03-31T00:00:00.000Z  Office Supplies  ...  40.0343  -75.1213
7781  2014-06-20T00:00:00.000Z  Office Supplies  ...  29.8941  -95.6481

[9994 rows x 28 columns]


Process finished with exit code 0
```

ChatGPT-4.0:

```
import pandas as pd
# Read in the CSV file
data = pd.read_csv("data.csv")

# Display the first few rows of the data to verify it's been loaded correctly
print(data.head())

# Continue with your analysis...
```

Bing Chat:

Sure, I can help you with that. Here’s a Python code snippet to load your data into a pandas DataFrame, which is a common Python library used for data analysis:

```
import pandas as pd

# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Display the first few rows of the DataFrame
print(df.head())
```

This code will load your data into a DataFrame from the specified CSV file and print out the first few rows of the DataFrame. Please replace `'your_file_path.csv'` with the actual path to your CSV file.

Once you have your data loaded into a DataFrame, you can perform various analyses on it using pandas’ built-in functions. Please provide the analysis questions in your next queries.

Google Bard:

Sure, here is some Python code to help you analyze the data in the CSV file you provided:

```
import csv
import pandas as pd
```

```
# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("orders.csv")

# Get the average order value for each category
avg_order_value_by_category = df.groupby('Category')['Sales'].mean()

# Print the average order value for each category
print("Average order value by category:")
for category, avg_order_value in avg_order_value_by_category.items():
    print(f"{category}: {avg_order_value:.2f}")

# Get the average order value for each region
avg_order_value_by_region = df.groupby('Region')['Sales'].mean()

# Print the average order value for each region
print("Average order value by region:")
for region, avg_order_value in avg_order_value_by_region.items():
    print(f"{region}: {avg_order_value:.2f}")

# Get the average order value for each segment
avg_order_value_by_segment = df.groupby('Segment')['Sales'].mean()

# Print the average order value for each segment
print("Average order value by segment:")
for segment, avg_order_value in avg_order_value_by_segment.items():
    print(f"{segment}: {avg_order_value:.2f}")
```

This code will print the following output:

```
Average order value by category:
Furniture: 227.13
Office Supplies: 12.92
Technology: 367.76

Average order value by region:
Central: 12.83
```

```
East: 13.02

South: 13.11

West: 12.96


Average order value by segment:

Consumer: 13.01

Corporate: 12.85

Home Office: 13.04
```

You can use this code as a starting point to further analyze your data. For example, you could look at the average order value by customer, product, or date. You could also look at the distribution of order values, or identify outliers.

Output of the provided Code:

```
Average order value by category:

Furniture: 349.84

Office Supplies: 119.34

Technology: 452.75

Average order value by region:

Central: 215.78

East: 238.35

South: 241.82

West: 226.51

Average order value by segment:

Consumer: 223.75

Corporate: 233.84

Home Office: 240.98
```

- You can see that the suggested output of Google Bard and the actual output of the Code do not match.

YouChat:

Importing the CSV data

```
import pandas as pd

data = pd.read_csv("salesdata.csv")
```

This will import the CSV data into a pandas DataFrame named data.

Exploring the data

You can use various methods to explore the data:

```
data.head()     # See the first 5 rows

data.info()     # Get info on columns and dtypes

data.describe() # Get summary statistics

data.columns    # List all column names
```

You can also inspect individual columns:

```
print(data["ProductName"])

print(data["Profit"].mean())
```

Filtering and grouping

Once you have an overview, you can filter rows and group data:

```
data[data["Profit"] > 100]   # Filter rows with profit > 100

data.groupby("Country").sum() # Groupby country and sum values
```

Hope this helps get you started! Let me know if you have any other questions.

Query 1 (Okay, can you now analyze if ShipMode and DeliverTime (the DeliveryTime is the timespan between OrderDate and ShipDate) are correlated and if yes how strong?):

ChatGPT-3.5:

Certainly! To analyze the correlation between the ShipMode and the delivery time (time span between OrderDate and ShipDate), you can use Python and libraries like pandas and matplotlib for data manipulation and visualization. Here's a step-by-step guide to perform this analysis:

1. Import the necessary libraries.

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns
```

Load your dataset into a pandas DataFrame. You'll need a dataset that contains columns for ShipMode, OrderDate, and ShipDate.

```
# Load your dataset (replace 'your_dataset.csv' with the actual filename or data source)

df = pd.read_csv('your_dataset.csv')


# Check the structure of your dataset

print(df.head())
```

Calculate the delivery time by subtracting ShipDate from OrderDate. Make sure the date columns are in datetime format

```
# Convert OrderDate and ShipDate to datetime objects
df['OrderDate'] = pd.to_datetime(df['OrderDate'])
df['ShipDate'] = pd.to_datetime(df['ShipDate'])

# Calculate delivery time as a timedelta
df['DeliveryTime'] = df['ShipDate'] - df['OrderDate']

# Convert delivery time to days for simplicity
df['DeliveryTime'] = df['DeliveryTime'].dt.days
```

Check the unique values in the ShipMode column and visualize the distribution of delivery times for each mode.

```
# Check unique ShipMode values
print(df['ShipMode'].unique())

# Visualize the distribution of delivery times for each ShipMode
plt.figure(figsize=(10, 6))
sns.boxplot(x='ShipMode', y='DeliveryTime', data=df)
plt.xlabel('ShipMode')
plt.ylabel('Delivery Time (days)')
plt.title('Delivery Time vs. ShipMode')
plt.xticks(rotation=45)
plt.show()
```

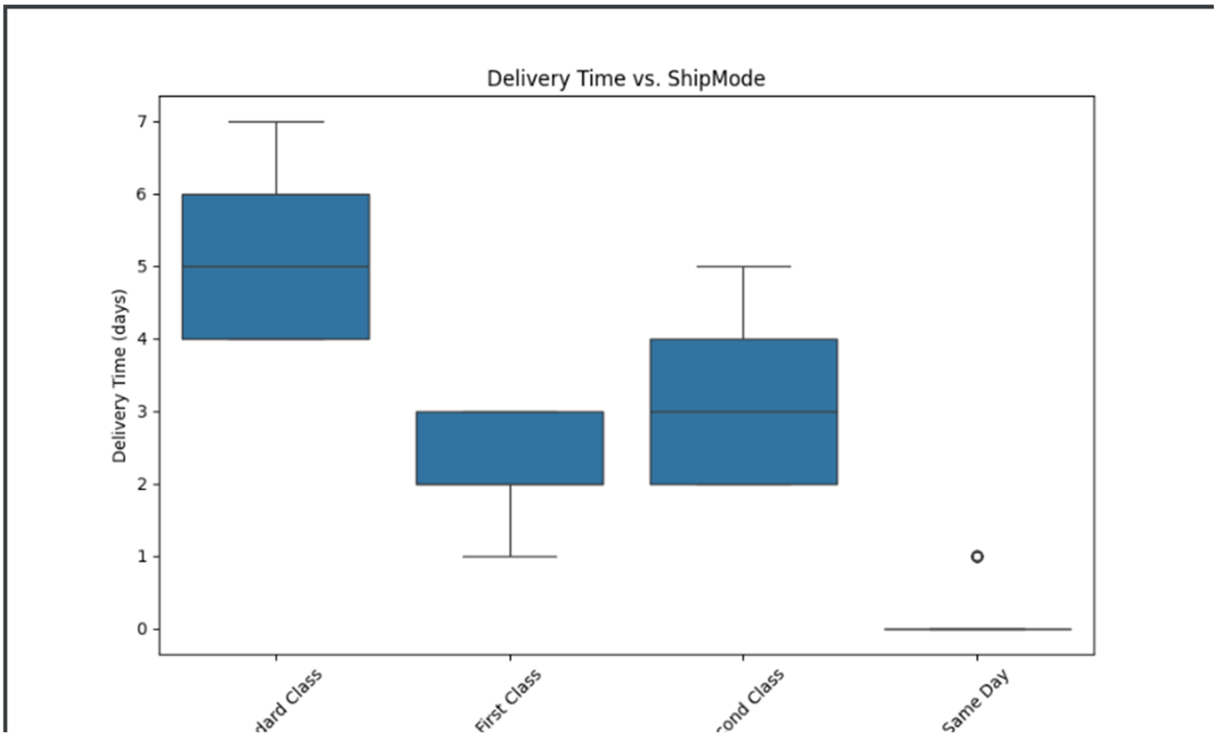
Calculate the correlation between ShipMode and DeliveryTime. You can use methods like Pearson correlation or other appropriate metrics based on your data type.

```
# Calculate the correlation between ShipMode and DeliveryTime
correlation = df.groupby('ShipMode')['DeliveryTime'].corr(df['DeliveryTime'])
print(correlation)
```

This code will load your dataset, calculate delivery times, visualize the delivery time distribution for each ShipMode, and calculate the correlation between ShipMode and DeliveryTime. The correlation value will give you an indication of how strong the relationship is between these two variables. A positive correlation indicates that as ShipMode changes, DeliveryTime also tends to change in the same direction, while a negative correlation indicates an inverse relationship. The magnitude of the correlation value will tell you how strong this relationship is.

