Supermarket-Grocery-Sales-Retail-Data-Analysis-Project/EDA

This dataset can be used for predictive data analytics purposes.

Introduction

The Supermart Grocery Sales - Retail Analytics Dataset is a fictional dataset designed to provide data analysts with an opportunity to practice exploratory data analysis and data visualization. It contains data on orders placed by customers using a grocery delivery application in the state of Tamil Nadu, India. The dataset is a useful resource for understanding consumer behavior in the grocery retail industry, and for developing insights into the factors that drive sales in this sector. By analyzing the dataset, data analysts can identify patterns, trends, and correlations that can help retailers optimize their marketing, pricing, and product strategies to increase sales and revenue.

Step 1: Import Required Libraries

In [12]:
```
import numpy as np # linear Algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns # Statistical plot visualization
import matplotlib.pyplot as plt # visualize the data
import os # redirect the file path from anywhere
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Step 2: Load the Dataset

In [13]:
```
df = pd.read_csv("C:\\Users\\aman\\Downloads\\Supermart Grocery Sales - Retail Anal
df.head()
```

Out[13]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	P
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.12	4
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.18	1
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.21	1
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.25	
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.26	9

Step 3: Understand the dataset

- Check the number of rows and columns in the dataset
- Check the data types of each column.
- Check for any missing or null values

In [14]:
```
# Check the number of rows and columns in the dataset
print('Number of rows:', df.shape[0])
print('Number of columns:', df.shape[1])
```

```
Number of rows: 9994
Number of columns: 11
```

In [15]:
```
# Check the data types of each column
df.dtypes
```

Out[15]:
```
Order ID          object
Customer Name     object
Category          object
Sub Category      object
City              object
Order Date        object
Region            object
Sales              int64
Discount         float64
Profit           float64
State             object
dtype: object
```

In [16]:
```
# Check for any missing or null values
```

```
print(df.isnull().sum())
```

```
Order ID         0
Customer Name    0
Category         0
Sub Category     0
City             0
Order Date       0
Region           0
Sales            0
Discount         0
Profit           0
State            0
dtype: int64
```

Step 4: Data cleaning

We can remove any unnecessary columns, rename the columns if needed, and convert the date column into a datetime format if it's not already.

In [17]:
```
# Remove unnecessary columns
df = df.drop(columns=['Order ID', 'State'])

# Rename columns
df = df.rename(columns={'Sub Category': 'Sub_Category', 'Order Date': 'Order_Date'}
```

In [18]:
```
# Convert date column to datetime format
df['Order_Date'] = pd.to_datetime(df['Order_Date'], errors='coerce')

# Check for any remaining null values in the date column
df[df['Order_Date'].isnull()]

# If any null values are found, check the original data to identify the correct dat
#format parameter in the to_datetime function accordingly.
```

Out[18]:

	Customer Name	Category	Sub_Category	City	Order_Date	Region	Sales	Discou
12	Sharon	Snacks	Cookies	Dindigul	NaT	South	1659	0.
14	Sundar	Eggs, Meat & Fish	Chicken	Kanyakumari	NaT	Central	831	0.
15	Ramesh	Oil & Masala	Edible Oil & Ghee	Krishnagiri	NaT	Central	1440	0.
17	Arutra	Beverages	Health Drinks	Bodi	NaT	West	1617	0.
18	Haseena	Eggs, Meat & Fish	Mutton	Tenkasi	NaT	West	1757	0.
...	...	...	...	...	...	...	...	
9985	Shree	Eggs, Meat & Fish	Mutton	Kanyakumari	NaT	West	1286	0.
9987	Ganesh	Fruits & Veggies	Fresh Vegetables	Theni	NaT	West	1350	0.
9989	Sudeep	Eggs, Meat & Fish	Eggs	Madurai	NaT	West	945	0.
9992	Peer	Oil & Masala	Spices	Pudukottai	NaT	West	1659	0.
9993	Ganesh	Food Grains	Atta & Flour	Tirunelveli	NaT	West	1034	0.

5952 rows × 9 columns

In [19]:
```python
# Convert date column to datetime format
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='%m/%d/%Y')
```

In [20]:
```python
# Convert date column to datetime format
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='%m/%d/%Y')

