INTERNSHIP -DATA ANALYSIS

Name-Aryan Zende

Project Report: Grocery Store Sales Analysis Using Python and VS Code

1. Introduction

Grocery stores are an essential part of daily life, meeting the needs of millions for food and household essentials. With the convenience of grocery delivery applications, customers can shop from the comfort of their homes, and each online transaction contributes valuable data. This project focuses on analyzing a dataset sourced from Kaggle, which simulates grocery store sales activities in Tamil Nadu, India. The aim of the project is to extract meaningful insights about store performance and customer purchasing behavior through data processing, analysis, and visualization.

2. Objectives

The objectives of this project include:

- Understanding sales performance by examining transaction trends.
- Identifying popular products and peak transaction times.
- Analyzing customer buying patterns and seasonal influences.
- Providing actionable insights for improving store performance.

3. Tools and Technologies Used

- **Python**: For data analysis, manipulation, and cleaning using libraries like pandas, numpy, and datetime.
- **Data Visualization**: Using matplotlib, seaborn, and plotly for visualizations to provide an intuitive understanding of the data.
- **VS Code**: A popular code editor used for writing, debugging, and running Python code.

- **Dataset Source**: Kaggle, which contains simulated data on grocery store transactions in Tamil Nadu.

4. Dataset Overview

The dataset includes various columns with detailed information about each transaction made at the supermarket. Key columns are as follows:

- **Transaction ID**: Unique identifier for each transaction.
- **Transaction Date**: Date of the transaction.
- **Store ID**: Identifier for the store where the transaction occurred.
- **Product ID**: Identifier for each product.
- **Product Category**: Category to which the product belongs.
- **Quantity**: Quantity of each product purchased.
- **Unit Price**: Price per unit of each product.
- **Total Sale**: Total sale amount for each transaction.

5. Data Preparation and Cleaning

Using Python, data preparation and cleaning involved the following steps:

- **Missing Values Handling**: Checked for missing values in essential columns, filling them where possible or dropping rows with significant missing information.
- **Data Type Conversion**: Ensured date fields were converted to datetime type for time-based analysis.
- **Feature Engineering**: Extracted additional fields such as day, month, and year from the Transaction Date column to aid in seasonal and time-based analysis.
- **Data Validation**: Validated all fields for consistency (e.g., ensuring quantities and prices are non-negative).

6. Exploratory Data Analysis (EDA)

EDA was performed to gain insights into the data, with the following findings:

6.1. Sales Overview

- Calculated total sales, average sales per transaction, and total quantity sold.
- Plotted monthly sales trends to identify peak months and potential seasonality.

6.2. Product Analysis

- Analyzed sales by product category to identify best-selling categories.
- Identified top products based on total sales and quantity sold.

6.3. Customer Behavior

- Analyzed purchase patterns, including the most common purchase days.
- Examined transaction data to identify peak shopping hours.

7. Data Visualization

Visualizations were created using **matplotlib**, **seaborn**, and **plotly** in Python to provide a clear understanding of the data. Key visualizations included:

7.1. Sales Trends

- **Line Charts**: Monthly and weekly sales trends were visualized to identify peak sales periods.
- **Heatmaps**: Used to illustrate sales by day and hour, highlighting busy periods for stores.

7.2. Product Performance

- **Bar Charts**: Displayed top-selling product categories and best-selling products.
- **Pie Charts**: Showed the percentage of sales by product category for quick insights.

7.3. Customer Insights

- **Histogram**: Analyzed transaction frequency to understand customer shopping habits.
- **Box Plot**: Used to examine transaction amounts and detect outliers.

8. Key Findings

Based on the EDA and visualizations, several insights emerged:

- **Sales Patterns**: Peak sales occurred during specific months, suggesting seasonal buying trends.

- **Popular Products**: Certain product categories consistently generated higher sales, indicating customer preferences.
- **Customer Behavior**: Most transactions occurred during specific hours and days, highlighting the busiest times for stores.
- **Store Performance**: Some stores outperformed others in terms of sales volume, likely influenced by location or customer demographics.

9. Conclusion and Recommendations

This project provided valuable insights into grocery store performance and customer purchasing patterns:

- **Stocking Strategy**: Stock popular items in higher quantities, particularly during peak months, to maximize sales.
- **Promotional Timing**: Implement promotions during off-peak periods or targeted hours to boost sales.

- **Staff Scheduling**: Align staff scheduling with peak shopping hours to improve customer service efficiency.

10. Future Scope

- **Customer Segmentation**: Further analysis could be done to segment customers based on buying patterns for targeted marketing.
- **Predictive Analysis**: Machine learning techniques could be applied to predict sales trends, enabling data-driven decision-making for inventory management.

This project report provides a structured approach to analyzing grocery store transaction data, generating insights through Python and visualizations in VS Code. By leveraging these insights, grocery stores can make informed decisions to improve performance and customer satisfaction.

supermarket-sales-analysis

October 30, 2024

Aryan Zende

Unified Mentor

DATA SCIENCE INTERNSHIP

Sales Data Visualization Using Python

This project involves visualizing product sales data to understand the impact of advertising expenditures across various platforms. Through detailed data visualization using Python, this analysis helps identify trends and insights, enabling optimization of advertising strategies to enhance sales potential.

Library Imports

[3]:
```
# Importing Necessary Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
<p style="text-align: center; font-family: Verdana; font-size: 20px; padding: 10px; margin: 0;'
    Reading CSV File
</p>
```

[4]:
```
# Reading CSV File
df = pd.read_csv(r"C:\Users\hp\Downloads\Supermarket\Supermart Grocery Sales -␣
 ↪Retail Analytics Dataset.csv")
df.sample(5)
```

[4]:

	Order ID	Customer Name	Category	Sub Category	City	Order Date	Region	Sales	Discount	Profit	State
1935	OD1936	Adavan	Beverages	Soft Drinks	Dindigul	12/21/2015	West	696	0.10	34.80	Tamil Nadu
6287	OD6288	Sharon	Oil & Masala	Masalas	Bodi	8/29/2018	Central	1700	0.11	204.00	Tamil Nadu
7028	OD7029	Willams	Eggs, Meat & Fish	Eggs	Kanyakumari	9/22/2015	Central	1134	0.23	90.72	Tamil Nadu
7363	OD7364	Esther	Eggs, Meat & Fish	Fish	Tenkasi						
1471	OD1472	Ridhesh	Oil & Masala	Spices	Theni						

```
7363  11-05-2016     East   1229      0.20  184.35  Tamil Nadu
1471  08-05-2015  Central   2175      0.23  674.25  Tamil Nadu
```

```
<p style="text-align: center; font-family: Verdana; font-size: 20px; padding: 10px; margin: 0;'
    Dropping Unnecessary Column
</p>
```

[5]:
```
# Dropping Order ID Column
df.drop(["Order ID"], axis=1, inplace=True)
```

```
<p style="text-align: center; font-family: Verdana; font-size: 20px; padding: 10px; margin: 0;'
    Shape of the DataFrame
</p>
```

[6]:
```
# Shape of the DataFrame
df.shape
```

[6]:
```
(9994, 10)
```

Size of the DataFrame

[7]:
```
# Size of the DataFrame
df.size
```

[7]:
```
99940
```

Index of the DataFrame

[8]:
```
# Index of the DataFrame
df.index
```

[8]:
```
RangeIndex(start=0, stop=9994, step=1)
```

Columns in the DataFrame

[9]:
```
# Columns in the DataFrame
df.columns
```

[9]:
```
Index(['Customer Name', 'Category', 'Sub Category', 'City', 'Order Date',
       'Region', 'Sales', 'Discount', 'Profit', 'State'],
      dtype='object')
```

Info of the DataFrame

[10]:
```
# Info of the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 10 columns):
```

```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Customer Name  9994 non-null   object
 1   Category       9994 non-null   object
 2   Sub Category   9994 non-null   object
 3   City           9994 non-null   object
 4   Order Date     9994 non-null   object
 5   Region         9994 non-null   object
 6   Sales          9994 non-null   int64
 7   Discount       9994 non-null   float64
 8   Profit         9994 non-null   float64
 9   State          9994 non-null   object
dtypes: float64(2), int64(1), object(7)
memory usage: 780.9+ KB
```

Null values in the DataFrame

[11]:
```
# Null values in the DataFrame
df.isna().sum()
```

[11]:
```
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
```

Unique values in the DataFrame

[12]:
```
# Unique values in Customer Name Column
df["Customer Name"].unique()
```

[12]:
```
array(['Harish', 'Sudha', 'Hussain', 'Jackson', 'Ridhesh', 'Adavan',
       'Jonas', 'Hafiz', 'Krithika', 'Ganesh', 'Yadav', 'Sharon', 'Peer',
       'Sundar', 'Ramesh', 'Alan', 'Arutra', 'Haseena', 'Verma', 'Muneer',
       'Veronica', 'Shah', 'Mathew', 'Akash', 'Anu', 'Sabeela', 'James',
       'Willams', 'Malik', 'Amrish', 'Vince', 'Suresh', 'Esther', 'Yusuf',
       'Komal', 'Veena', 'Shree', 'Roshan', 'Sudeep', 'Vinne', 'Vidya',
       'Arvind', 'Kumar', 'Amy', 'Ravi', 'Sheeba', 'Ram', 'Rumaiza',
       'Aditi', 'Surya'], dtype=object)
```

[13]:
```
# Unique values in Category Column
df["Category"].unique()
```

[13]:
```
array(['Oil & Masala', 'Beverages', 'Food Grains', 'Fruits & Veggies',
       'Bakery', 'Snacks', 'Eggs, Meat & Fish'], dtype=object)
```

[14]:
```
# Unique values in Sub Category Column
df["Sub Category"].unique()
```

[14]:
```
array(['Masalas', 'Health Drinks', 'Atta & Flour', 'Fresh Vegetables',
       'Organic Staples', 'Fresh Fruits', 'Biscuits', 'Cakes',
       'Chocolates', 'Eggs', 'Cookies', 'Chicken', 'Edible Oil & Ghee',
       'Mutton', 'Soft Drinks', 'Dals & Pulses', 'Organic Vegetables',
       'Noodles', 'Organic Fruits', 'Fish', 'Spices', 'Rice',
       'Breads & Buns'], dtype=object)
```

[15]:
```
# Unique values in City Column
df["City"].unique()
```

[15]:
```
array(['Vellore', 'Krishnagiri', 'Perambalur', 'Dharmapuri', 'Ooty',
       'Trichy', 'Ramanadhapuram', 'Tirunelveli', 'Chennai', 'Karur',
       'Namakkal', 'Dindigul', 'Kanyakumari', 'Bodi', 'Tenkasi',
       'Viluppuram', 'Madurai', 'Salem', 'Cumbum', 'Nagercoil',
       'Pudukottai', 'Theni', 'Coimbatore', 'Virudhunagar'], dtype=object)
```

[16]:
```
# Unique values in Region Column
df["Region"].unique()
```

[16]:
```
array(['North', 'South', 'West', 'Central', 'East'], dtype=object)
```

Converting OrderDate Column to DateTime Format

[17]:
```
# Sample Data from Order Date Column
df["Order Date"].sample(5)
```

[17]:
```
6843    11/18/2015
6200    07-08-2015
7452    04-04-2015
2054     8/18/2017
7147    12/18/2017
Name: Order Date, dtype: object
```

[18]:
```
# Converting Order Date Column to Pandas DateTime Format
# By using (format="mixed") because Order Date has mixed format data (8/27/
  ↪2016, 06-11-2016 etc.)
df["Order Date"] = pd.to_datetime(df["Order Date"], format="mixed")
```

[19]:
```
# Sample Data from Order Date Column after transformation
df["Order Date"].sample(5)
```

[19]:
```
3860   2015-09-05
7538   2017-11-20
5139   2018-12-03
9964   2018-05-30
8045   2018-12-16
Name: Order Date, dtype: datetime64[ns]
```

Extracting Year, Month and Date from OrderDate Column

[20]:
```
# Extracting Year from Order Date Column
df["Year"] = df["Order Date"].dt.year

# Extracting Month Names from Order Date Column
df["Month"] = df["Order Date"].dt.month_name()

# Extracting Date from Order Date Column
df["Date"] = df["Order Date"].dt.day
```