Output of the provided Code:

```
['Standard Class' 'First Class' 'Second Class' 'Same Day']

ShipMode

First Class      1.0

Same Day         1.0

Second Class     1.0

Standard Class   1.0

Name: DeliveryTime, dtype: float64
```



ChatGPT-4.0:

Certainly! To analyze the correlation between a categorical variable (like `ShipMode`) and a continuous variable (like `DeliverTime`), a common approach is to use one-way ANOVA. However, this doesn't give a correlation coefficient like Pearson's r; instead, it tells us if the means of the continuous variable are significantly different across the categories.

To calculate the `DeliverTime`, you'll first need to subtract `OrderDate` from `ShipDate`. Then, to perform the one-way ANOVA, you can use the `stats` module from the `scipy` library.

Here's the Python code to do this:

```
import pandas as pd
```

```
from scipy import stats


# Read in the CSV file
data = pd.read_csv("data.csv")


# Convert OrderDate and ShipDate to datetime format
data['OrderDate'] = pd.to_datetime(data['OrderDate'])
data['ShipDate'] = pd.to_datetime(data['ShipDate'])


# Calculate DeliverTime
data['DeliverTime'] = (data['ShipDate'] - data['OrderDate']).dt.days


# Perform one-way ANOVA
f_statistic, p_value = stats.f_oneway(*[data['DeliverTime'][data['ShipMode'] == mode] for mode in data['ShipMode'].unique()])


print(f"F-statistic: {f_statistic}")
print(f"P-value: {p_value}")
# Interpret the result
alpha = 0.05
if p_value < alpha:
    print("There's a statistically significant difference in DeliverTime across the different ShipModes.")
else:
    print("There's no statistically significant difference in DeliverTime across the different ShipModes.")
```

This code will let you know if the `DeliverTime` means are statistically different across the different `ShipModes`. If the p-value is small (typically < 0.05), it suggests there is a statistically significant difference.

If you were interested in a correlation coefficient, you'd typically be looking at two continuous variables. With a categorical and continuous mix, ANOVA is a more appropriate method.

Output of the provided Code:

```
F-statistic: 7075.338730314725
P-value: 0.0
There's a statistically significant difference in DeliverTime across the different ShipModes.
```

Bing Chat:

Sure, I can help you with that. Here’s a Python code snippet that calculates the correlation between ShipMode and DeliverTime:

```
import pandas as pd
from datetime import datetime

# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Convert OrderDate and ShipDate to datetime format
df['OrderDate'] = pd.to_datetime(df['OrderDate'])
df['ShipDate'] = pd.to_datetime(df['ShipDate'])

# Calculate DeliverTime as the difference in days between OrderDate and ShipDate
df['DeliverTime'] = (df['ShipDate'] - df['OrderDate']).dt.days

# Convert categorical variable 'ShipMode' into dummy/indicator variables
shipmode_dummies = pd.get_dummies(df['ShipMode'])

# Join the dummy variables to the main dataframe
df = pd.concat([df, shipmode_dummies], axis=1)

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Print the correlation between DeliverTime and each of the ShipMode dummy variables
for column in shipmode_dummies.columns:
    print(f'Correlation between DeliverTime and {column}: {correlation_matrix.loc["DeliverTime", column]}')
```

This code will calculate the correlation between DeliverTime and each category of ShipMode. The correlation coefficient ranges from -1 to 1. A value closer to 1 implies a strong positive correlation, a value closer to -1 implies a strong negative correlation, and a value around 0 implies no correlation.
Please replace `'your_file_path.csv'` with the actual path to your CSV file. Also, please ensure that the ‘OrderDate’ and ‘ShipDate’ in your CSV file are in a format that pandas can recognize as a date-time format. If not, you may need to adjust the date-time conversion code accordingly.

- Has a data Type error. Is able to fix it on Request

Fix:

The error message indicates that the `corr()` function is trying to calculate the correlation for all columns in the DataFrame, but it’s encountering a problem when it tries to convert the ‘Category’ column (which contains string values) to float.

The `corr()` function can only be used with numerical data. To fix this, you should select only the numerical columns when calculating the correlation. Here's how you can modify the code:

```
import pandas as pd
from datetime import datetime
import numpy as np


# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Convert OrderDate and ShipDate to datetime format
df['OrderDate'] = pd.to_datetime(df['OrderDate'])
df['ShipDate'] = pd.to_datetime(df['ShipDate'])

# Calculate DeliverTime as the difference in days between OrderDate and ShipDate
df['DeliverTime'] = (df['ShipDate'] - df['OrderDate']).dt.days

# Convert categorical variable 'ShipMode' into dummy/indicator variables
shipmode_dummies = pd.get_dummies(df['ShipMode'])

# Join the dummy variables to the main dataframe
df = pd.concat([df, shipmode_dummies], axis=1)

# Select only the numerical columns for correlation calculation
numerical_columns = df.select_dtypes(include=[np.number]).columns

# Calculate the correlation matrix for only numerical columns
correlation_matrix = df[numerical_columns].corr()

# Print the correlation between DeliverTime and each of the ShipMode dummy variables
for column in shipmode_dummies.columns:
    if column in correlation_matrix.columns:
        print(f'Correlation between DeliverTime and {column}: {correlation_matrix.loc["DeliverTime", column]}')
```

This code will calculate the correlation between DeliverTime and each category of ShipMode, but only for numerical data. Please replace `'your_file_path.csv'` with the actual path to your CSV file. Also, please ensure that the 'OrderDate' and 'ShipDate' in your CSV file are in a format that pandas can recognize as a date-time format. If not, you may need to adjust the date-time conversion code accordingly.

- Outputs nothing → Does not show the existing correlation

Google Bard:

Sure, here is some Python code to analyze if ShipMode and DeliverTime are correlated and how strong the correlation is:

```
import pandas as pd
from scipy.stats import pearsonr

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("orders.csv")

# Convert the ShipMode column to numerics
df['ShipMode'] = pd.factorize(df['ShipMode'])[0]

# Calculate the Pearson correlation coefficient between ShipMode and DeliverTime
correlation, p_value = pearsonr(df['ShipMode'], df['DaystoShipActual'])

# Print the correlation coefficient and p-value
print(f"Correlation coefficient: {correlation:.3f}")
print(f"p-value: {p_value:.3f}")

# Interpret the correlation coefficient
if correlation > 0.7:
    print("Strong positive correlation")
elif correlation > 0.3 and correlation <= 0.7:
    print("Moderate positive correlation")
elif correlation > 0 and correlation <= 0.3:
    print("Weak positive correlation")
elif correlation == 0:
    print("No correlation")
elif correlation < 0 and correlation >= -0.3:
    print("Weak negative correlation")
elif correlation < -0.3 and correlation >= -0.7:
    print("Moderate negative correlation")
elif correlation < -0.7:
    print("Strong negative correlation")
```

This code will print the Pearson correlation coefficient and p-value. The Pearson correlation coefficient is a measure of the linear relationship between two variables. It can range from -1 to 1, with a value of 1 indicating a perfect positive correlation, a value of -1 indicating a perfect negative correlation, and a value of 0 indicating no correlation.

The p-value is the probability of obtaining a correlation coefficient as extreme or more extreme than the observed value, assuming that there is no real correlation between the two variables. A small p-value (< 0.05) indicates that the correlation coefficient is unlikely to be due to chance, and therefore that there is a statistically significant correlation between the two variables.

To interpret the correlation coefficient, we can use the following guidelines:

- Correlation coefficient > 0.7: Strong positive correlation
- Correlation coefficient between 0.3 and 0.7: Moderate positive correlation
- Correlation coefficient between 0 and 0.3: Weak positive correlation
- Correlation coefficient = 0: No correlation
- Correlation coefficient between 0 and -0.3: Weak negative correlation
- Correlation coefficient between -0.3 and -0.7: Moderate negative correlation
- Correlation coefficient < -0.7: Strong negative correlation

In the context of your data, a positive correlation between ShipMode and DeliverTime would mean that orders shipped using faster shipping methods tend to have shorter delivery times. A negative correlation would mean that orders shipped using faster shipping methods tend to have longer delivery times.