# Convert date column to datetime format
df['Order_Date'] = pd.to_datetime(df['Order_Date'], format='%m-%d-%Y')
```

Step 5: Exploratory data analysis

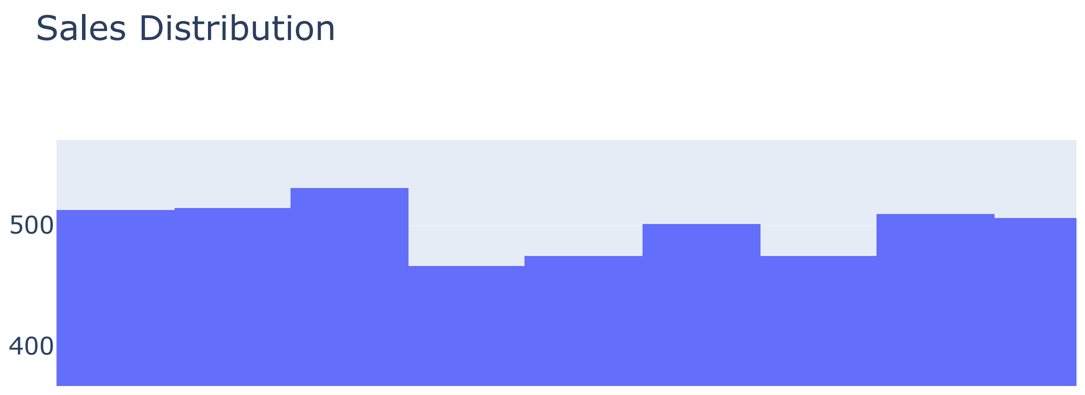
We can calculate basic statistics such as mean, median, and mode for the numerical columns, and create visualizations such as histograms, scatterplots, and boxplots to understand the distribution of the data.

In [21]:
```
# Calculate basic statistics
df.describe()
```

Out[21]:

	Order_Date	Sales	Discount	Profit
count	4042	9994.000000	9994.000000	9994.000000
mean	2017-04-28 03:16:17.931716864	1496.596158	0.226817	374.937082
min	2015-01-03 00:00:00	500.000000	0.100000	25.250000
25%	2016-05-09 06:00:00	1000.000000	0.160000	180.022500
50%	2017-07-01 00:00:00	1498.000000	0.230000	320.780000
75%	2018-06-01 00:00:00	1994.750000	0.290000	525.627500
max	2018-12-11 00:00:00	2500.000000	0.350000	1120.950000
std	NaN	577.559036	0.074636	239.932881

In [22]:
```
# Create histogram of sales
fig = px.histogram(df, x='Sales', nbins=30, title='Sales Distribution')
fig.show()
```



In [23]:
```
# Create scatterplot of sales and profit
fig = px.scatter(df, x='Sales', y='Profit', title='Sales vs. Profit')
fig.show()
```

Sales vs. Profit
1000
800

In [24]:
```
# Create boxplot of sales by region
fig = px.box(df, x='Region', y='Sales', title='Sales by Region')
fig.show()
```

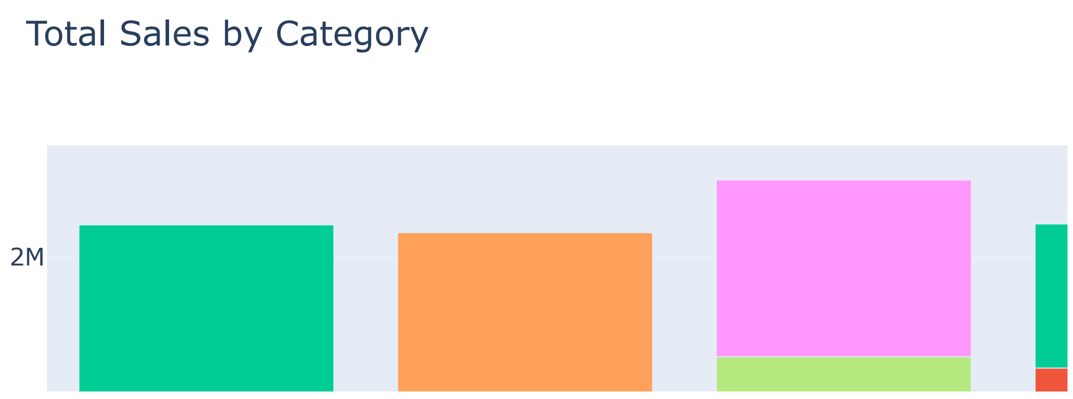
Sales by Region
2500
2000

Step 6: Analyze sales by category and sub-category¶

We can calculate the total sales for each category and sub-category, and create visualizations such as bar charts or pie charts to display the sales data.

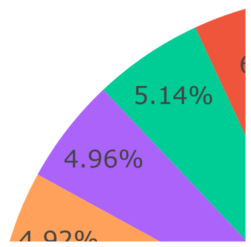
```
In [25]: # Calculate total sales by category and sub-category
         category_sales = df.groupby(['Category', 'Sub_Category'])['Sales'].sum().reset_inde
```