Extracting Discount Amount from Discount Column

[21]:
```
# We are extracting Discount Amount from Discount Percentage
# By using formula : CP = (SP*100/100-Discount%)
# And then we are subtracting CP from SP to get Discount Amount
df["Discount_Amt"] = round((df["Sales"]*100)/(100-(df["Discount"]*100))).
 ↪astype(int) - df["Sales"]
```

Dropping Unnecessary Column

[22]:
```
# Dropping Order Date Column
df.drop(["Order Date","Discount"], axis=1, inplace=True)
```

Final DataFrame

[23]:
```
# Final DataFrame
df.head(5)
```

[23]:

	Customer Name	Category	Sub Category	City	Region	Sales	Profit	State	Year	Month	Date	Discount_Amt
0	Harish	Oil & Masala	Masalas	Vellore	North	1254	401.28	Tamil Nadu	2017	November	8	171
1	Sudha	Beverages	Health Drinks	Krishnagiri	South	749	149.80	Tamil Nadu	2017	November	8	164
2	Hussain	Food Grains	Atta & Flour	Perambalur	West	2360	165.20	Tamil Nadu	2017	June	12	627
3	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	South	896	89.60	Tamil Nadu	2016	October	11	299
4	Ridhesh	Food Grains	Organic Staples	Ooty	South							

```
4   2355  918.45  Tamil Nadu  2016   October    11          827
```

No. of Products sold in each Category

[24]:
```
# Count of products sold in each Category
df["Category"].value_counts()
```

[24]:
```
Category
Snacks               1514
Eggs, Meat & Fish    1490
Fruits & Veggies     1418
Bakery               1413
Beverages            1400
Food Grains          1398
Oil & Masala         1361
Name: count, dtype: int64
```

No. of Products sold in each Sub Category

[25]:
```
# Count of products sold in each Sub Category
df["Sub Category"].value_counts()
```

[25]:
```
Sub Category
Health Drinks         719
Soft Drinks           681
Cookies               520
Breads & Buns         502
Chocolates            499
Noodles               495
Masalas               463
Biscuits              459
Cakes                 452
Edible Oil & Ghee     451
Spices                447
Mutton                394
Eggs                  379
Organic Staples       372
Fresh Fruits          369
Fish                  369
Fresh Vegetables      354
Atta & Flour          353
Organic Fruits        348
Chicken               348
Organic Vegetables    347
Dals & Pulses         343
Rice                  330
Name: count, dtype: int64
```

No. of Products sold in each City

[26]:
```
# Count of products sold in each City
df["City"].value_counts()
```

[26]:
```
City
Kanyakumari       459
Tirunelveli       446
Bodi              442
Krishnagiri       440
Vellore           435
Perambalur        434
Tenkasi           432
Chennai           432
Salem             431
Karur             430
Pudukottai        430
Coimbatore        428
Ramanadhapuram    421
Cumbum            417
Virudhunagar      416
Madurai           408
Ooty              404
Namakkal          403
Viluppuram        397
Dindigul          396
Theni             387
Dharmapuri        376
Nagercoil         373
Trichy            357
Name: count, dtype: int64
```

No. of Products sold in each Region

[27]:
```
# Count of products sold in each Region
df["Region"].value_counts()
```

[27]:
```
Region
West       3203
East       2848
Central    2323
South      1619
North         1
Name: count, dtype: int64
```

[28]:
```
# As we have only 1 sales data for North region
# We can remove it for a balance data overall
df = df[df["Region"] != "North"]
```

[29]:
```
# Count of products sold in each Region after transformation
df["Region"].value_counts()
```

[29]:
```
Region
West       3203
East       2848
Central    2323
South      1619
Name: count, dtype: int64
```

No. of Products sold each Year, Month and Date

[30]:
```
# Count of products sold each Year
df["Year"].value_counts()
```

[30]:
```
Year
2018    3312
2017    2586
2016    2102
2015    1993
Name: count, dtype: int64
```

[31]:
```
# Count of products sold each Month
df["Month"].value_counts()
```

[31]:
```
Month
November     1470
December     1408
September    1383
October       819
May           735
June          717
July          710
August        706
March         696
April         668
January       381
February      300
Name: count, dtype: int64
```

[32]:
```
# Count of products sold each Date
df["Date"].value_counts()
```

[32]:
```
Date
20    398
21    396
2     379
```

```
5     366
3     365
26    365
11    359
23    358
8     355
17    352
14    348
12    345
9     344
13    338
1     337
19    335
25    334
18    328
4     308
24    302
10    300
7     298
30    295
28    285
6     285
22    284
27    273
15    271
16    269
29    238
31    183
Name: count, dtype: int64
```

Total sale in each Category

[33]:
```
# Total sale in each Category
df.groupby(["Category"])["Sales"].sum().sort_values(ascending=False)
```

[33]:
```
Category
Eggs, Meat & Fish    2267401
Snacks               2237546
Food Grains          2115272
Bakery               2112281
Fruits & Veggies     2100727
Beverages            2085313
Oil & Masala         2037188
Name: Sales, dtype: int64
```

Total sale in each Sub Category

[34]:
```
# Total sale in each Sub Category
df.groupby(["Sub Category"])["Sales"].sum().sort_values(ascending=False)
```

[34]:
```
Sub Category
Health Drinks         1051439
Soft Drinks           1033874
Cookies                768213
Breads & Buns          742586
Noodles                735435
Chocolates             733898
Masalas                696226
Cakes                  685612
Biscuits               684083
Spices                 672876
Edible Oil & Ghee      668086
Mutton                 611200
Eggs                   575156
Fish                   560548
Organic Staples        558929
Fresh Fruits           551212
Atta & Flour           534649
Fresh Vegetables       525842
Dals & Pulses          523371
Chicken                520497
Organic Vegetables     520271
Organic Fruits         503402
Rice                   498323
Name: Sales, dtype: int64
```

Total sale in each Region

[35]:
```
# Total sale in each Region
df.groupby(["Region"])["Sales"].sum().sort_values(ascending=False)
```

[35]:
```
Region
West       4798743
East       4248368
Central    3468156
South      2440461
Name: Sales, dtype: int64
```

Total sale in each City

[36]:
```
# Total sale in each City
df.groupby(["City"])["Sales"].sum().sort_values(ascending=False)
```

[36]:
```
City
Kanyakumari       706764
```

```
Vellore           675296
Bodi              667177
Tirunelveli       659812
Perambalur        659738
Salem             657093
Pudukottai        653179
Tenkasi           643652
Karur             642273
Krishnagiri       637273
Chennai           634963
Coimbatore        634748
Ramanadhapuram    634386
Cumbum            626047
Madurai           617836
Virudhunagar      606820
Ooty              599292
Namakkal          598530
Viluppuram        581274
Theni             579553
Dindigul          575631
Dharmapuri        571553
Nagercoil         551435
Trichy            541403
Name: Sales, dtype: int64
```

Total sale in each Month

[37]:
```
# Total sale in each Month
df.groupby(["Month"])["Sales"].sum().sort_values(ascending=False)
```

[37]:
```
Month
November     2192670
December     2088076
September    2064266
October      1243289
July         1089385
May          1086920
June         1057808
March        1053980
August       1046807
April         998453
January       577972
February      456102
Name: Sales, dtype: int64
```

Total sale in each Year

[38]:
```
# Total sale in each Year
df.groupby(["Year"])["Sales"].sum().sort_values(ascending=False)
```

[38]:
```
Year
2018    4977512
2017    3870658
2016    3131959
2015    2975599
Name: Sales, dtype: int64
```

Total profit in each Category

[39]:
```
# Total profit in each Category
df.groupby(["Category"])["Profit"].sum().sort_values(ascending=False)
```

[39]:
```
Category
Snacks                568178.85
Eggs, Meat & Fish     567357.22
Fruits & Veggies      530400.38
Food Grains           529162.64
Bakery                528521.06
Beverages             525605.76
Oil & Masala          497494.01
Name: Profit, dtype: float64
```

Total profit in each Sub Category

[40]:
```
# Total profit in each Sub Category
df.groupby(["Sub Category"])["Profit"].sum().sort_values(ascending=False)
```

[40]:
```
Sub Category
Health Drinks         267469.79
Soft Drinks           258135.97
Noodles               193685.81
Breads & Buns         190764.98
Cookies               190643.70
Chocolates            183849.34
Biscuits              169357.62
Masalas               168597.83
Edible Oil & Ghee     168593.58
Cakes                 168398.46
Spices                160302.60
Mutton                151389.40
Fish                  147248.01
Eggs                  144669.92
Organic Staples       144136.89
Fresh Fruits          134668.35
Organic Vegetables    133596.37
```

```
Fresh Vegetables      131273.33
Organic Fruits        130862.33
Dals & Pulses         130232.29
Atta & Flour          127861.10
Rice                  126932.36
Chicken               124049.89
Name: Profit, dtype: float64
```

Total profit in each Region

[41]:
```
# Total profit in each Region
df.groupby(["Region"])["Profit"].sum().sort_values(ascending=False)
```

[41]:
```
Region
West       1192004.61
East       1074345.58
Central     856806.84
South       623562.89
Name: Profit, dtype: float64
```

Total profit in each City

[42]:
```
# Total profit in each City
df.groupby(["City"])["Profit"].sum().sort_values(ascending=False)
```

[42]:
```
City
Vellore           173671.73
Bodi              173655.13
Kanyakumari       172217.74
Perambalur        171132.19
Karur             169305.94
Tirunelveli       165169.01
Pudukottai        164072.63
Chennai           160921.33
Salem             160899.30
Krishnagiri       160477.48
Ramanadhapuram    158951.03
Coimbatore        157399.41
Cumbum            156355.13
Tenkasi           156230.72
Madurai           152548.61
Virudhunagar      150816.69
Ooty              150078.92
Namakkal          145502.10
Dindigul          144872.95
Viluppuram        144200.64
Theni             142739.78
Dharmapuri        141593.05
```

```
Nagercoil          137848.47
Trichy             136059.94
Name: Profit, dtype: float64
```

Total profit in each Month

[43]:
```
# Total profit in each Month
df.groupby(["Month"])["Profit"].sum().sort_values(ascending=False)
```

[43]:
```
Month
November     555646.77
December     530036.44
September    517788.56
October      309376.90
July         274594.57
March        267347.33
May          263643.40
June         263296.16
August       258912.09
April        247476.97
January      142518.52
February     116082.21
Name: Profit, dtype: float64
```

Total profit in each Year

[44]:
```
# Total profit in each Year
df.groupby(["Year"])["Profit"].sum().sort_values(ascending=False)
```

[44]:
```
Year
2018    1244182.88
2017     952814.94
2016     797192.99
2015     752529.11
Name: Profit, dtype: float64
```

Customer's with Highest Total Orders

[45]:
```
# Customer's with highest total orders
df.groupby(["Customer Name"])["Sales"].sum().sort_values(ascending=False).
  ↪head(10)
```

[45]:
```
Customer Name
Krithika    334361
Amrish      333351
Verma       331665
Arutra      325720
Vidya       321798
```

```
Vinne       319565
Shah        318588
Suresh      315973
Adavan      315341
Surya       312645
Name: Sales, dtype: int64
```

Most purchased product from each category by Customer's

[46]:
```
# Most purchased product from each category by Customer's
df.groupby(["Customer Name","Category"])["Category"].count().
  ↪sort_values(ascending=False).head(10)
```

[46]:
```
Customer Name  Category
Veronica       Fruits & Veggies     43
Ganesh         Snacks               42
Yusuf          Oil & Masala         42
Amrish         Bakery               42
Vidya          Eggs, Meat & Fish    41
Verma          Eggs, Meat & Fish    41
Vinne          Snacks               40
Amrish         Food Grains          40
Shah           Bakery               40
Ram            Beverages            40
Name: Category, dtype: int64
```

Most purchased product from each sub category by Customer's

[47]:
```
# Most purchased product from each sub category by Customer's
df.groupby(["Customer Name","Sub Category"])["Sub Category"].count().
  ↪sort_values(ascending=False).head(10)
```