Please note that this code is just a starting point. You may need to modify it to fit your specific needs. For example, you may want to filter the data to only include orders from a certain time period or region. You may also want to use a different correlation coefficient, such as the Spearman correlation coefficient or the Kendall correlation coefficient, depending on the type of data you have.

Output of the provided Code:

```
Correlation coefficient: -0.710
p-value: 0.000
Strong negative correlation
```

YouChat:

Certainly! To analyze the correlation between the `ShipMode` and `DeliveryTime` variables in a dataset using Python, you can follow these steps:

1. Import the required libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load your dataset into a pandas DataFrame:

```
# Assuming you have a CSV file named 'your_data.csv' with columns 'ShipMode', 'OrderDate', 'ShipDate', etc.
df = pd.read_csv('your_data.csv')
```

3. Calculate the delivery time by subtracting the `OrderDate` from the `ShipDate`:

```
df['OrderDate'] = pd.to_datetime(df['OrderDate'])
df['ShipDate'] = pd.to_datetime(df['ShipDate'])
df['DeliveryTime'] = df['ShipDate'] - df['OrderDate']
```

4. Convert the delivery time to numeric values (in days, hours, minutes, etc.) for correlation analysis:

```
df['DeliveryTime'] = df['DeliveryTime'].dt.total_seconds() / (24 * 60 * 60)  # Convert to days
```

5. Analyze the correlation between `ShipMode` and `DeliveryTime` using a scatter plot and a correlation matrix:

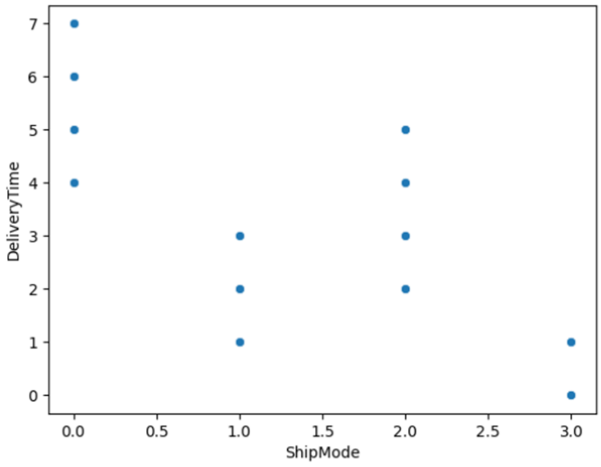
```
# Scatter plot
sns.scatterplot(data=df, x='ShipMode', y='DeliveryTime')
plt.show()

# Correlation matrix
correlation_matrix = df[['ShipMode', 'DeliveryTime']].corr()
print(correlation_matrix)
```

The scatter plot will visualize the relationship between the `ShipMode` and `DeliveryTime` variables, while the correlation matrix will give you the correlation coefficient, which indicates the strength and direction of the correlation.
Keep in mind that this code assumes you have a dataset in a CSV file format, and you may need to modify it to fit your specific data format and column names.

- Had a Datatype error, but was able to fix it right away, when asked for it

Output of the provided Code:

```
          ShipMode  DeliveryTime
ShipMode      1.000000     -0.709877
DeliveryTime -0.709877      1.000000
```



Query 2 (Okay, can you now analyze which Categories, Products or Sub_Categories make the most profit?):

ChatGPT-3.5:

To analyze which categories, products, or sub-categories make the most profit, you can use Python and Pandas to perform the analysis on your dataset. Here's a sample code to get you started:

```
import pandas as pd


# Load your dataset (replace 'your_dataset.csv' with the actual filename or data source)

df = pd.read_csv('your_dataset.csv')


# Assuming your dataset has columns 'Category', 'Sub_Category', 'ProductName', 'Profit'

# If the column names are different, replace them accordingly


# Calculate profit for each category

category_profit = df.groupby('Category')['Profit'].sum().reset_index()

category_profit = category_profit.sort_values(by='Profit', ascending=False)

print("Category-wise Profit:")

print(category_profit)


# Calculate profit for each sub-category

sub_category_profit = df.groupby(['Category', 'Sub_Category'])['Profit'].sum().reset_index()

sub_category_profit = sub_category_profit.sort_values(by='Profit', ascending=False)

print("\nSub-Category-wise Profit:")

print(sub_category_profit)


# Calculate profit for each product

product_profit = df.groupby(['Category', 'Sub_Category', 'ProductName'])['Profit'].sum().reset_index()

product_profit = product_profit.sort_values(by='Profit', ascending=False)

print("\nProduct-wise Profit:")

print(product_profit)
```

Make sure to replace 'your_dataset.csv' with the actual filename or data source, and adjust the column names ('Category', 'Sub_Category', 'Product', 'Profit') to match your dataset's column names.

This code will group your data by category, sub-category, and product, and then calculate and display the total profit for each of them, sorted in descending order. You'll be able to see which categories, sub-categories, and products are generating the most profit in your dataset.

Output of the provided Code:

```
Category-wise Profit:
        Category  Profit
2      Technology  145429
1  Office Supplies  122474
0       Furniture   18444

Sub-Category-wise Profit:
         Category Sub_Category  Profit
14      Technology     Copiers   55618
16      Technology      Phones   44492
13      Technology  Accessories   41932
10  Office Supplies       Paper   34053
6   Office Supplies     Binders   30200
1        Furniture      Chairs   26586
11  Office Supplies     Storage   21280
4   Office Supplies   Appliances   18132
2        Furniture  Furnishings   13070
7   Office Supplies    Envelopes    6956
5   Office Supplies         Art    6530
9   Office Supplies      Labels    5558
15      Technology    Machines    3387
8   Office Supplies    Fasteners     952
12  Office Supplies    Supplies   -1187
0        Furniture    Bookcases   -3479
3        Furniture      Tables  -17733

Product-wise Profit:
           Category  ... Profit
1591      Technology  ...  25200
715   Office Supplies  ...   7751
1594      Technology  ...   6984
1588      Technology  ...   4571
1619      Technology  ...   4095
```

```
...            ... ...   ...

346       Furniture  ...  -1933

352       Furniture  ...  -2876

1613      Technology  ...  -3840

1631      Technology  ...  -4590

1612      Technology  ...  -8880


[1849 rows x 4 columns]
```

ChatGPT-4.0:

Certainly! To analyze which Categories, Products, or Sub-Categories generate the most profit, you can utilize the `groupby` method available in `pandas`. This method allows you to group the data by certain columns and then aggregate the values in another column.

Here's how you can do it:

```
import pandas as pd


# Read in the CSV file

data = pd.read_csv("data.csv")


# Group by Category and sum up Profit

category_profit = data.groupby('Category')['Profit'].sum().sort_values(ascending=False)


# Group by Product and sum up Profit

product_profit = data.groupby('Product')['Profit'].sum().sort_values(ascending=False)


# Group by Sub-Category and sum up Profit

sub_category_profit = data.groupby('Sub_Category')['Profit'].sum().sort_values(ascending=False)


# Display the results

print("Profit by Category:")

print(category_profit)


print("\nProfit by Product:")
print(product_profit.head(10))  # Displaying top 10 products by profit


print("\nProfit by Sub-Category:")
print(sub_category_profit)
```

Here's a brief explanation of what the code does:

1. Read in the CSV file.
2. Group the data by `Category` and then sum up the `Profit`. Sort the results in descending order to get categories that generate the most profit at the top.
3. Do the same for `Product` and `Sub_Category`.
4. Display the results.

This code will provide you with insights on which categories, products, or sub-categories are most profitable in your dataset. Adjust the code as necessary if your column names are different.