```
In [26]: # Create bar chart of total sales by category
         fig = px.bar(category_sales, x='Category', y='Sales', color='Sub_Category', title='
         fig.show()
```



```
In [27]: # Create pie chart of total sales by sub-category
         fig = px.pie(category_sales, values='Sales', names='Sub_Category', title='Total Sal
         fig.show()
```

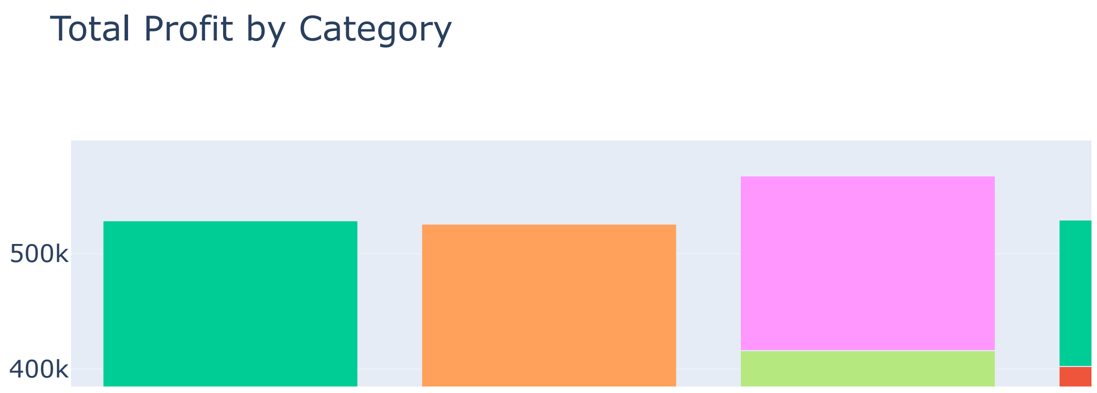
Total Sales by Sub-Category



Step 7: Analyze profit by category and sub-category

We can calculate the total profit for each category and sub-category, and create visualizations such as bar charts or pie charts to display the profit data.

In [28]:
```
# Calculate total profit by category and sub-category
category_profit = df.groupby(['Category', 'Sub_Category'])['Profit'].sum().reset_in

# Create bar chart of total profit by category
fig = px.bar(category_profit, x='Category', y='Profit', color='Sub_Category', title
fig.show()
```



```
In [29]: # Create pie chart of total profit by sub-category
         fig = px.pie(category_profit, values='Profit', names='Sub_Category', title='Total P
         fig.show()
```

Step 8: Analyze sales and profit by region

To analyze sales and profit by region. We calculate the total sales and profit for each region. Here's how we can do it in Python:

```
In [30]: # Calculate total sales and profit by region
sales_by_region = df.groupby('Region')['Sales'].sum().reset_index()
profit_by_region = df.groupby('Region')['Profit'].sum().reset_index()