[47]:
```
Customer Name  Sub Category
Yadav          Soft Drinks      23
Amy            Health Drinks    22
Ram            Health Drinks    22
Veena          Soft Drinks      21
Ridhesh        Soft Drinks      20
Krithika       Cakes            20
James          Health Drinks    20
Rumaiza        Soft Drinks      20
Shah           Health Drinks    19
Amrish         Health Drinks    19
Name: Sub Category, dtype: int64
```

Most order placed from different cities by Customer's

[48]:
```
# Most order placed from different cities by Customer's
df.groupby(["Customer Name","City"])["City"].count().
 ↪sort_values(ascending=False).head(10)
```

[48]:
```
Customer Name  City
Sharon         Ooty              20
James          Ramanadhapuram    18
Sudeep         Madurai           18
Amrish         Viluppuram        18
Esther         Virudhunagar      17
Veronica       Tenkasi           16
Verma          Cumbum            16
Surya          Cumbum            16
Yadav          Perambalur        16
Jonas          Cumbum            16
Name: City, dtype: int64
```

Most order placed from different regions by Customer's

[49]:
```
# Most order placed from different regions by Customer's
df.groupby(["Customer Name","Region"])["Region"].count().
 ↪sort_values(ascending=False).head(10)
```

[49]:
```
Customer Name  Region
Krithika       West      90
Malik          West      77
Veena          West      76
Veronica       West      75
Mathew         West      75
Arutra         West      74
Muneer         West      73
Amrish         West      73
Krithika       East      73
Peer           West      73
Name: Region, dtype: int64
```

Most order placed in different months by Customer's

[50]:
```
# Most order placed in different months by Customer's
df.groupby(["Customer Name","Month"])["Month"].count().
 ↪sort_values(ascending=False).head(10)
```

[50]:
```
Customer Name  Month
Krithika       December     42
Yusuf          November     41
Veronica       September    41
Hussain        November     40
Arutra         December     39
```

```
Vinne          December     39
Arvind         November     37
Akash          September    37
Sudeep         November     37
Komal          September    37
Name: Month, dtype: int64
```

Most order placed in different years by Customer's

[51]:
```
# Most order placed in different years by Customer's
df.groupby(["Customer Name","Year"])["Year"].count().
  ↪sort_values(ascending=False).head(10)
```

[51]:
```
Customer Name  Year
Suresh         2018    80
Adavan         2018    79
Surya          2018    77
Verma          2018    76
Ram            2018    74
Veronica       2018    73
Akash          2018    72
Ravi           2018    72
Vidya          2018    72
Krithika       2018    72
Name: Year, dtype: int64
```

Total Discount on each Product Category

[52]:
```
# Total Discount on each Product Category
df.groupby(["Category"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[52]:
```
Category
Eggs, Meat & Fish    694041
Snacks               663963
Food Grains          652872
Fruits & Veggies     650275
Beverages            648970
Bakery               638128
Oil & Masala         617326
Name: Discount_Amt, dtype: int64
```

Total Discount on each Sub Product Category

[53]:
```
# Total Discount on each Sub Product Category
df.groupby(["Sub Category"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[53]:
```
Sub Category
Health Drinks         328816
```

```
Soft Drinks           320154
Breads & Buns         226696
Cookies               224374
Chocolates            221921
Noodles               217668
Edible Oil & Ghee     209592
Masalas               207747
Cakes                 206505
Biscuits              204927
Spices                199987
Mutton                185879
Organic Staples       175957
Fish                  175879
Eggs                  173843
Fresh Fruits          167977
Organic Vegetables    164475
Atta & Flour          163358
Fresh Vegetables      162803
Dals & Pulses         158886
Chicken               158440
Organic Fruits        155020
Rice                  154671
Name: Discount_Amt, dtype: int64
```

Total Discount in each Region

[54]:
```
# Total Discount in each Region
df.groupby(["Region"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[54]:
```
Region
West       1450419
East       1300786
Central    1068850
South       745520
Name: Discount_Amt, dtype: int64
```

Total Discount in each City

[55]:
```
# Total Discount in each City
df.groupby(["City"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[55]:
```
City
Vellore           212268
Kanyakumari       211570
Perambalur        207825
Tirunelveli       200610
Tenkasi           199804
Salem             197370
```

```
Karur              195645
Pudukottai         195166
Ramanadhapuram     194399
Chennai            194311
Cumbum             193613
Bodi               193461
Krishnagiri        191242
Madurai            187467
Coimbatore         186064
Namakkal           184304
Theni              181967
Ooty               180596
Viluppuram         180388
Dharmapuri         179366
Virudhunagar       178404
Trichy             174857
Nagercoil          174267
Dindigul           170611
Name: Discount_Amt, dtype: int64
```

Total Discount in each Month

[56]:
```
# Total Discount in each Month
df.groupby(["Month"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[56]:
```
Month
November     674780
December     630591
September    629729
October      382904
July         339438
May          327797
June         326871
March        324026
August       318511
April        303432
January      175569
February     131927
Name: Discount_Amt, dtype: int64
```

Total Discount in each Year

[57]:
```
# Total Discount in each Year
df.groupby(["Year"])["Discount_Amt"].sum().sort_values(ascending=False)
```

[57]:
```
Year
2018    1509381
2017    1181457
```

```
2016     961588
2015     913149
Name: Discount_Amt, dtype: int64
```

Customer's with most Discount

[58]:
```
# Customer's with most Discount
df.groupby(["Customer Name"])["Discount_Amt"].sum().
  ↪sort_values(ascending=False).head(10)
```

[58]:
```
Customer Name
Verma       108213
Amrish      103100
Muneer       99528
Vinne        99354
Shah         99249
Krithika     97860
Sheeba       97108
Hussain      96853
Arutra       96494
Mathew       96277
Name: Discount_Amt, dtype: int64
```

Sale of Different Category Products Year Wise

[59]:
```
# Sale of Different Category Products Year Wise
df.pivot_table(index="Year", columns="Category", values="Sales", aggfunc="sum")
```

[59]:
```
Category  Bakery  Beverages  Eggs, Meat & Fish  Food Grains  Fruits & Veggies  \
Year
2015      433979     409091             482982       356704            409212
2016      487965     425719             462055       438736            468441
2017      504263     543575             565101       609338            516400
2018      686074     706928             757263       710494            706674

Category  Oil & Masala  Snacks
Year
2015            423918  459713
2016            412725  436318
2017            562942  569039
2018            637603  772476
```

Sale of Different Category Products Month Wise

[60]:
```
# Sale of Different Category Products Month Wise
df.pivot_table(index="Month", columns="Category", values="Sales", aggfunc="sum")
```

[60]:
Category	Bakery	Beverages	Eggs, Meat & Fish	Food Grains
Month				
April	123181	151748	143530	128790
August	162277	152976	149694	182850
December	298939	265806	344238	291656
February	54701	60188	58331	62983
January	69139	91775	74158	79531
July	168795	137856	151179	166926
June	154397	140633	181754	128165
March	154253	143575	165977	140169
May	135113	163490	173024	143356
November	307450	328019	309944	306260
October	184036	174745	196215	192451
September	300000	274502	319357	292135

Category	Fruits & Veggies	Oil & Masala	Snacks
Month			
April	138084	157369	155751
August	140885	124935	133190
December	311754	270309	305374
February	74313	62044	83542
January	80753	95324	87292
July	162689	149179	152761
June	153431	129080	170348
March	126704	142637	180665
May	152491	182725	136721
November	309473	295323	336201
October	156168	166182	173492
September	293982	262081	322209

Sale of Different Category Products Region Wise

[61]:
```
# Sale of Different Category Products Region Wise
df.pivot_table(index="Region", columns="Category", values="Sales",␣
 ↪aggfunc="sum")
```

[61]:
Category	Bakery	Beverages	Eggs, Meat & Fish	Food Grains	Fruits & Veggies
Region					
Central	448343	501194	516642	463683	526507
East	588241	577206	677830	598868	564237
South	361385	344666	360963	353069	352162
West	714312	662247	711966	699652	657821

Category	Oil & Masala	Snacks
Region		
Central	453293	558494
East	622071	619915

```
South           298859  369357
West            662965  689780
```

Profit from Different Category Products Year Wise

[62]:
```
# Profit from Different Category Products Year Wise
df.pivot_table(index="Year", columns="Category", values="Profit", aggfunc="sum")
```

[62]:
```
Category     Bakery  Beverages  Eggs, Meat & Fish  Food Grains  \
Year
2015      107567.44  104834.61          126057.88     90837.84
2016      124536.04  110201.14          114208.28    113643.27
2017      122078.77  135179.89          135896.22    149775.58
2018      174338.81  175390.12          191194.84    174905.95

Category  Fruits & Veggies  Oil & Masala     Snacks
Year
2015              100586.06     105637.64  117007.64
2016              118900.21     102656.22  113047.83
2017              130630.77     137156.83  142096.88
2018              180283.34     152043.32  196026.50
```

Profit from Different Category Products Month Wise

[63]:
```
# Profit from Different Category Products Month Wise
df.pivot_table(index="Month", columns="Category", values="Profit",␣
 ↪aggfunc="sum")
```

[63]:
```
Category     Bakery  Beverages  Eggs, Meat & Fish  Food Grains  \
Month
April      28491.68   38426.16           37453.93     31279.89
August     40395.81   40139.14           38057.70     42734.55
December   74036.37   67086.34           86623.79     75671.33
February   12636.04   15080.17           15089.23     14680.57
January    18555.36   23958.38           17714.58     18795.67
July       39733.87   35285.65           40908.34     42397.92
June       38539.89   36220.45           44216.92     31192.94
March      42785.93   38049.03           43155.74     32659.45
May        31426.63   38256.46           44998.40     35674.88
November   76083.73   83913.01           77671.30     77997.19
October    44818.93   41466.33           47223.93     51687.74
September  81016.82   67724.64           74243.36     74390.51

Category   Fruits & Veggies  Oil & Masala    Snacks
Month
April              37907.45      36273.10  37644.76
August             34780.18      30230.38  32574.33
December           76006.63      71378.32  79233.66
```

```
February          19410.95      17374.45  21810.80
January           21362.53      22117.10  20014.90
July              40517.79      37037.06  38713.94
June              39272.91      31764.42  42088.63
March             30729.94      36544.52  43422.72
May               38603.40      41634.13  33049.50
November          77585.21      70164.57  92231.76
October           39395.36      40309.54  44475.07
September         74828.03      62666.42  82918.78
```

Profit from Different Category Products Region Wise

[64]:
```
# Profit from Different Category Products Region Wise
df.pivot_table(index="Region", columns="Category", values="Profit",␣
 ↪aggfunc="sum")
```

[64]:
```
Category     Bakery  Beverages  Eggs, Meat & Fish  Food Grains  \
Region
Central   109543.91  128017.14          126587.35    111517.05
East      153741.06  144498.17          164465.08    154932.01
South      89102.07   84058.78           93728.18     94823.55
West      176134.02  169031.67          182576.61    167890.03

Category  Fruits & Veggies  Oil & Masala     Snacks
Region
Central          131699.53     109261.91  140179.95
East             141292.78     155569.22  159847.26
South             92961.26      73909.18   94979.87
West             164446.81     158753.70  173171.77
```

Percent of Total Revenue spend on Discounts

[65]:
```
# Percent of total revenue spend on Discounts
amt_spend = int(df["Discount_Amt"].sum()/df["Sales"].sum()*100)
print(f"{amt_spend}% of total revenue was spend on Discounts.")
```

```
30% of total revenue was spend on Discounts.
```

Setting Plot Style

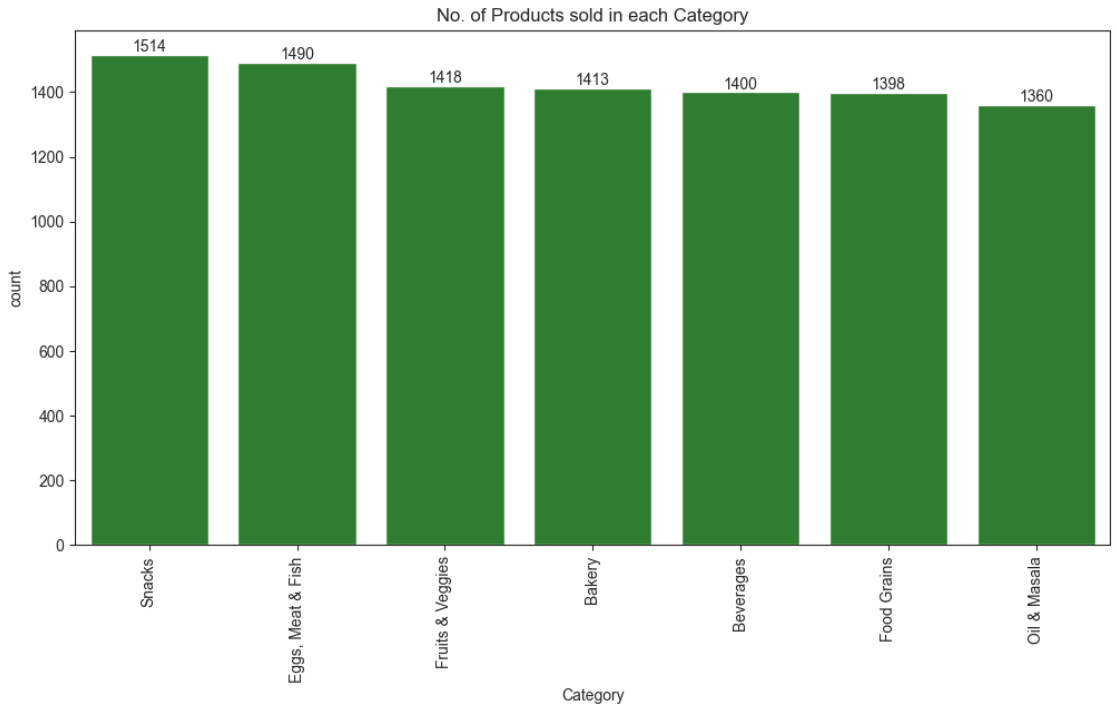
[66]:
```
# Setting Plot Style to "ticks"
sns.set_style("ticks")
```

No. of Products sold in each Category

Insights

• Snacks are the highest selling product in the entire category. • While, Oil and Masala being the lowest selling product in the entire category.

[67]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="Category", color="#228B22", order=df["Category"].
 ↪value_counts().index)
ax.set_title("No. of Products sold in each Category")
ax.tick_params(axis="x", rotation=90)
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```

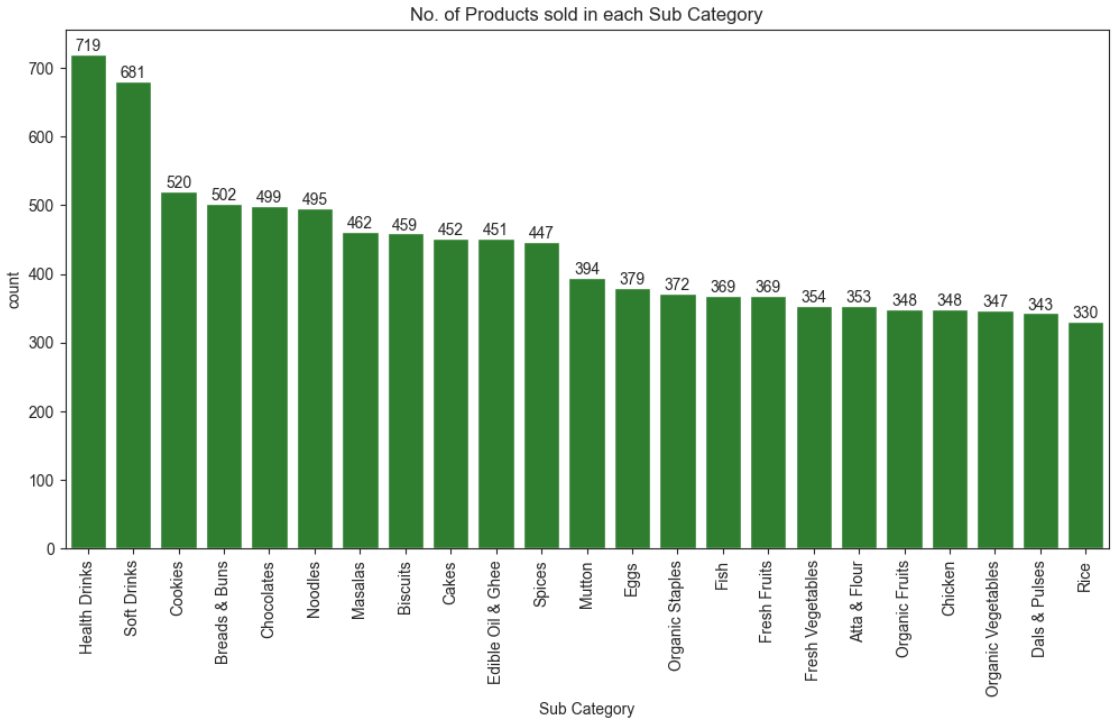


No. of Products sold in each Sub Category

Insights

- Health Drinks and Soft Drinks are the top 2 highest selling products in the entire sub category.
- Cookies is the third highest selling product in the entire sub category. • While, Rice being the lowest selling product in the entire sub category.

[68]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="Sub Category", color="#228B22", order=df["Sub␣
 ↪Category"].value_counts().index)
ax.set_title("No. of Products sold in each Sub Category")
ax.tick_params(axis='x', rotation=90)
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```

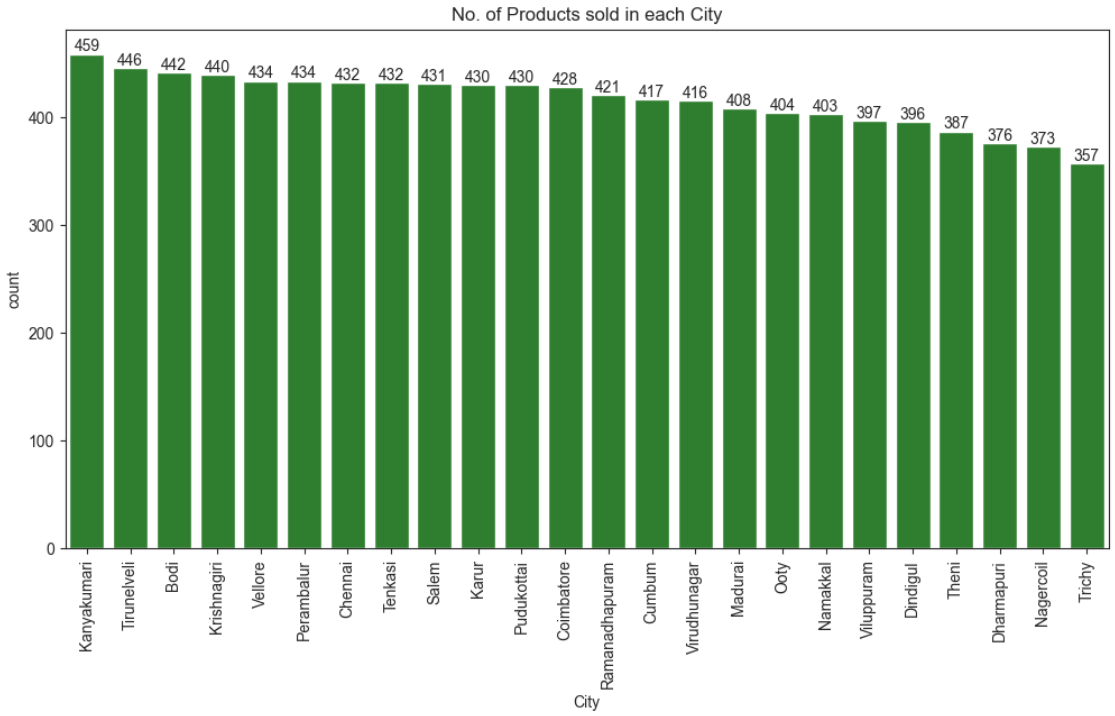


No. of Products sold in each City

Insights

• Kanyakumari is the city with highest number of sales in whole Tamil Nadu. • While, Trichy being the city with lowest number of sales in whole Tamil Nadu.

[69]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="City", color="#228B22", order=df["City"].
 ↪value_counts().index)
ax.set_title("No. of Products sold in each City")
ax.tick_params(axis='x', rotation=90)
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```

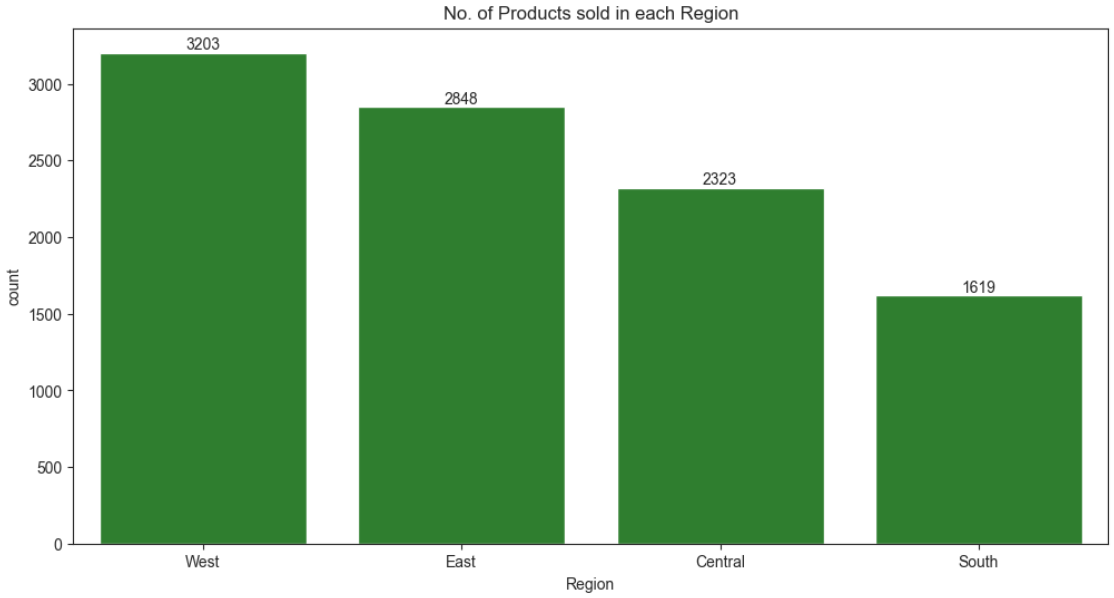


No. of Products sold in each Region

Insights

• West is the region with highest number of sales in Tamil Nadu. • While, South being the region with lowest number of sales in Tamil Nadu.

[70]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="Region", color="#228B22", order=df["Region"].
 ↪value_counts().index)
ax.set_title("No. of Products sold in each Region")
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```

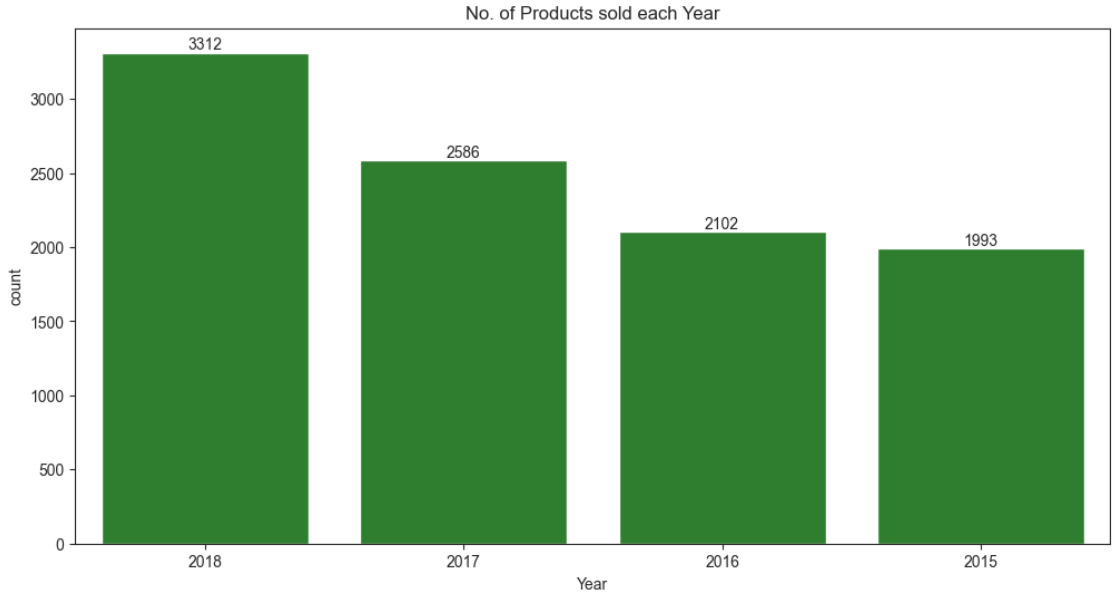


No. of Products sold each Year, Month and Date

Insights

• 2018 was the year with highest number of sales. • While, 2015 being the year with lowest number of sales.

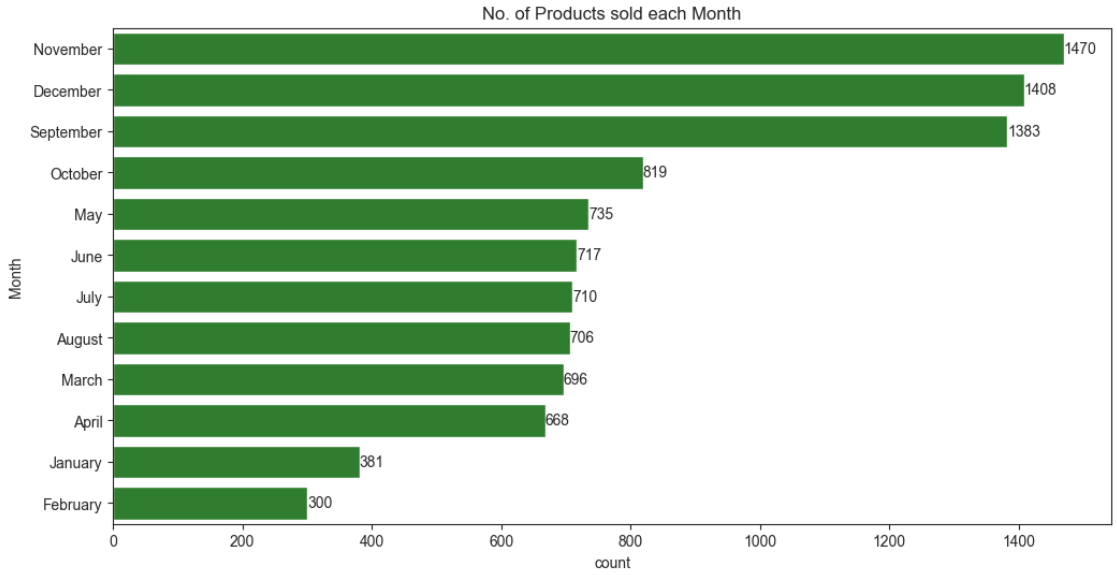
[71]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="Year", color="#228B22", order=df["Year"].
 ↪value_counts().index)
ax.set_title("No. of Products sold each Year")
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```



Insights

• November is the month with highest number of sales. • While, February being the month with lowest number of sales. • High sales are towards the end of the year while low sales are towards the start of the year.

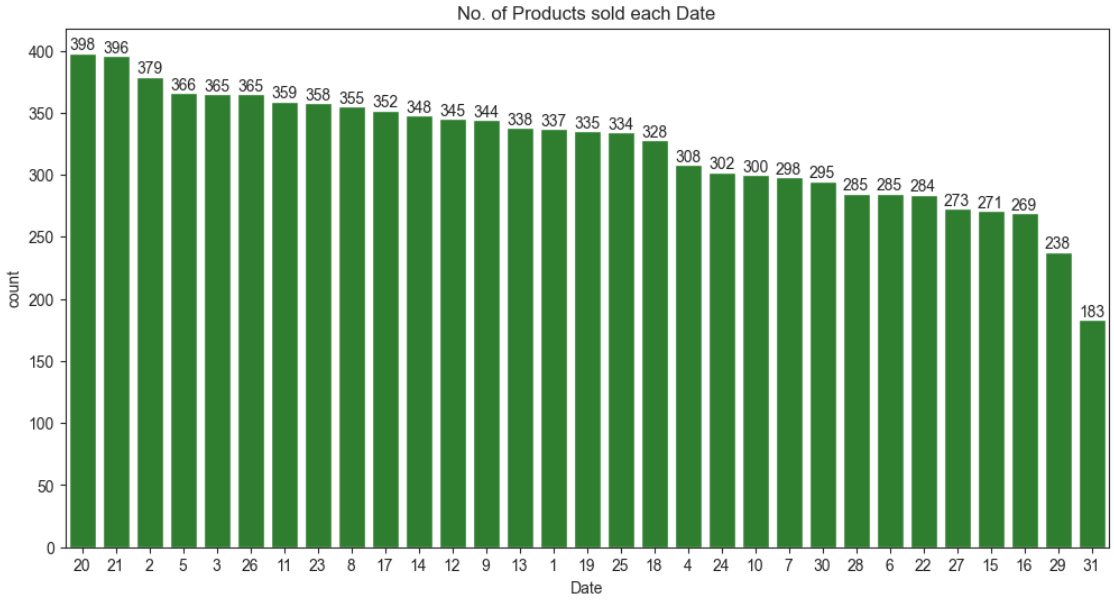
[72]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, y="Month", color="#228B22", order=df["Month"].
 ↪value_counts().index)
ax.set_title("No. of Products sold each Month")
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```



Insights

• 20 and 21 are the dates in each month with highest number of sales. • While, 31 being the date in each month with lowest number of sales. • High sales are towards the mid of each month while low sales are towards the end of each month.

[73]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.countplot(data=df, x="Date", color="#228B22", order=df["Date"].
 ↪value_counts().index)
ax.set_title("No. of Products sold each Date")
ax.bar_label(ax.containers[0], fontsize=10)
plt.show()
```



Total sales in each Category

Insights

• Eggs, Meat & Fish has the highest number of total sales in the entire category. • While, Oil and Masala has the lowest number of total sales in the entire category.

[74]:
```
sales_by_category = df.groupby(["Category"])["Sales"].sum().
 ↪sort_values(ascending=False)
```

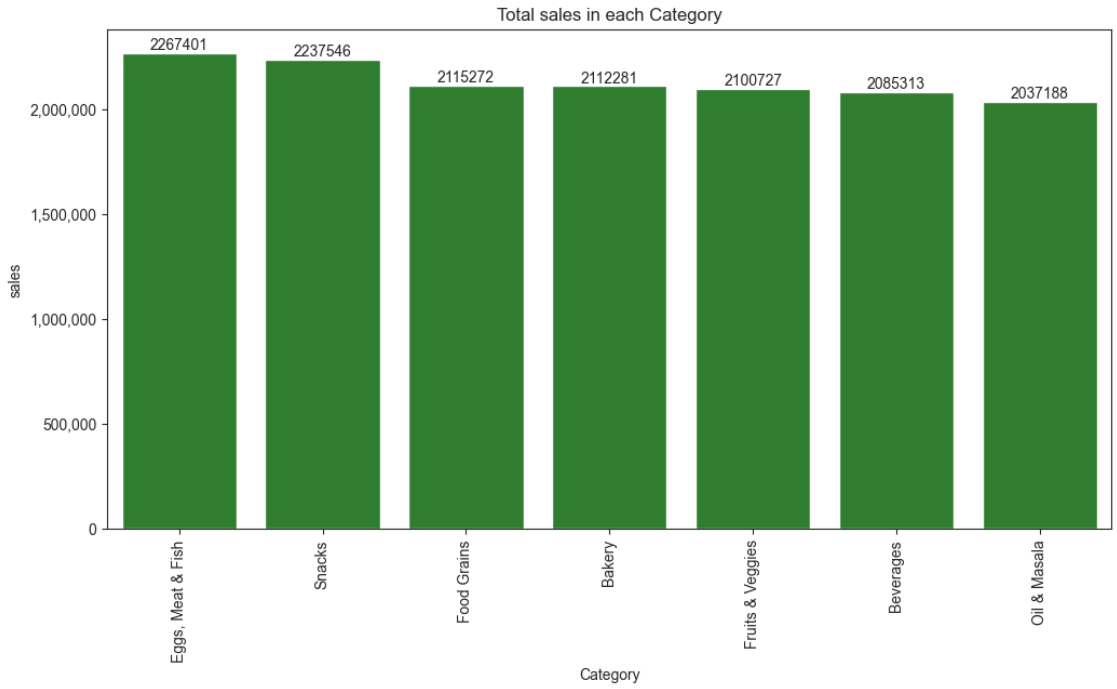
[75]:
```
from matplotlib.ticker import FuncFormatter

fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_category.index, y=sales_by_category.values,
            color="#228B22", order=df.groupby(["Category"])["Sales"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.tick_params(axis="x", rotation=90)
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total sales in each Category")

plt.show()
```



Total sales in each Sub Category

Insights

- Health Drinks and Soft Drinks has the highest number of total sales in the entire sub category.
- While, Rice has the lowest number of total sales in the entire sub category.

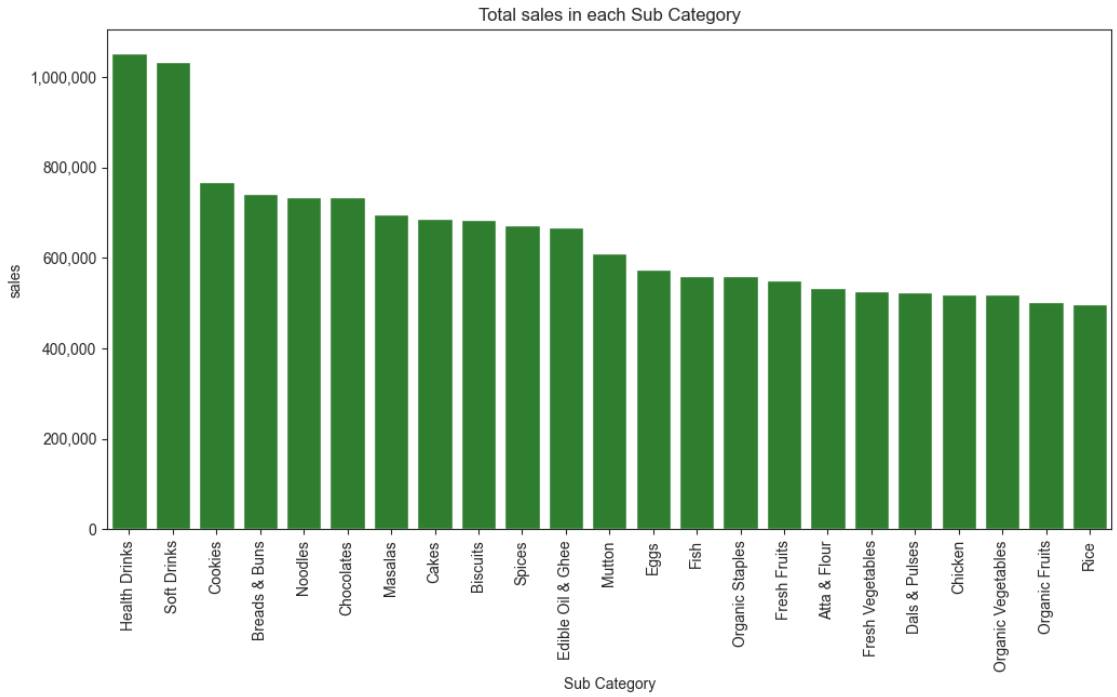
[76]:
```
sales_by_sub_category = df.groupby(["Sub Category"])["Sales"].sum().
 ↪sort_values(ascending=False)
```

[77]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_sub_category.index, y=sales_by_sub_category.values,
            color="#228B22", order=df.groupby(["Sub Category"])["Sales"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.tick_params(axis='x', rotation=90)
ax.set_title("Total sales in each Sub Category")

plt.show()
```



Total sales in each Region

Insights

• West region has the highest number of total sales in the entire city. • While, South region has the lowest number of total sales in the entire city.

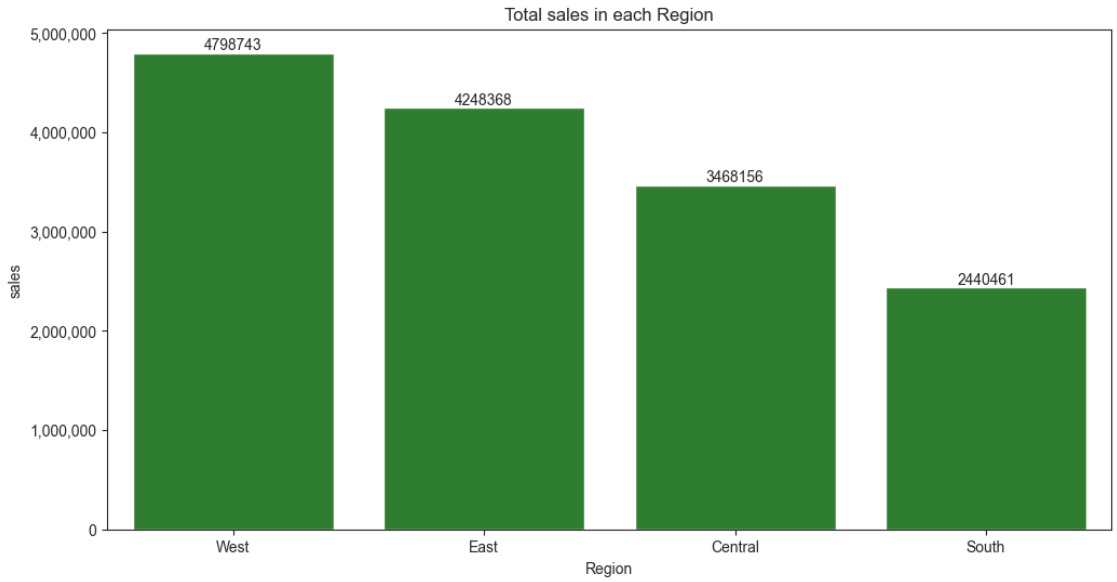
[78]:
```
sales_by_region = df.groupby(["Region"])["Sales"].sum().
 ↪sort_values(ascending=False)
```

[79]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_region.index, y=sales_by_region.values,
            color="#228B22", order=df.groupby(["Region"])["Sales"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total sales in each Region")

plt.show()
```



Total sales in each City

Insights

• Kanyakumari is the city with the highest number of total sales in Tamil Nadu. • While, Trichy is the city with the lowest number of total sales in Tamil Nadu.

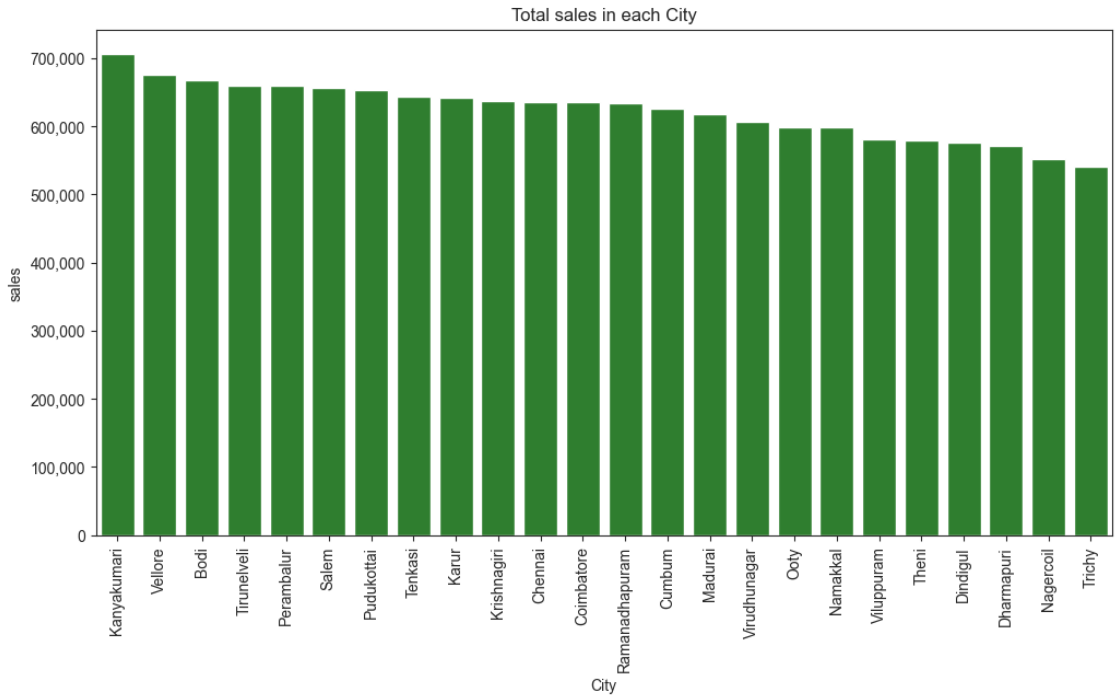
[80]:
```
sales_by_city = df.groupby(["City"])["Sales"].sum().sort_values(ascending=False)
```

[81]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_city.index, y=sales_by_city.values,
            color="#228B22", order=df.groupby(["City"])["Sales"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.tick_params(axis='x', rotation=90)
ax.set_title("Total sales in each City")

plt.show()
```



Total sales in each Month

Insights

• November is the month with the highest number of total sales in each Year. • While, February is the month with the lowest number of total sales in each Year.

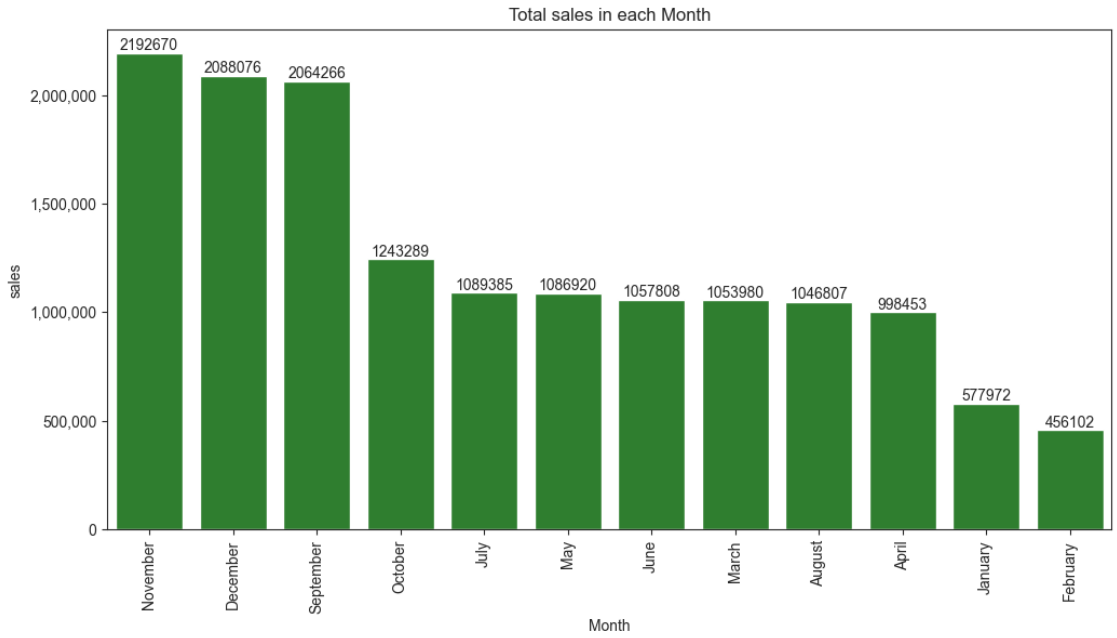
[82]:
```
sales_by_month = df.groupby(["Month"])["Sales"].sum().
↪sort_values(ascending=False)
```

[83]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_month.index, y=sales_by_month.values,
            color="#228B22", order=df.groupby(["Month"])["Sales"].sum().
↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.tick_params(axis='x', rotation=90)
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total sales in each Month")

plt.show()
```



Total sales in each Year

Insights

• 2018 was the year with the highest number of total sales. • While, 2015 being the year with the lowest number of total sales.

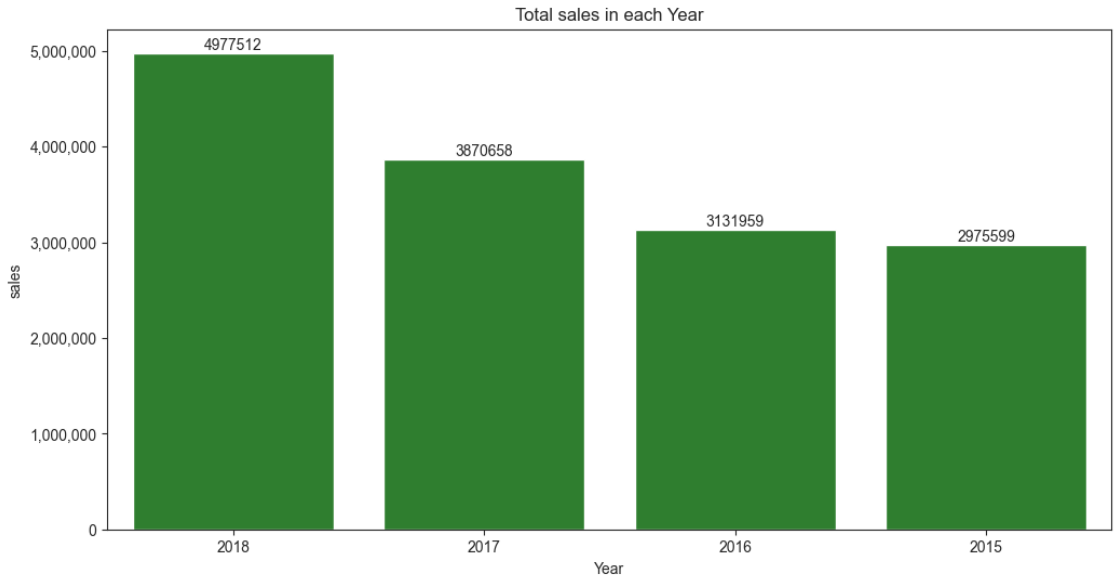
[84]:
```
sales_by_year = df.groupby(["Year"])["Sales"].sum().sort_values(ascending=False)
```

[85]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=sales_by_year.index, y=sales_by_year.values,
            color="#228B22", order=df.groupby(["Year"])["Sales"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total sales in each Year")

plt.show()
```



Total profit in each Category

Insights

• Snacks was the most profitable category of product in the Supermarket. • While, Oil & Masala being the least profitable category of product in the Supermarket.

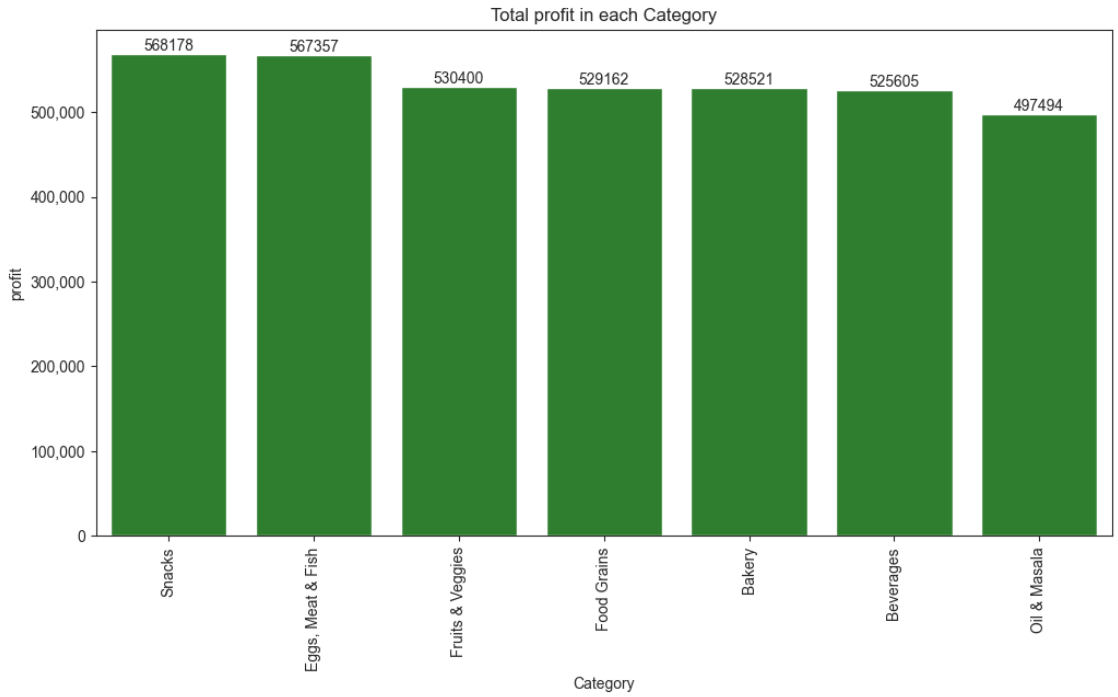
[86]:
```
profit_by_category = df.groupby(["Category"])["Profit"].sum().
↪sort_values(ascending=False)
```

[87]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_category.index, y=profit_by_category.values,
            color="#228B22", order=df.groupby(["Category"])["Profit"].sum().
↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.tick_params(axis='x', rotation=90)
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total profit in each Category")

plt.show()
```



Total profit in each Sub Category

Insights

- Health Drinks & Soft Drinks are the most profitable sub category of products in the Supermarket.
- While, Chicken being the least profitable sub category of product in the Supermarket.

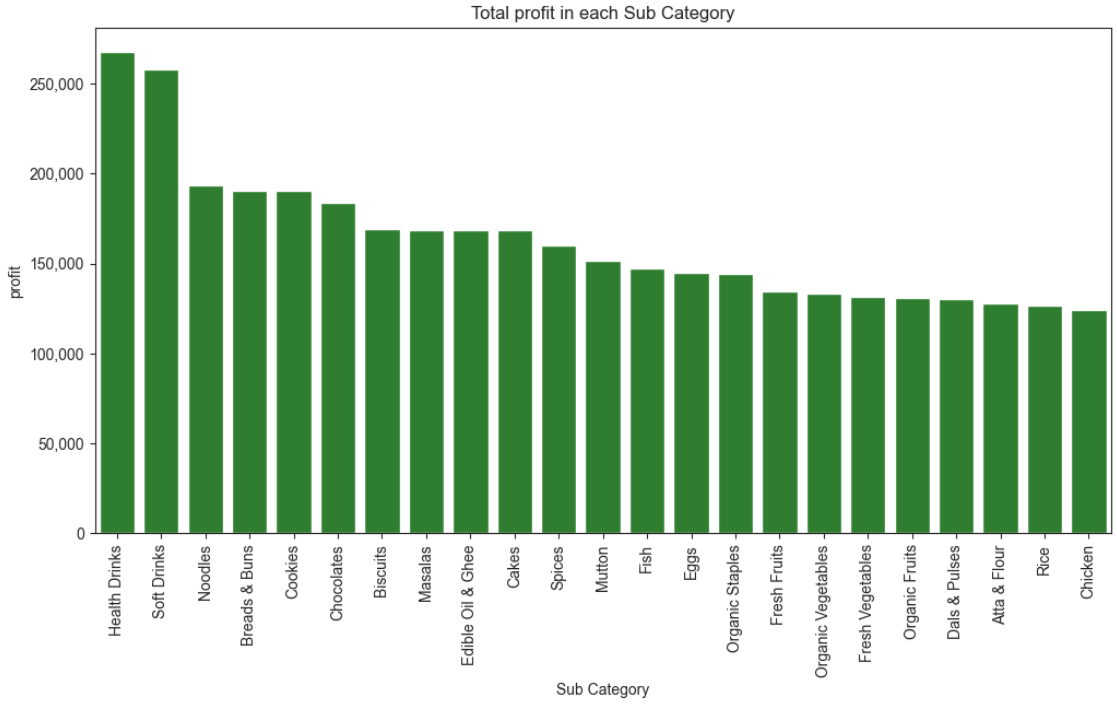
[88]:
```
profit_by_sub_category = df.groupby(["Sub Category"])["Profit"].sum().
↪sort_values(ascending=False)
```

[89]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_sub_category.index, y=profit_by_sub_category.values,
            color="#228B22", order=df.groupby(["Sub Category"])["Profit"].sum().
↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.tick_params(axis='x', rotation=90)
ax.set_title("Total profit in each Sub Category")

plt.show()
```



Total profit in each Region

Insights

• West was the most profitable region of the Supermarket. • While, South being the least profitable region of the Supermarket.

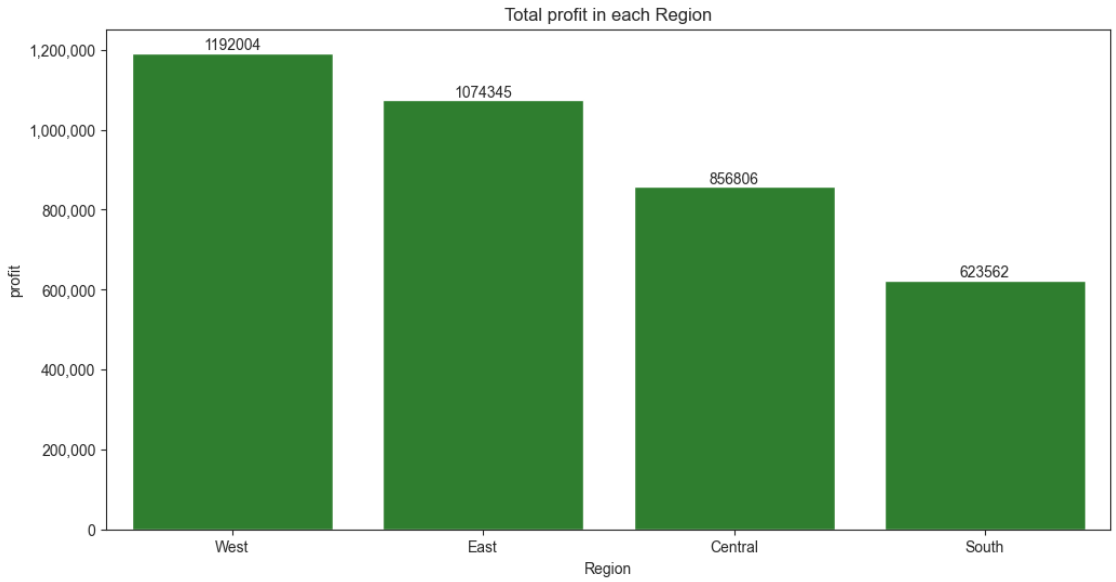
[90]:
```
profit_by_region = df.groupby(["Region"])["Profit"].sum().
 ↪sort_values(ascending=False)
```

[91]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_region.index, y=profit_by_region.values,
            color="#228B22", order=df.groupby(["Region"])["Profit"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total profit in each Region")

plt.show()
```



Total profit in each City

Insights

• Vellore was the most profitable city in Tamil Nadu. • While, Trichy being the least profitable city in Tamil Nadu.

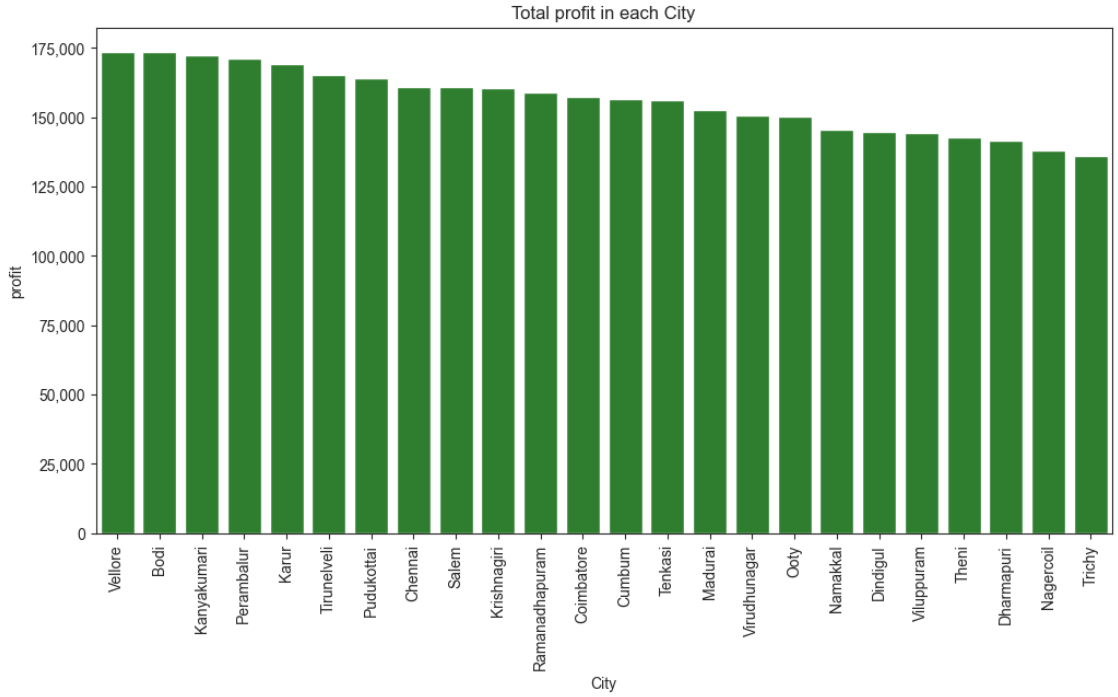
[92]:
```
profit_by_city = df.groupby(["City"])["Profit"].sum().
 ↪sort_values(ascending=False)
```

[93]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_city.index, y=profit_by_city.values,
            color="#228B22", order=df.groupby(["City"])["Profit"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.tick_params(axis='x', rotation=90)
ax.set_title("Total profit in each City")

plt.show()
```



Total profit in each Month

Insights

• November was the most profitable month for the Supermarket. • While, February being the least profitable month for the Supermarket.

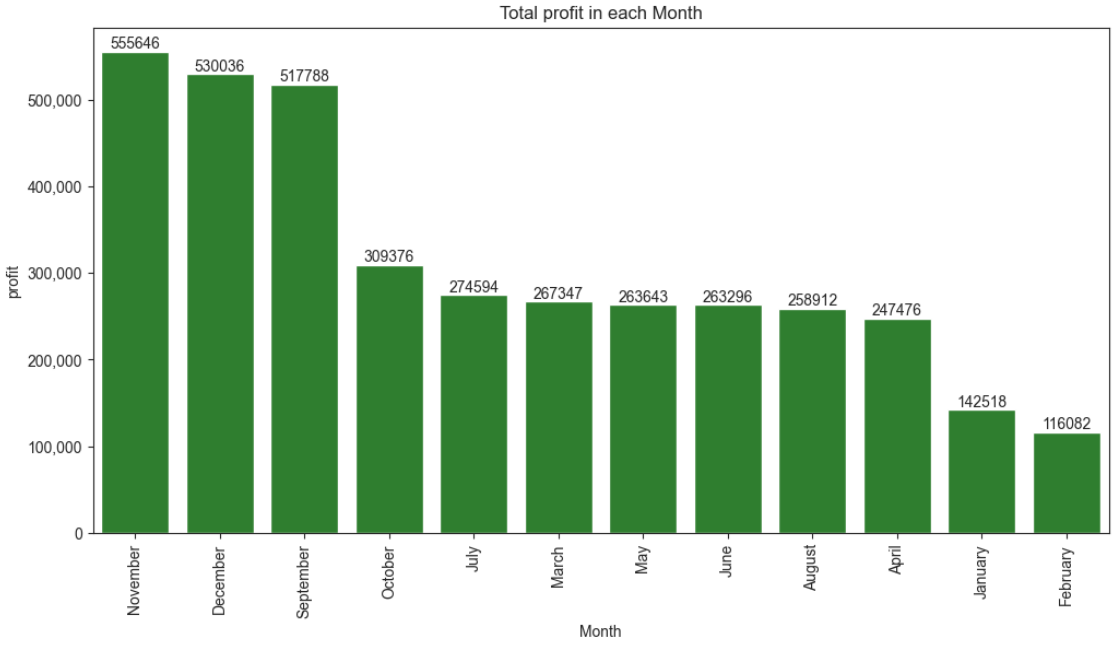
[94]:
```
profit_by_month = df.groupby(["Month"])["Profit"].sum().
↪sort_values(ascending=False)
```

[95]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_month.index, y=profit_by_month.values,
            color="#228B22", order=df.groupby(["Month"])["Profit"].sum().
↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.tick_params(axis='x', rotation=90)
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total profit in each Month")

plt.show()
```



Total profit in each Year

Insights

• 2018 was the most profitable year for the Supermarket. • While, 2015 being the least profitable year for the Supermarket.

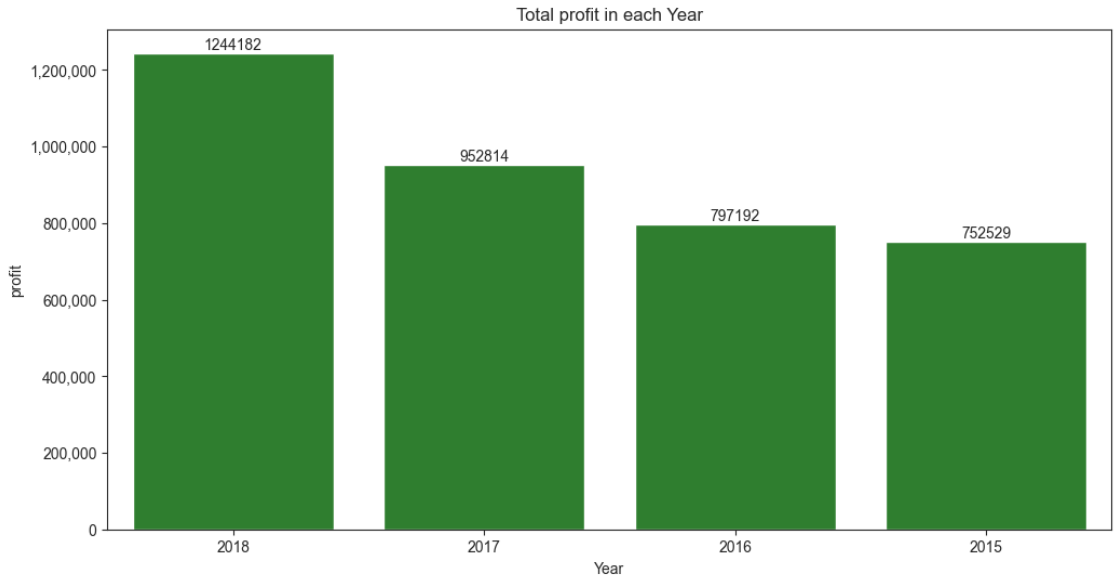
[96]:
```
profit_by_year = df.groupby(["Year"])["Profit"].sum().
 ↪sort_values(ascending=False)
```

[97]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_by_year.index, y=profit_by_year.values,
            color="#228B22", order=df.groupby(["Year"])["Profit"].sum().
 ↪sort_values(ascending=False).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total profit in each Year")

plt.show()
```



Customers with Highest Amount of Total Sales

Insights

• Krithika has purchased most amount of items from the Supermarket among all Customers. • Amrish has purchased second largest amount of items from the Supermarket.

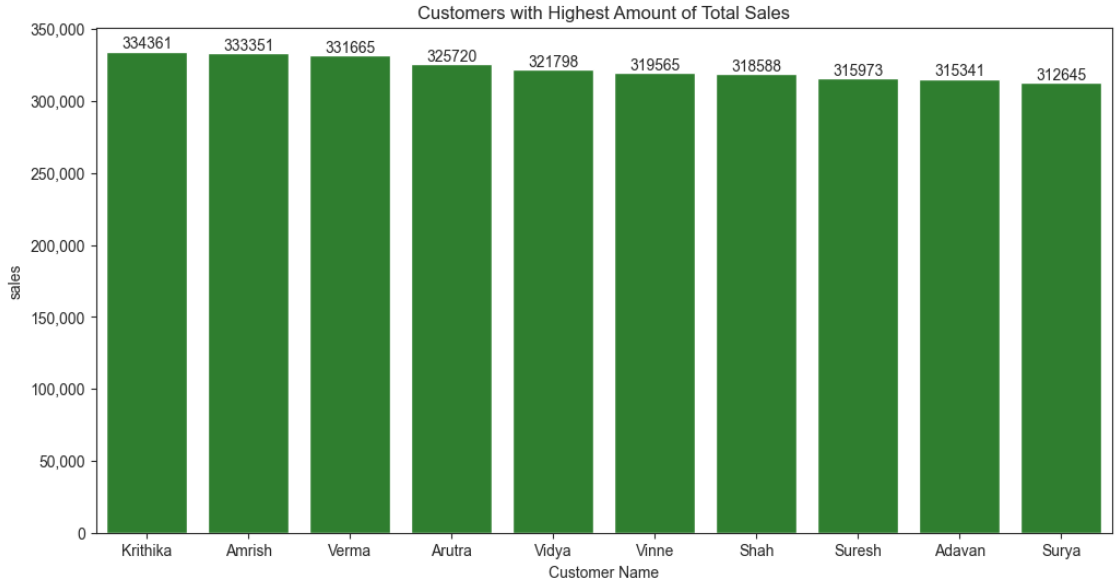
[98]:
```
highest_total_sales_by_customers = df.groupby(["Customer Name"])["Sales"].sum().
↪sort_values(ascending=False).head(10)
```

[99]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=highest_total_sales_by_customers.index,␣
↪y=highest_total_sales_by_customers.values,
            color="#228B22", order=df.groupby(["Customer Name"])["Sales"].sum().
↪sort_values(ascending=False).head(10).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('sales')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Customers with Highest Amount of Total Sales")

plt.show()
```



Customers with Highest Profit on their Purchase

Insights

• Arutra has the most profitable purchase from the Supermarket saving a lot of money. • Vidya has the second highest profitable purchase from the Supermarket.

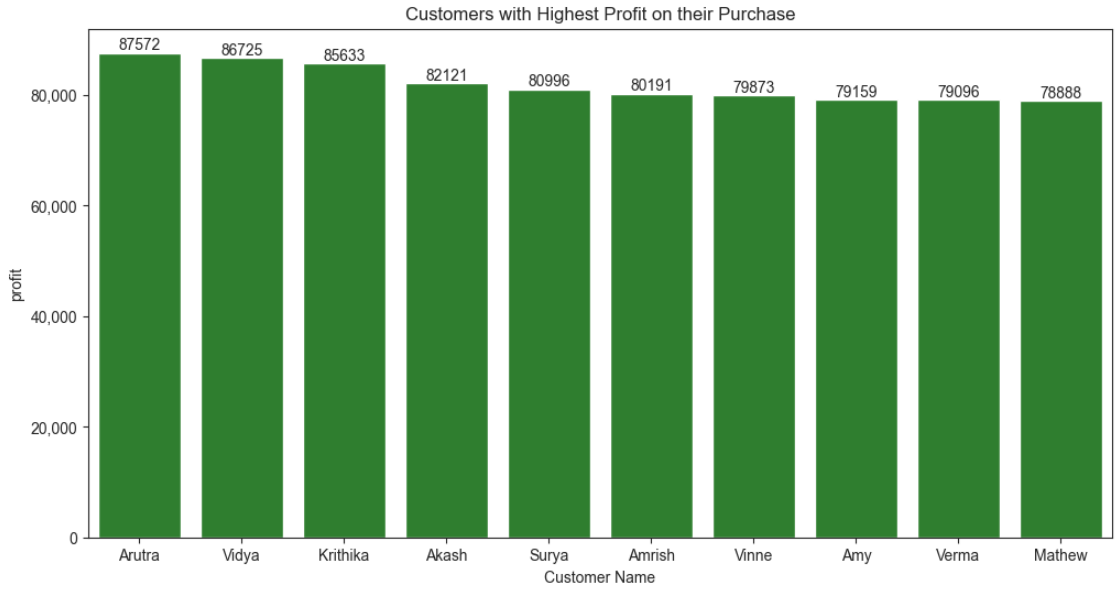
[100]:
```
profit_on_purchase_by_customers = df.groupby(["Customer Name"])["Profit"].sum().
↪sort_values(ascending=False).head(10)
```

[101]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=profit_on_purchase_by_customers.index,␣
↪y=profit_on_purchase_by_customers.values,
            color="#228B22", order=df.groupby(["Customer Name"])["Profit"].
↪sum().sort_values(ascending=False).head(10).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('profit')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Customers with Highest Profit on their Purchase")

plt.show()
```



Total Discount availed by Customers

Insights

• Verma got the highest total discount on his purchase from the Supermarket. • Amrish got the second highest total discount on his purchase from the Supermarket.

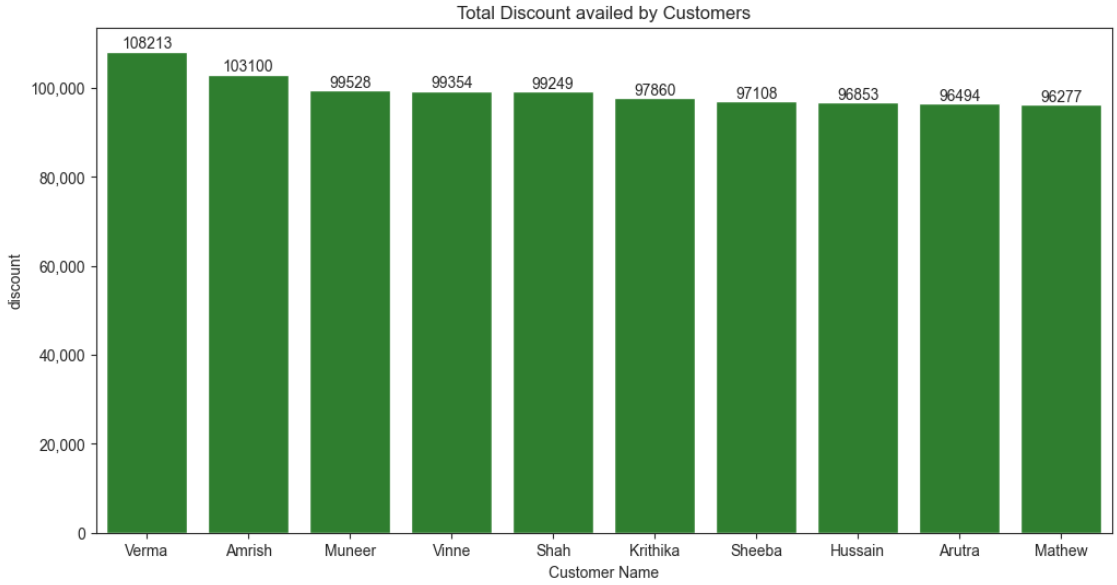
[102]:
```
discount_availed_by_customers = df.groupby(["Customer Name"])["Discount_Amt"].
↪sum().sort_values(ascending=False).head(10)
```

[103]:
```
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=discount_availed_by_customers.index,␣
↪y=discount_availed_by_customers.values,
            color="#228B22", order=df.groupby(["Customer␣
↪Name"])["Discount_Amt"].sum().sort_values(ascending=False).head(10).index)

formatter = FuncFormatter(lambda x, _: f'{int(x):,}')
ax.yaxis.set_major_formatter(formatter)

ax.set_ylabel('discount')
ax.bar_label(ax.containers[0], fmt='%d')
ax.set_title("Total Discount availed by Customers")

plt.show()
```


Total Discount availed by Customers
108213
103100
99528
99354
99249
97860
97108
96853
96494
96277
100,000
80,000
60,000
40,000
20,000
0
discount
Verma
Amrish
Muneer
Vinne
Shah
Krithika
Sheeba
Hussain
Arutra
Mathew
Customer Name