Output of the provided Code:

```
Profit by Category:
Category
Technology       145429
Office Supplies    122474
Furniture        18444
Name: Profit, dtype: int64


Profit by Product:
ProductName
Canon imageCLASS 2200 Advanced Copier                         25200
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind    7751
Hewlett Packard LaserJet 3310 Copier                          6984
Canon PC1060 Personal Laser Copier                            4571
HP Designjet T520 Inkjet Large Format Printer - 24 Color                4095
Ativa V4110MDD Micro-Cut Shredder                             3773
3D Systems Cube Printer, 2nd Generation, Magenta                      3718
Plantronics Savi W720 Multi-Device Wireless Headset System                3697
Ibico EPK-21 Electric Binding System                          3345
Zebra ZM400 Thermal Label Printer                             3344
Name: Profit, dtype: int64
```

```
Profit by Sub-Category:
Sub_Category
Copiers       55618
Phones        44492
Accessories   41932
Paper         34053
Binders       30200
Chairs        26586
Storage       21280
Appliances    18132
Furnishings   13070
Envelopes      6956
Art            6530
Labels         5558
Machines       3387
Fasteners       952
Supplies      -1187
Bookcases     -3479
Tables       -17733
Name: Profit, dtype: int64
```

Bing Chat:

Sure, I can help you with that. Here’s a Python code snippet that calculates the total profit for each Category, Product, and Sub-Category:

```
import pandas as pd

# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Calculate total profit for each Category
category_profit = df.groupby('Category')['Profit'].sum().sort_values(ascending=False)
print("Total profit for each Category:")
print(category_profit)

# Calculate total profit for each Product
product_profit = df.groupby('ProductName')['Profit'].sum().sort_values(ascending=False)
print("\nTotal profit for each Product:")
print(product_profit)

# Calculate total profit for each Sub-Category
subcategory_profit = df.groupby('Sub_Category')['Profit'].sum().sort_values(ascending=False)
```

```
print("\nTotal profit for each Sub-Category:")
print(subcategory_profit)
```

This code will calculate the total profit for each Category, Product, and Sub-Category in your data, and print them out in descending order. Please replace `'your_file_path.csv'` with the actual path to your CSV file.

Remember to install pandas library if you haven't done so. You can install it using pip:

```
pip install pandas
```

Output of the provided Code:

```
Total profit for each Category:
Category
Technology        145429
Office Supplies    122474
Furniture          18444
Name: Profit, dtype: int64


Total profit for each Product:
ProductName
Canon imageCLASS 2200 Advanced Copier                                   25200
Fellowes PB500 Electric Punch Plastic Comb Binding Machine with Manual Bind     7751
Hewlett Packard LaserJet 3310 Copier                                   6984
Canon PC1060 Personal Laser Copier                                     4571
HP Designjet T520 Inkjet Large Format Printer - 24 Color                 4095
                                        ...
Bush Advantage Collection Racetrack Conference Table                   -1933
Chromcraft Bull-Nose Wood Oval Conference Tables & Bases                  -2876
Cubify CubeX 3D Printer Triple Head Print                             -3840
Lexmark MX611dhe Monochrome Laser Printer                              -4590
Cubify CubeX 3D Printer Double Head Print                             -8880
Name: Profit, Length: 1840, dtype: int64


Total profit for each Sub-Category:
Sub_Category
Copiers       55618
Phones        44492
Accessories    41932
Paper         34053
Binders       30200
Chairs        26586
Storage       21280
```

```
Appliances    18132

Furnishings   13070

Envelopes      6956

Art          6530

Labels        5558

Machines       3387

Fasteners       952

Supplies      -1187

Bookcases     -3479

Tables      -17733

Name: Profit, dtype: int64
```

Google Bard:

Sure, here is some Python code to analyze which categories, products, or sub-categories make the most profit:

```
import pandas as pd

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("orders.csv")

# Calculate the total profit for each category, product, and sub-category
total_profit_by_category = df.groupby('Category')['Profit'].sum()
total_profit_by_product = df.groupby('ProductName')['Profit'].sum()
total_profit_by_sub_category = df.groupby('Sub_Category')['Profit'].sum()

# Print the total profit for each category, product, and sub-category
print("Total profit by category:")
print(total_profit_by_category)

print("Total profit by product:")
print(total_profit_by_product)

print("Total profit by sub-category:")
print(total_profit_by_sub_category)
```

This code will print the total profit for each category, product, and sub-category. You can then use this information to identify the categories, products, or sub-categories that are making the most profit.

For example, if you are interested in identifying the most profitable products, you could sort the `total_profit_by_product` DataFrame by the `Profit` column in descending order. The products at the top of the DataFrame will be the most profitable products.

You can also use this code to identify the least profitable categories, products, or sub-categories. Simply sort the DataFrame in ascending order by the `Profit` column.

The categories, products, or sub-categories at the bottom of the DataFrame will be the least profitable.

You can also use this code to identify trends in profitability over time. For example, you could create a new column in the DataFrame that calculates the profit margin for each category, product, or sub-category. You could then use this column to identify categories, products, or sub-categories that are becoming more or less profitable over time.

I hope this helps!

Output of the provided Code:

```
Total profit by category:
Category
Furniture         18444
Office Supplies    122474
Technology        145429
Name: Profit, dtype: int64
Total profit by product:
ProductName
#10 Gummed Flap White Envelopes, 100/Box                                          17
#10 Self-Seal White Envelopes                                            52
#10 White Business Envelopes,4 1/8 x 9 1/2                                     224
#10- 4 1/8 x 9 1/2 Recycled Envelopes                                        115
#10- 4 1/8 x 9 1/2 Security-Tint Envelopes                                     64
                                                          ...
iKross Bluetooth Portable Keyboard + Cell Phone Stand Holder + Brush for Apple iPhone 5S 5C 5, 4S 4    116
iOttie HLCRIO102 Car Mount                                              -12
iOttie XL Car Mount                                               -50
invisibleSHIELD by ZAGG Smudge-Free Screen Protector                              170
netTALK DUO VoIP Telephone Service                                        429
Name: Profit, Length: 1840, dtype: int64
Total profit by sub-category:
Sub_Category
Accessories    41932
Appliances     18132
Art           6530
Binders       30200
Bookcases     -3479
Chairs        26586
Copiers       55618
```

```
Envelopes      6956

Fasteners       952

Furnishings   13070

Labels        5558

Machines       3387

Paper        34053

Phones        44492

Storage      21280

Supplies      -1187

Tables      -17733

Name: Profit, dtype: int64
```

YouChat:

Certainly! To analyze which categories, products, or sub-categories make the most profit in a dataset using Python, you can follow these steps:

1. Import the required libraries:

```
import pandas as pd

import matplotlib.pyplot as plt
```

2. Load your dataset into a pandas DataFrame:

```
# Assuming you have a CSV file named 'your_data.csv' with columns like 'Category', 'Product', 'Sub_Category', 'Profit', etc.

df = pd.read_csv('your_data.csv')
```

3. Group the data by category, product, or sub-category and calculate the total profit for each group:

```
category_profit = df.groupby('Category')['Profit'].sum()

product_profit = df.groupby('ProductName')['Profit'].sum()

sub_category_profit = df.groupby('Sub_Category')['Profit'].sum()
```

4. Sort the profit values in descending order:

```
category_profit = category_profit.sort_values(ascending=False)

product_profit = product_profit.sort_values(ascending=False)

sub_category_profit = sub_category_profit.sort_values(ascending=False)
```

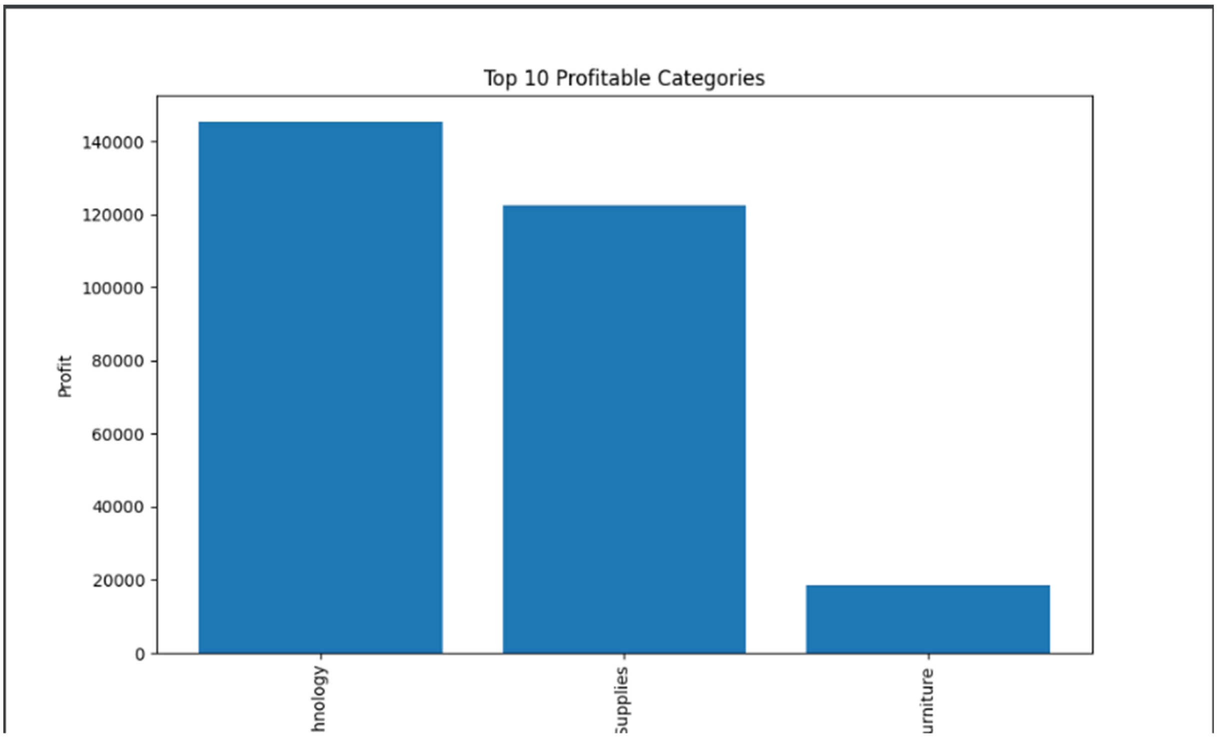
5. Visualize the top 10 profit values using a bar plot:

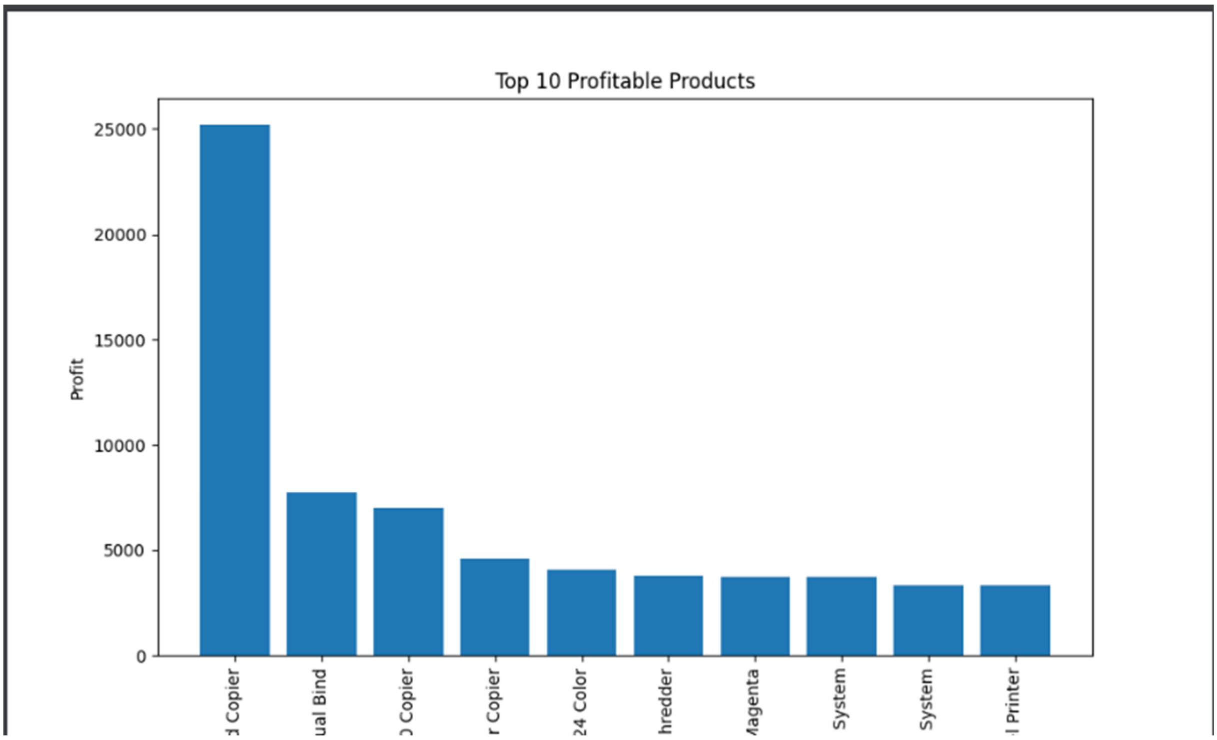
```
top_10_category_profit = category_profit.head(10)
top_10_product_profit = product_profit.head(10)
top_10_sub_category_profit = sub_category_profit.head(10)

# Bar plot for category profit
plt.figure(figsize=(10, 6))
plt.bar(top_10_category_profit.index, top_10_category_profit.values)
plt.xlabel('Category')
plt.ylabel('Profit')
plt.title('Top 10 Profitable Categories')
plt.xticks(rotation=90)
plt.show()

# Bar plot for product profit
plt.figure(figsize=(10, 6))
plt.bar(top_10_product_profit.index, top_10_product_profit.values)
plt.xlabel('ProductName')
plt.ylabel('Profit')
plt.title('Top 10 Profitable Products')
plt.xticks(rotation=90)
plt.show()

# Bar plot for sub-category profit
plt.figure(figsize=(10, 6))
plt.bar(top_10_sub_category_profit.index, top_10_sub_category_profit.values)
plt.xlabel('Sub-Category')
plt.ylabel('Profit')
plt.title('Top 10 Profitable Sub-Categories')
plt.xticks(rotation=90)
plt.show()
```

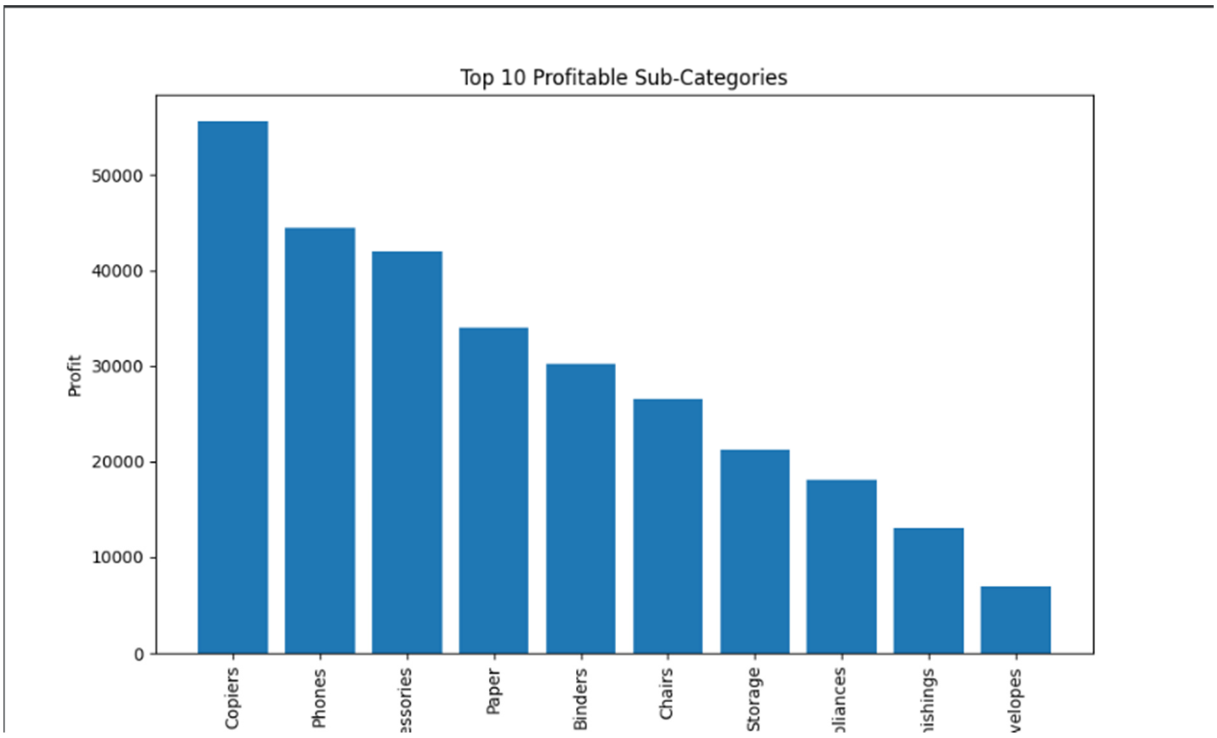
These steps will group the data by category, product, or sub-category and calculate the total profit for each group. Then, by sorting the profit values in descending order, you can identify the categories, products, or sub-categories that make the most profit. Finally, the bar plots will visualize the top 10 profit values for each category, product, or sub-category.

Please ensure that you replace `'your_data.csv'` with the actual path and filename of your dataset. Let me know if you have any further questions!

Output of the provided Code:







Query 3 (Okay based on this profit analysis can you now analyze if giving a discount ("Discount" column) has a positive or negative impact on the overall profit?):

ChatGPT-3.5:

Certainly! You can analyze if giving a discount has a positive or negative impact on overall profit by comparing the profit for orders with and without discounts. Here's the Python code to do that:

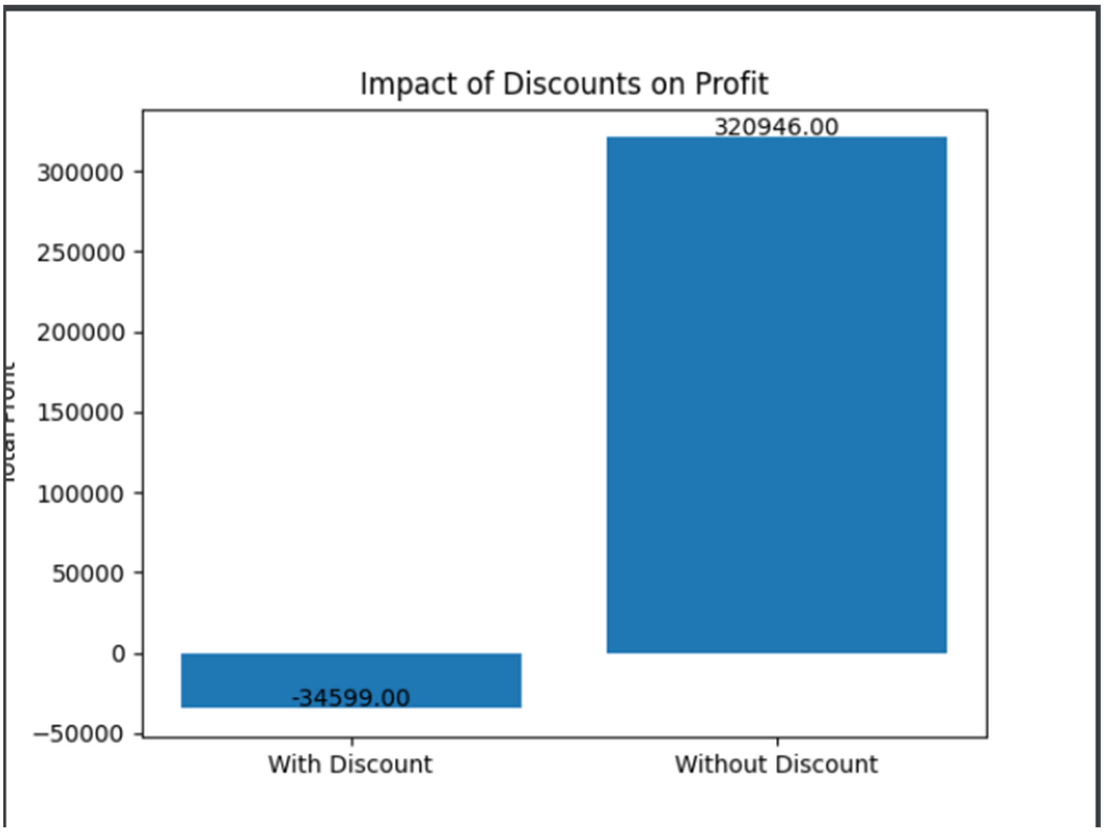
```
import pandas as pd

import matplotlib.pyplot as plt


# Load your dataset (replace 'your_dataset.csv' with the actual filename or data source)

df = pd.read_csv('your_dataset.csv')


# Assuming your dataset has columns 'Profit' and 'Discount'

# If the column names are different, replace them accordingly


# Calculate profit for orders with discounts

profit_with_discount = df[df['Discount'] > 0]['Profit'].sum()


# Calculate profit for orders without discounts
```

```
profit_without_discount = df[df['Discount'] == 0]['Profit'].sum()


# Calculate the total profit for all orders
total_profit = df['Profit'].sum()


# Calculate the percentage change in profit when a discount is given
percentage_change = ((profit_with_discount - profit_without_discount) / total_profit) * 100


# Plot the results
labels = ['With Discount', 'Without Discount']
profits = [profit_with_discount, profit_without_discount]


plt.bar(labels, profits)
plt.ylabel('Total Profit')
plt.title('Impact of Discounts on Profit')
plt.text(0, profits[0] + 1000, f'{profit_with_discount:.2f}', ha='center')
plt.text(1, profits[1] + 1000, f'{profit_without_discount:.2f}', ha='center')
plt.show()


# Display the percentage change in profit
if percentage_change > 0:
    print(f"Giving discounts has a positive impact on overall profit, increasing it by {percentage_change:.2f}%.")
elif percentage_change < 0:
    print(f"Giving discounts has a negative impact on overall profit, decreasing it by {percentage_change:.2f}%.")
else:
    print("Giving discounts has no significant impact on overall profit.")
```

Replace 'your_dataset.csv' with the actual filename or data source, and make sure the column names ('Profit' and 'Discount') match your dataset's column names. This code will calculate and display the impact of giving discounts on overall profit and visualize it using a bar chart.

Output of the provided Code:

```
Giving discounts has a negative impact on overall profit, decreasing it by -124.17%.
```



ChatGPT-4.0:

Of course. One way to analyze the impact of discounts on profit is to compare the average profit for transactions with discounts to those without. Let's write code to achieve this:

```
# Split data based on whether a discount was given or not
with_discount = data[data['Discount'] > 0]
without_discount = data[data['Discount'] == 0]

# Calculate average profit for both groups
avg_profit_with_discount = with_discount['Profit'].mean()
avg_profit_without_discount = without_discount['Profit'].mean()

# Display results
print(f"Average profit with discount: ${avg_profit_with_discount:.2f}")
print(f"Average profit without discount: ${avg_profit_without_discount:.2f}")

# Determine the impact
if avg_profit_with_discount > avg_profit_without_discount:
    print("Offering discounts tends to result in higher average profit.")
```

```
else:
    print("Offering discounts tends to result in lower average profit.")
```

This code compares the average profit for orders with discounts to the average profit for orders without discounts. The conclusion about the impact of discounts on profit is then based on these averages. However, please note that this is a simple comparison and other factors might be at play. More in-depth analysis using statistical methods would give a better understanding of the relationship between discounts and profits.

Output of the provided Code:

```
Average profit with discount: $-6.66

Average profit without discount: $66.89

Offering discounts tends to result in lower average profit.
```

Bing Chat:

Sure, I can help you with that. Here’s a Python code snippet that calculates the average profit for orders with and without a discount:

```
import pandas as pd

# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Create a new column 'HasDiscount' that is True if Discount > 0 and False otherwise
df['HasDiscount'] = df['Discount'] > 0

# Calculate average profit for orders with and without a discount
average_profit = df.groupby('HasDiscount')['Profit'].mean()
print("Average profit for orders:")
print(average_profit)
```

This code will calculate the average profit for orders with a discount and orders without a discount. If the average profit is higher for orders with a discount, it suggests that giving a discount has a positive impact on the overall profit. Conversely, if the average profit is higher for orders without a discount, it suggests that giving a discount has a negative impact on the overall profit.
Please replace `'your_file_path.csv'` with the actual path to your CSV file.

Remember to install pandas library if you haven’t done so. You can install it using pip:

```
pip install pandas
```

Output of the provided Code:

```
Average profit for orders:
HasDiscount
```

```
False    66.891622
True     -6.658776
Name: Profit, dtype: float64
```

Google Bard:

Sure, here is some Python code to analyze the impact of discounts on overall profit:

```
import pandas as pd

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("orders.csv")

# Calculate the profit with discount
profit_with_discount = df['Profit'] * (1 - df['Discount'])

# Calculate the difference between profit with and without discount
profit_difference = df['Profit'] - profit_with_discount

# Calculate the average profit difference
average_profit_difference = profit_difference.mean()

# Print the average profit difference
print(f"Average profit difference: {average_profit_difference:.2f}")

# Interpret the average profit difference
if average_profit_difference > 0:
   print("Giving discounts has a negative impact on overall profit.")
elif average_profit_difference < 0:
   print("Giving discounts has a positive impact on overall profit.")
else:
   print("Giving discounts has no impact on overall profit.")
```

This code will print the average profit difference between orders with and without discounts. If the average profit difference is positive, then giving discounts has a negative impact on overall profit. If the average profit difference is negative, then giving discounts has a positive impact on overall profit. If the average profit difference is zero, then giving discounts has no impact on overall profit.

It is important to note that this code is just a starting point. You may need to modify it to fit your specific needs. For example, you may want to filter the data to only include orders from a certain time period or region. You may also want to consider other factors that could affect profit, such as the cost of goods sold and marketing expenses.

Output of the provided Code:

```
Average profit difference: -6.14
Giving discounts has a positive impact on overall profit.
```

- Analysis does not show the intended information.

YouChat:

Certainly! To analyze the impact of giving a discount on the overall profit, you can follow these steps:

1. Import the required libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
```

2. Load your dataset into a pandas DataFrame:

```
# Assuming you have a CSV file named 'your_data.csv' with columns like 'Discount' and 'Profit'
df = pd.read_csv('your_data.csv')
```

3. Group the data by the discount amount and calculate the mean profit for each discount group:

```
discount_profit = df.groupby('Discount')['Profit'].mean()
```

4. Plot the average profit against the discount amount:

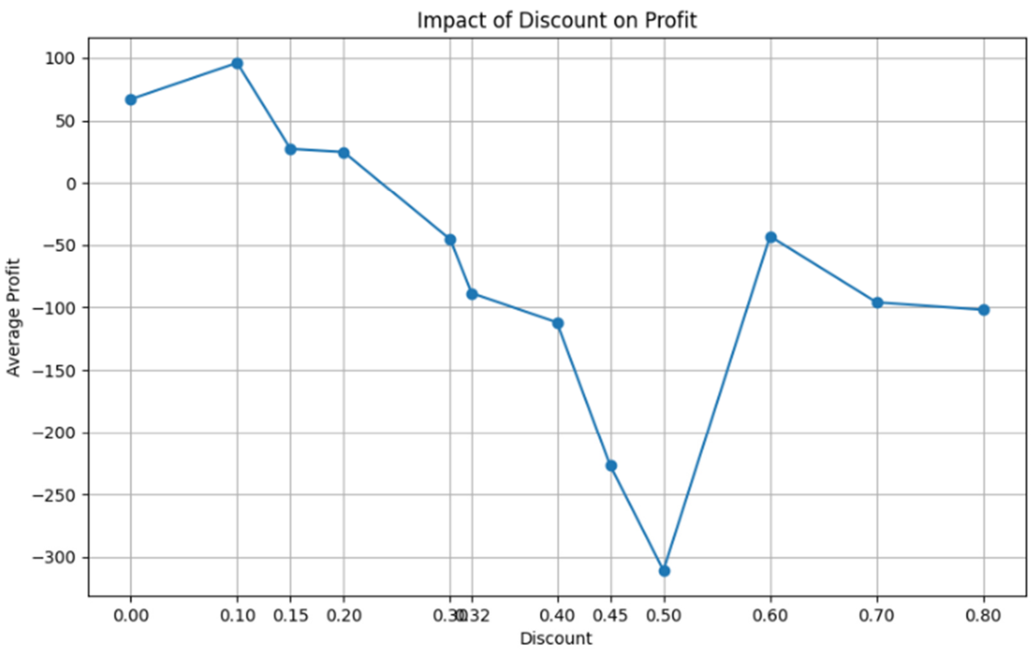
```
plt.figure(figsize=(10, 6))
plt.plot(discount_profit.index, discount_profit.values, marker='o')
plt.xlabel('Discount')
plt.ylabel('Average Profit')
plt.title('Impact of Discount on Profit')
plt.xticks(discount_profit.index)
plt.grid(True)
plt.show()
```

This code will group the data by the discount amount and calculate the average profit for each discount group. The resulting plot will show the impact of discount on the average profit.

If the plot shows an increasing trend as the discount amount increases, it suggests that giving a discount has a positive impact on the overall profit. On the other hand, if the plot shows a decreasing trend, it indicates that giving a discount has a negative impact on the overall profit.

Please ensure that you replace `'your_data.csv'` with the actual path and filename of your dataset. Let me know if you have any further questions!

Output of the provided Code:



Query 4 (Can you also include the value of the discount rate in the analysis, so you can see if there is a difference between high and low discounts?):

ChatGPT-3.5:

Certainly! To include the value of the discount rate in the analysis and compare the difference between high and low discounts based on both the discount rate and profit, you can create two separate groups: high discount rates and low discount rates. Here's the Python code to do that:

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

# Load your dataset (assuming it's named 'data')
# Replace 'data.csv' with your actual dataset file
data = pd.read_csv('data.csv')

# Define the step size for discount groups
discount_step = 0.1

# Create a range of discount values from 0 to 1 with the specified step size
```

```
discount_values = np.arange(0, 1.1, discount_step)

# Create an empty DataFrame to store the results
discount_profit_summary = pd.DataFrame(columns=['Discount Range', 'Average Profit'])

# Iterate through each discount range
for i in range(len(discount_values) - 1):
    lower_bound = discount_values[i]
    upper_bound = discount_values[i + 1]

    # Filter the data for the current discount range
    filtered_data = data[(data['Discount'] >= lower_bound) & (data['Discount'] < upper_bound)]

    # Calculate the average profit for the current discount range
    average_profit = filtered_data['Profit'].mean()

    # Append the results to the summary DataFrame
    discount_range_label = f"{lower_bound}-{upper_bound}"
    discount_profit_summary = pd.concat([discount_profit_summary, pd.DataFrame({'Discount Range': discount_range_label,
                                                 'Average Profit': average_profit},
                                                index=[0])],
                                        ignore_index=True)

# Plot the results
plt.figure(figsize=(12, 6))
plt.bar(discount_profit_summary['Discount Range'], discount_profit_summary['Average Profit'], color='blue')
plt.xlabel('Discount Range')
plt.ylabel('Average Profit')
plt.title('Average Profit by Discount Range')
plt.xticks(rotation=90)
plt.show()

# Display the summary table
```

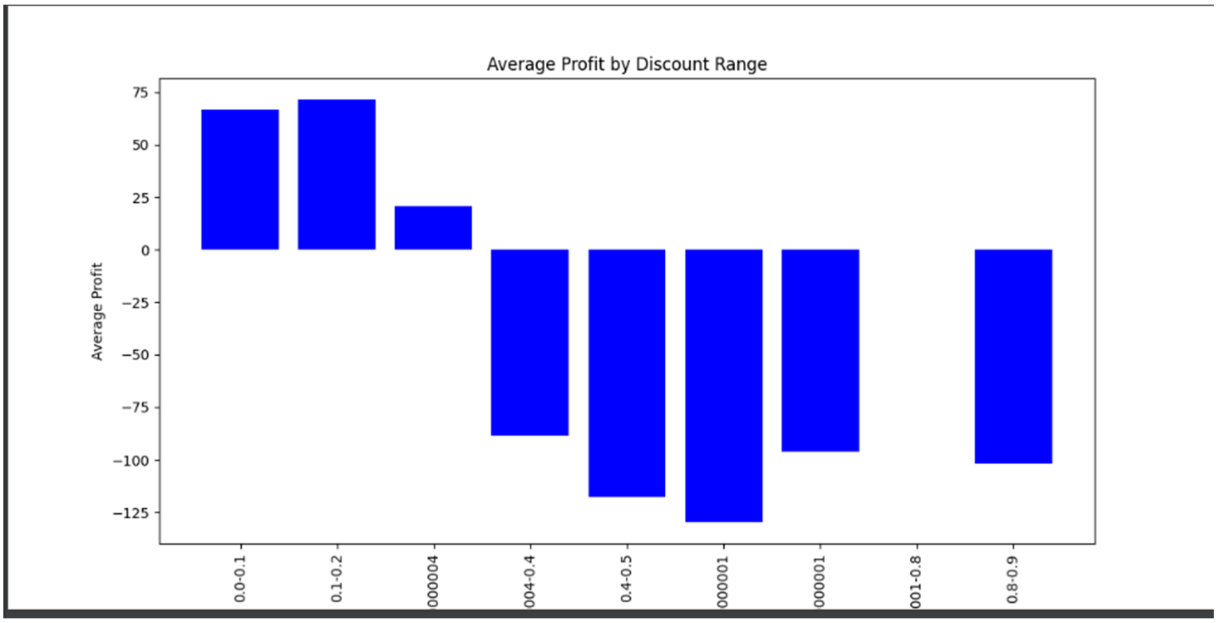
```
print(discount_profit_summary)
```

In this code, we categorize discounts into "High" or "Low" based on the defined threshold (`discount_threshold`). Then, we calculate the average profit and discount rate for each category and plot the results to visualize the difference in profit between high and low discount levels. The summary table displays the average profit and discount rate for each category.

You can adjust the `discount_threshold` to define what you consider as a high discount rate. This will help you analyze the impact of different discount levels on profit.

Output of the provided Code:

```
Discount Range  Average Profit
0                    0.0-0.1      66.891622
1                    0.1-0.2      71.513699
2           0.2-0.30000000000000004      20.592688
3           0.30000000000000004-0.4     -88.518519
4                    0.4-0.5    -117.718894
5            0.5-0.6000000000000001    -129.666667
6  0.6000000000000001-0.7000000000000001     -95.906699
7            0.7000000000000001-0.8          NaN
8                    0.8-0.9    -101.820000
9                    0.9-1.0          NaN
```



ChatGPT-4.0:

Certainly! A deeper analysis would require segregating the data based on discount rates to understand how different levels of discounts might impact the profit.

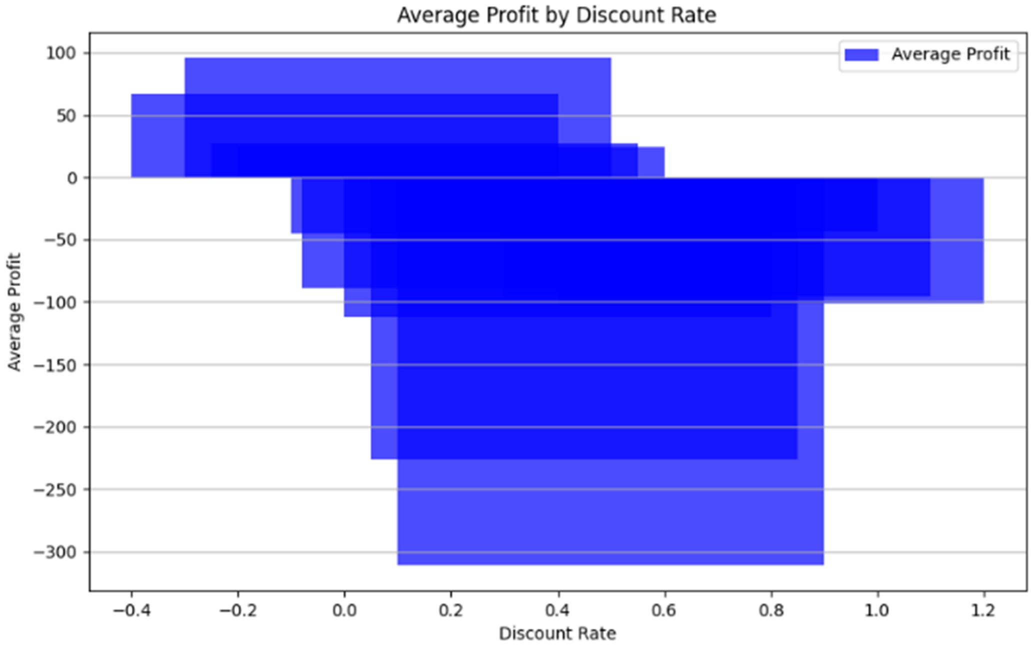
Here's a Python code that groups the data by discount rates and calculates the average profit for each group:

```
import pandas as pd


# Read in the CSV file

data = pd.read_csv("data.csv")


# Group data by discount rate and calculate the average profit for each group

grouped_data = data.groupby('Discount').agg(AverageProfit=('Profit', 'mean'), NumberOfTransactions=('Profit', 'size')).reset_index()


# Sort data by discount for better visualization

grouped_data = grouped_data.sort_values(by='Discount')


# Display results

print(grouped_data)


# If you'd like to visualize the results, you can use the following code (ensure you have matplotlib installed):

import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))

plt.bar(grouped_data['Discount'], grouped_data['AverageProfit'], color='blue', alpha=0.7, label='Average Profit')

plt.xlabel('Discount Rate')

plt.ylabel('Average Profit')

plt.title('Average Profit by Discount Rate')

plt.legend()

plt.grid(axis='y')

plt.show()
```

This code gives an aggregated view of the average profit for each distinct discount rate present in the data. By visualizing the results with a bar chart, you can easily identify trends or patterns regarding how different discount rates might impact profit.

Output of the provided Code:

```
Discount  AverageProfit  NumberOfTransactions
0      0.00     66.891622            4798
1      0.10     96.010638              94
2      0.15     27.230769              52
3      0.20     24.704403            3657
4      0.30    -45.647577             227
5      0.32    -88.518519              27
6      0.40   -111.902913              206
7      0.45   -226.636364               11
8      0.50   -310.727273               66
9      0.60    -43.072464             138
10      0.70    -95.906699              418
11      0.80   -101.820000               300
```



Bing Chat:

Absolutely, we can modify the analysis to consider the value of the discount rate. Here’s a Python code snippet that calculates the average profit for different ranges of discounts:

```
import pandas as pd

# Replace 'your_file_path.csv' with the path to your CSV file
df = pd.read_csv('your_file_path.csv')

# Define discount ranges
bins = [0, 0.2, 0.4, 0.6, 0.8, 1]

# Create a new column 'DiscountRange' that categorizes the discount into the defined ranges
df['DiscountRange'] = pd.cut(df['Discount'], bins)

# Calculate average profit for each discount range
average_profit = df.groupby('DiscountRange')['Profit'].mean()
print("Average profit for each discount range:")
print(average_profit)
```

This code will calculate the average profit for different ranges of discounts: 0-20%, 20-40%, 40-60%, 60-80%, and 80-100%. This will give you a more detailed view of how the value of the discount rate affects the overall profit.
Please replace `'your_file_path.csv'` with the actual path to your CSV file.

Remember to install pandas library if you haven’t done so. You can install it using pip:
`pip install pandas`

Output of the provided Code:

```
Average profit for each discount range:
DiscountRange
(0.0, 0.2]     26.501446
(0.2, 0.4]    -77.834783
(0.4, 0.6]   -134.627907
(0.6, 0.8]    -98.377437
(0.8, 1.0]           NaN
Name: Profit, dtype: float64
```

- Asked to also include 0 discount into the overview. Fixed code worked right away.

```
bins = [-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1]
```

```
Average profit for each discount range:
DiscountRange
(-0.1, 0.0]    66.891622
(0.0, 0.2]     26.501446
(0.2, 0.4]    -77.834783
(0.4, 0.6]    -134.627907
(0.6, 0.8]     -98.377437
(0.8, 1.0]            NaN
Name: Profit, dtype: float64
```

Google Bard:

Sure, here is an updated version of my code that includes the value of the discount rate and compares the profit difference between high and low discounts:

```
import pandas as pd

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("orders.csv")

# Calculate the profit without discount
profit_without_discount = df['Profit'] / (1 - df['Discount'])

# Calculate the difference between profit with and without discount
profit_difference = profit_without_discount - df['Profit']

# Calculate the average profit difference for high and low discounts
high_discount_profit_difference = profit_difference[df['Discount'] >= 0.2].mean()
low_discount_profit_difference = profit_difference[df['Discount'] < 0.2].mean()

# Print the average profit difference for high and low discounts
print(f"Average profit difference for high discounts: {high_discount_profit_difference:.2f}")
print(f"Average profit difference for low discounts: {low_discount_profit_difference:.2f}")
```

This code will print the average profit difference between orders with and without discounts, as well as the average profit difference for high and low discounts. If the average profit difference for high discounts is greater than the average profit difference for low discounts, then high discounts have a greater negative impact on overall profit.

It is important to note that this code is just a starting point. You may need to modify it to fit your specific needs. For example, you may want to use a different threshold for high and low discounts. You may also want to consider other factors that could affect profit, such as the cost of goods sold and marketing expenses.

Output of the provided Code:

```
Average profit difference for high discounts: -48.62

Average profit difference for low discounts: 0.25
```

- Due to the very similar code to Query 3. The output still does not show the intended information.

You Chat:

- Did not need to ask the 4. query because the answer on the 3. query already gave all the relevant information.