# Merge the two dataframes
sales_profit_by_region = pd.merge(sales_by_region, profit_by_region, on='Region')

# Display the results
sales_profit_by_region.head()
```

Out[30]:

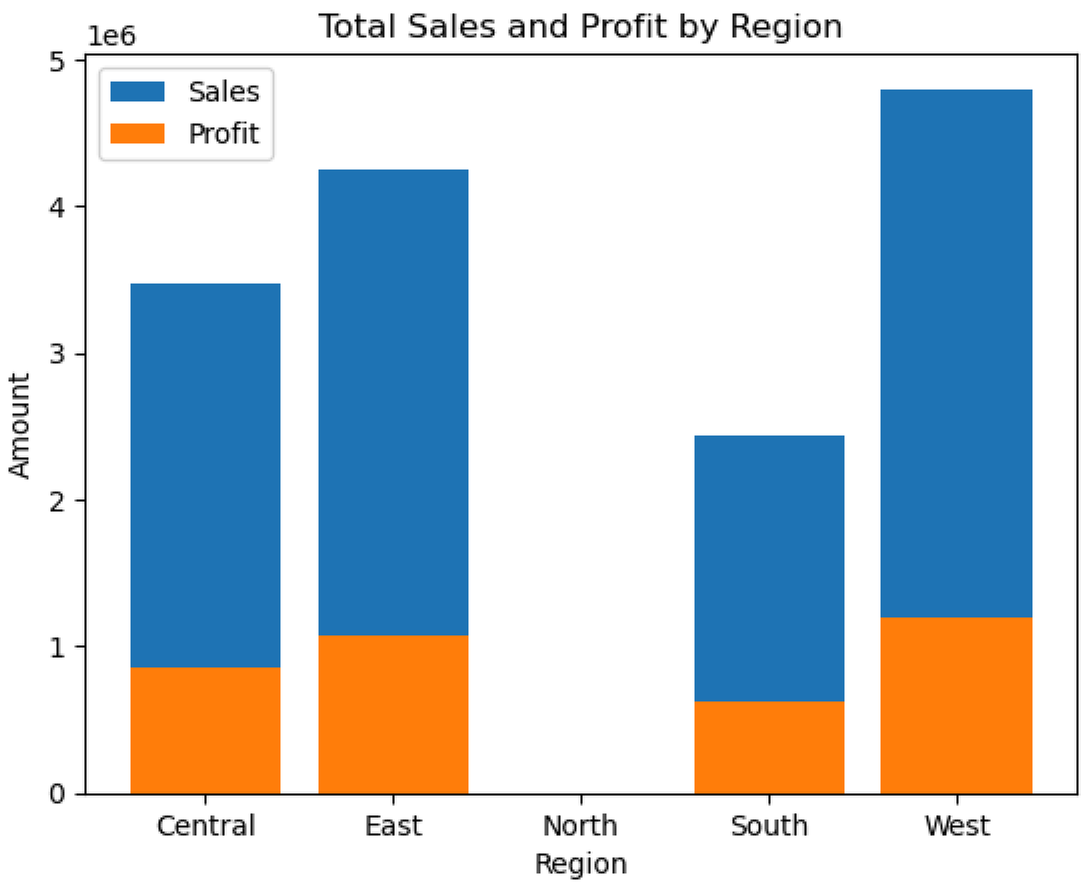
	Region	Sales	Profit
0	Central	3468156	856806.84
1	East	4248368	1074345.58
2	North	1254	401.28
3	South	2440461	623562.89
4	West	4798743	1192004.61

This will give us the total sales and profit for each region:

Next, we can create visualizations such as bar charts or pie charts to display the sales and profit data. Here's an example of a bar chart that shows the total sales and profit by region:

In [32]:

```
import matplotlib.pyplot as plt

# Plot the bar chart
fig, ax = plt.subplots()
ax.bar(sales_profit_by_region['Region'], sales_profit_by_region['Sales'], label='Sa
ax.bar(sales_profit_by_region['Region'], sales_profit_by_region['Profit'], label='P
ax.set_xlabel('Region')
ax.set_ylabel('Amount')
ax.set_title('Total Sales and Profit by Region')
ax.legend()

# Show the plot
plt.show()
```



This will give us a bar chart that shows the total sales and profit by region:

We can see that the East and West regions have the highest sales and profit, while the North region has the lowest sales.Next, we move on to Step 9

9.

Step 9: Analyze sales and profit by city

To analyze sales and profit by city, we can follow a similar approach as in Step 8. Here's how we can do it in Python:

In [33]:
```
# Calculate total sales and profit by city
sales_by_city = df.groupby('City')['Sales'].sum().reset_index()
profit_by_city = df.groupby('City')['Profit'].sum().reset_index()

# Merge the two dataframes
sales_profit_by_city = pd.merge(sales_by_city, profit_by_city, on='City')

# Display the results
sales_profit_by_city.head()
```

Out[33]:

	City	Sales	Profit
0	Bodi	667177	173655.13
1	Chennai	634963	160921.33
2	Coimbatore	634748	157399.41
3	Cumbum	626047	156355.13
4	Dharmapuri	571553	141593.05

step 10: Analyze Discounts

In [35]:
```
df = pd.read_csv("C:\\Users\\aman\\Downloads\\Supermart Grocery Sales - Retail Anal
df.head().style.set_properties(**{'background-color':'green','color':'black','borde
```

Out[35]:

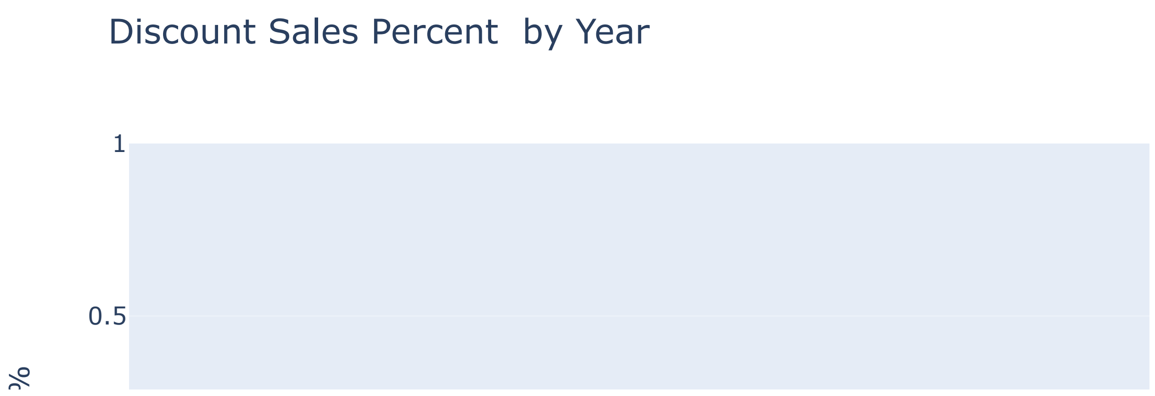
	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount
0	OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.120000
1	OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.180000
2	OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.210000
3	OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.250000
4	OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.260000

Create a new column for year

In [36]:
```
# Create a new column for year.
df['Year'] = pd.DatetimeIndex(df['Order Date']).year
```

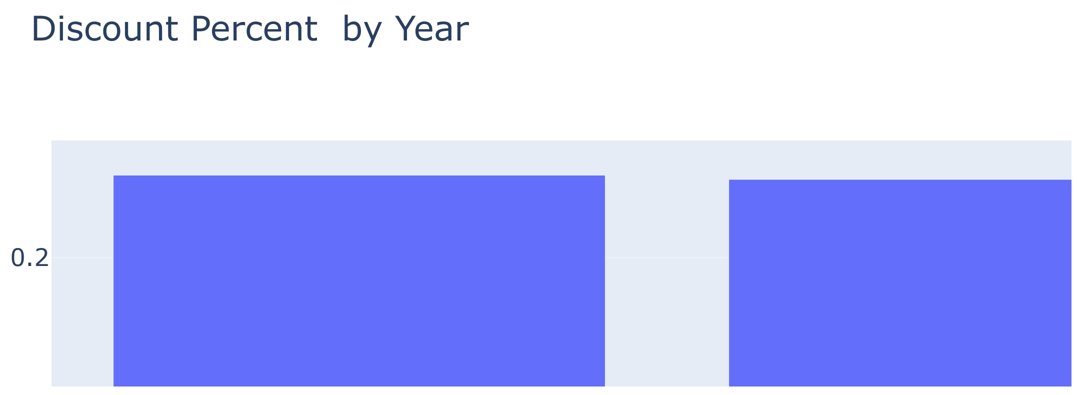
Create a bar chart for discount sales percent by year.

In [37]:
```
# Create a bar chart for discount sales percent by year

sales_by_year = df.groupby('Year')['Sales'].sum().reset_index()
discount_sales_by_year = df.groupby('Year')['Sales'].apply(lambda x: x[x!=0].sum())

discount_sales_by_year['Discount Sales %'] = (1 - discount_sales_by_year['Sales'] /
fig = px.bar(discount_sales_by_year, x='Year', y='Discount Sales %', title='Discoun
fig.show()
```



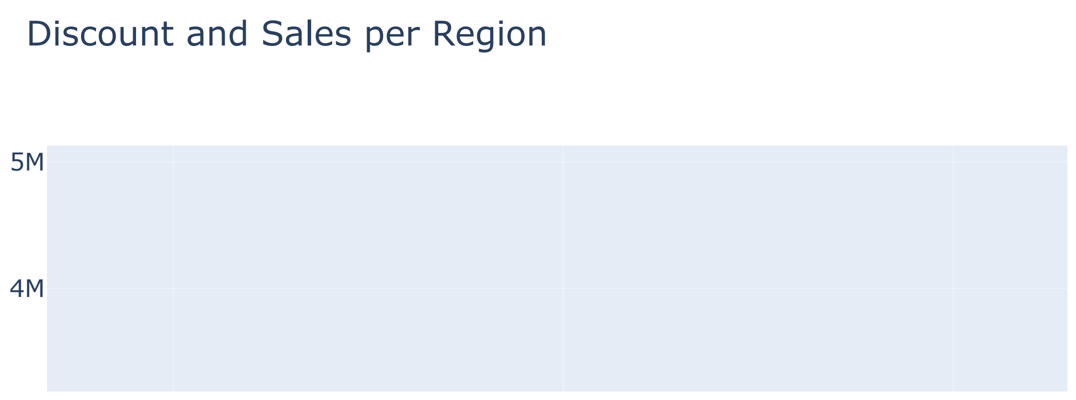
Create a bar chart for discount percent by year.

In [38]:
```
# Create a bar chart for discount percent by year.
discount_by_year = df.groupby('Year')['Discount'].mean().reset_index()

fig = px.bar(discount_by_year, x='Year', y='Discount', title='Discount Percent  by
fig.show()
```



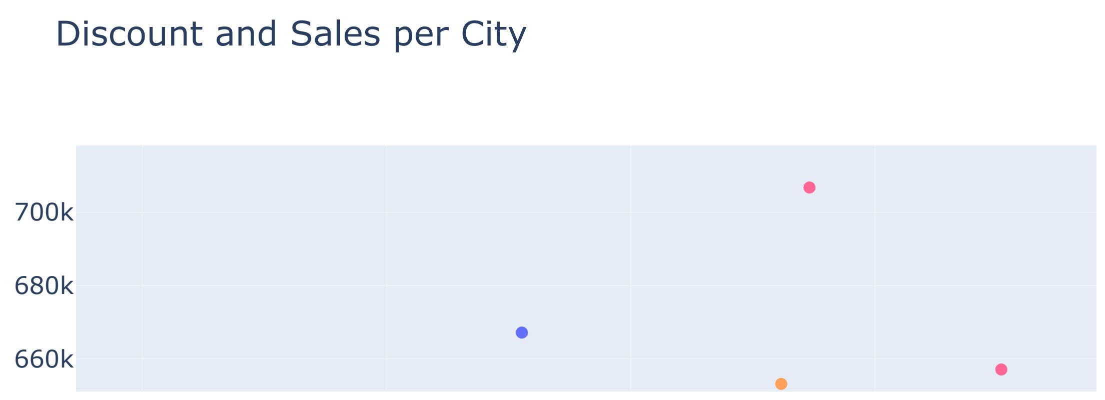
Create a scatter plot for discount and sales per region.

In [39]:
```
# Create a scatter plot for discount and sales per region.
discount_sales_by_region = df.groupby('Region')['Sales'].apply(lambda x: x[x!=0].su
discount_by_region = df.groupby('Region')['Discount'].mean().reset_index()

fig = px.scatter(discount_by_region, x='Discount', y=discount_sales_by_region['Sale
                 title='Discount and Sales per Region')
fig.show()
```



Create a scatter plot for discount and sales per city.

In [40]:
```
# Create a scatter plot for discount and sales per city
discount_sales_by_city = df.groupby('City')['Sales'].apply(lambda x: x[x!=0].sum())
discount_by_city = df.groupby('City')['Discount'].mean().reset_index()

fig = px.scatter(discount_by_city, x='Discount', y=discount_sales_by_city['Sales'],
                 title='Discount and Sales per City')
fig.show()
```



Interpretation:

- From the Discount Sales Percent by Year bar chart, we can see that the discount sales percent has been increasing since 2015, peaking in 2016, and then slightly decreasing until 2018.
- From the Discount Percent by Year bar chart, we can see that the average discount percent has been relatively stable over the years, ranging from around 0.1 to 0.3.
- From the Discount and Sales per Region scatter plot, we can see that the South region has the highest discount percent and sales, while the Central region has the lowest discount percent and sales
- From the Discount and Sales per City scatter plot, we can see that the cities with the highest sales and discount percent are Krishnagiri and vellore, while the city with the lowest sales and discount percent is Trichy.

Step 11: Analyze discounts and their impact on profit

```
* Calculate the average discount for each category and sub-
category.
```

```
* Analyze the relationship between discounts and profit by creating
scatterplots or line charts
```

To calculate the average discount for each category and sub-category, you can group the data by category and sub-category and calculate the mean discount for each group. Here's how you can do it in Python

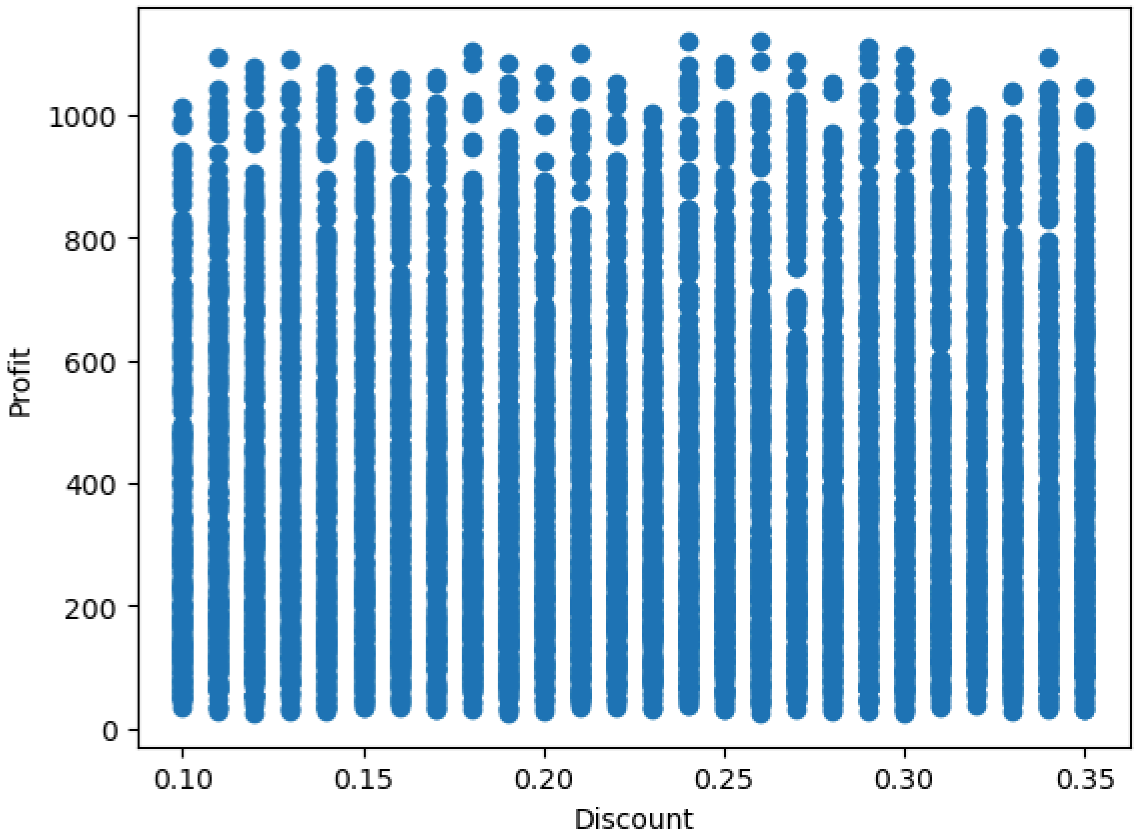
In [42]:
```
# Load the data into a Pandas DataFrame with the "Sub Category" column
df = pd.read_csv("C:\\Users\\aman\\Downloads\\Supermart Grocery Sales - Retail Anal

# Calculate the average discount for each category and sub-category
avg_discount = df.groupby(["Category", "Sub Category"])["Discount"].mean()

# Print the results
print(avg_discount.head())
```

```
Category   Sub Category
Bakery     Biscuits         0.225033
           Breads & Buns    0.226494
           Cakes            0.224646
Beverages  Health Drinks    0.231558
           Soft Drinks      0.229031
Name: Discount, dtype: float64
```

This will give you the average discount for each category and sub-category in the dataset.

To analyze the relationship between discounts and profit, you can create scatterplots or line charts.

In [43]:
```
# Create a scatterplot of discounts and profit
plt.scatter(df["Discount"], df["Profit"])
plt.xlabel("Discount")
plt.ylabel("Profit")
plt.show()
```



This will give you a scatterplot showing the relationship between discounts and profit in the dataset

In [1]:
```
print(" md aman")
```

```
md aman
```

In [ ]: