IMPORT LIBRARIES AND LOAD THE DATASET

```
In [2]: import matplotlib.pyplot as plt import seaborn as sns import
        plotly.express as px import pandas as pd
        sns.set(style="whitegrid") df=pd.read_csv("/content/Sample -
        Superstore.csv",encoding="latin1")
```

DATA CLEANING

```
In [3]: #infromation and null values print("DATA
        INFORMATION..\n") df.info() print("\nMISSING
        VALUES:\n\n",df.isnull().sum())
```

```
DATA INFORMATION..

< class 'pandas.core.frame.DataFrame' >
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #    Column          Non-Null Count  Dtype
---   ------          --------------  -----
0    Row ID          9994 non-null   int64
1    Order ID        9994 non-null   object
2    Order Date      9994 non-null   object
3    Ship Date       9994 non-null   object
4    Ship Mode       9994 non-null   object
5    Customer ID     9994 non-null   object
6    Customer Name   9994 non-null   object
7    Segment         9994 non-null   object
8    Country         9994 non-null   object
9    City            9994 non-null   object
10   State           9994 non-null   object
11   Postal Code     9994 non-null   int64
12   Region          9994 non-null   object
13   Product ID      9994 non-null   object
14   Category        9994 non-null   object
15   Sub-Category    9994 non-null   object
16   Product Name    9994 non-null   object
17   Sales           9994 non-null   float64
18   Quantity        9994 non-null   int64
19   Discount        9994 non-null   float64
20   Profit          9994 non-null   float64
dtypes: float64(3), int64(3), object(15) memory
 usage: 1.6+ MB
```

```
MISSING VALUES:

Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

```
In [5]: df=df.drop_duplicates() #removing duplicates
        df.info() print("\n\nTHERE IS NO DUPLICATES IN THE
        DATASET!!")
       < class 'pandas.core.frame.DataFrame' >
       RangeIndex: 9994 entries, 0 to 9993
       Data columns (total 21 columns):
        #   Column          Non-Null Count  Dtype
       ---  ------          --------------  -----
       0    Row ID          9994 non-null   int64
       1    Order ID        9994 non-null   object
       2    Order Date      9994 non-null   object
       3    Ship Date       9994 non-null   object
       4    Ship Mode       9994 non-null   object
       5    Customer ID     9994 non-null   object
       6    Customer Name   9994 non-null   object
       7    Segment         9994 non-null   object
       8    Country         9994 non-null   object
       9    City            9994 non-null   object
       10   State           9994 non-null   object
       11   Postal Code     9994 non-null   int64
       12   Region          9994 non-null   object
       13   Product ID      9994 non-null   object
       14   Category        9994 non-null   object
       15   Sub-Category    9994 non-null   object
       16   Product Name    9994 non-null   object
       17   Sales           9994 non-null   float64
       18   Quantity        9994 non-null   int64
       19   Discount        9994 non-null   float64  20  Profit          9994 non-
            null   float64 dtypes: float64(3), int64(3), object(15) memory usage:
            1.6+ MB
```

THERE IS NO DUPLICATES IN THE DATASET!!

In [6]:
```
#converting date columns into datetime datatype df['Order
Date']=pd.to_datetime(df['Order Date'])#order date
df['Ship Date']=pd.to_datetime(df['Ship Date'])#ship date
df.info()
```

```
< class 'pandas.core.frame.DataFrame' >
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
0    Row ID         9994 non-null   int64
1    Order ID       9994 non-null   object
2    Order Date     9994 non-null   datetime64[ns]
3    Ship Date      9994 non-null   datetime64[ns]
4    Ship Mode      9994 non-null   object
5    Customer ID    9994 non-null   object
6    Customer Name  9994 non-null   object
7    Segment        9994 non-null   object
8    Country        9994 non-null   object
9    City           9994 non-null   object
10   State          9994 non-null   object
11   Postal Code    9994 non-null   int64
12   Region         9994 non-null   object
13   Product ID     9994 non-null   object
14   Category       9994 non-null   object
15   Sub-Category   9994 non-null   object
16   Product Name   9994 non-null   object
17   Sales          9994 non-null   float64
18   Quantity       9994 non-null   int64
19   Discount       9994 non-null   float64       20   Profit          9994
     non-null    float64       dtypes: datetime64[ns](2), float64(3),
     int64(3), object(13) memory usage: 1.6+ MB
```

SUMMARY OF THE DATA

In [7]:
```
print("THE SUMMARY OF THE DATASET IS AS FOLLOWS:\n")
df.describe()
```

THE SUMMARY OF THE DATASET IS AS FOLLOWS:

Out[7]:

	Row ID	Order Date	Ship Date	Postal Code	Sales	
count	9994.000000	9994	9994	9994.000000	9994.000000	9
mean	4997.500000	2016-04-30 00:07:12.259355648	2016-05-03 23:06:58.571142912	55190.379428	229.858001	
min	1.000000	2014-01-03 00:00:00	2014-01-07 00:00:00	1040.000000	0.444000	
25%	2499.250000	2015-05-23 00:00:00	2015-05-27 00:00:00	23223.000000	17.280000	
50%	4997.500000	2016-06-26 00:00:00	2016-06-29 00:00:00	56430.500000	54.490000	
75%	7495.750000	2017-05-14 00:00:00	2017-05-18 00:00:00	90008.000000	209.940000	

max	9994.000000	2017-12-30 00:00:00	2018-01-05 00:00:00	99301.000000	22638.480000
std	2885.163629	NaN	NaN	32063.693350	623.245101

EXPLORATORY DATA ANALYSIS

In [8]:
```
print("TOTAL SALES:",df['Sales'].sum().round())

print("TOTAL PROFITS:",df['Profit'].sum().round())

print("TOTAL UNITS SOLD:",df['Quantity'].sum())
```

```
TOTAL SALES: 2297201.0
TOTAL PROFITS: 286397.0
TOTAL UNITS SOLD: 37873
```

In [9]:
```
#sales by region
region_sales=df.groupby('Region')['Sales'].sum().sort_values(ascending=F
al print("THE TOTAL SALES BY REGION:\n") print(region_sales)

#sales by catagories
catagory_sales=df.groupby('Category')['Sales'].sum().round().reset_index
() print("\n\nTHE TOTAL SALES BY CATAGORIES:\n") print(catagory_sales)

#sales over a time(based on year)
time_sales=df.groupby(df['Order
Date'].dt.year)['Sales'].sum().round().res print("\n\nTHE TOTAL SALES
OVER THE YEARS:\n") print(time_sales)

#profit by sub_catagories
sub_catagory=df.groupby('Sub-
Category')['Profit'].sum().round().reset_inde print('THE TOTAL PROFIT
BASED ON SUB_CATAGORY\n:') print(sub_catagory)
```

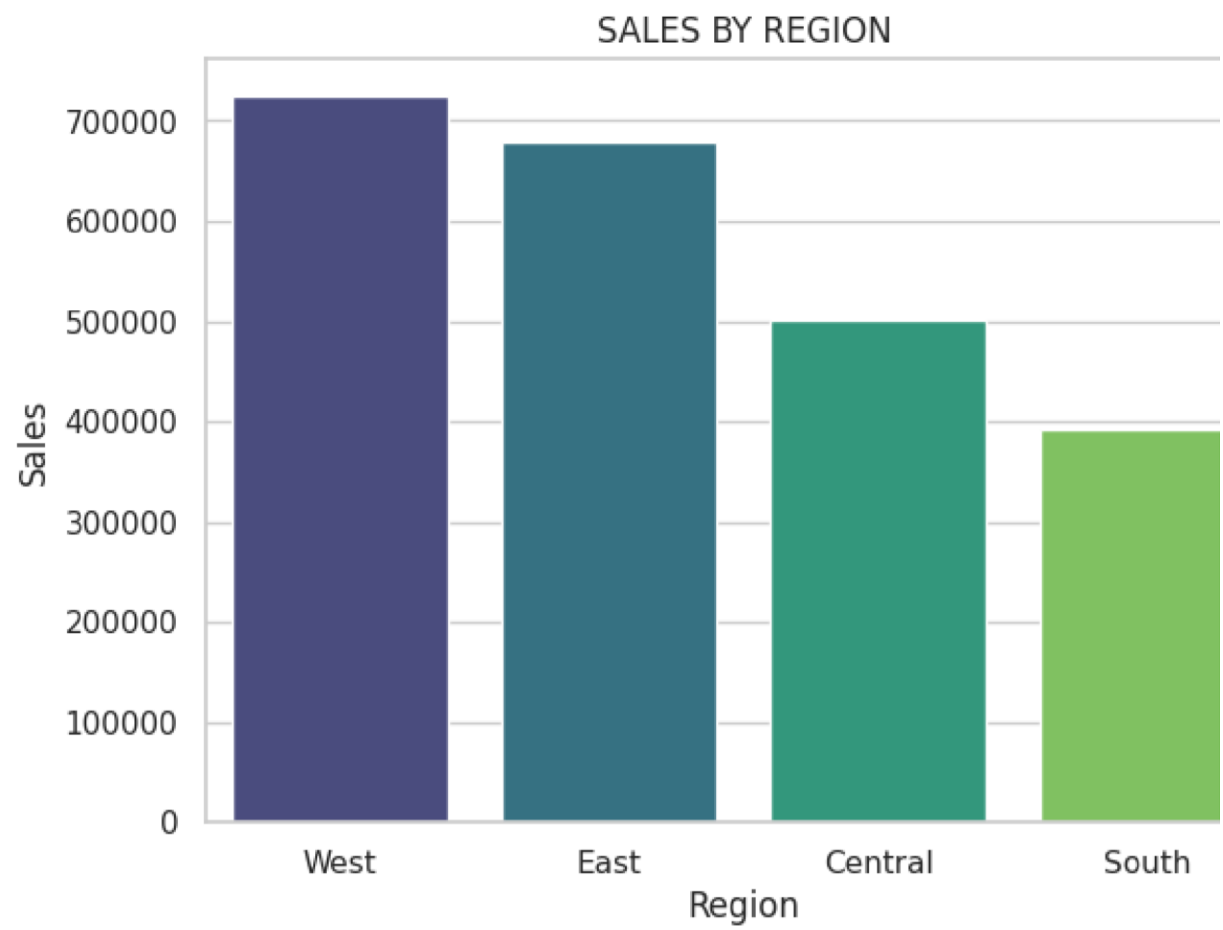
```
THE TOTAL SALES BY REGION:

    Region     Sales
0     West   725458.0
1     East   678781.0
2     Central   501240.0
3     South   391722.0


THE TOTAL SALES BY CATAGORIES:

          Category     Sales
0        Furniture   742000.0
1        Office Supplies   719047.0
2        Technology   836154.0


THE TOTAL SALES OVER THE YEARS:

   Order Date     Sales
0        2014  484247.0
1        2015  470533.0
2        2016  609206.0
3        2017  733215.0
THE TOTAL PROFIT BASED ON SUB_CATAGORY
:
   Sub-Category   Profit
0           Accessories   41937.0
1           Appliances   18138.0
2           Art    6528.0
3           Binders   30222.0
4           Bookcases   -3473.0
5           Chairs   26590.0
6           Copiers   55618.0
7           Envelopes    6964.0
8           Fasteners     950.0
9           Furnishings   13059.0
10          Labels    5546.0
11          Machines    3385.0
12          Paper   34054.0
13          Phones   44516.0
14          Storage   21279.0
15          Supplies   -1189.0
16          Tables  -17725.0
```

VIZUALIZATION

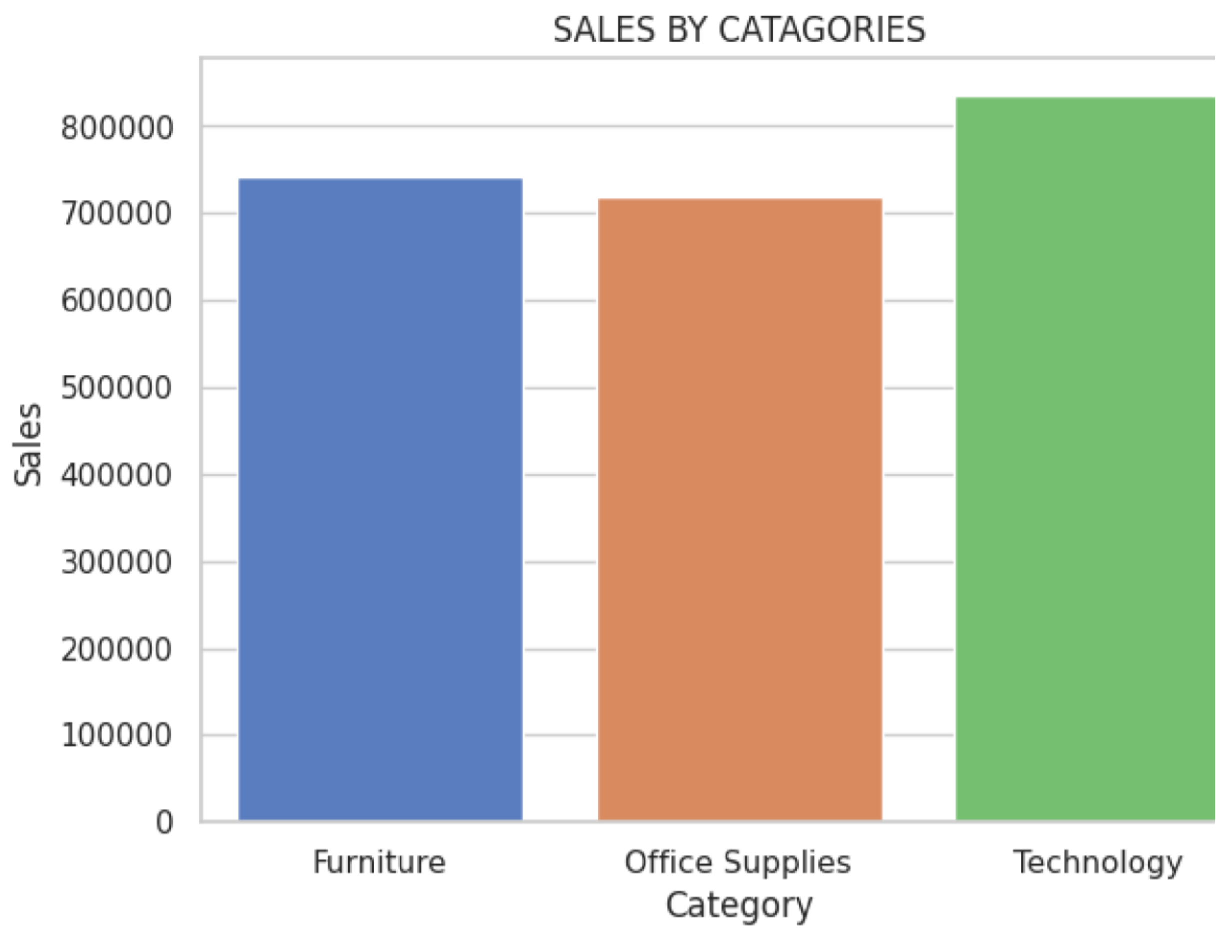
1. sales by region- bar chart
2. sales by catagories- bar chart
3. sales over a time - line plot
4. profit by sub_catagories - bar chart(using plotly)
5. correlation - heatmap

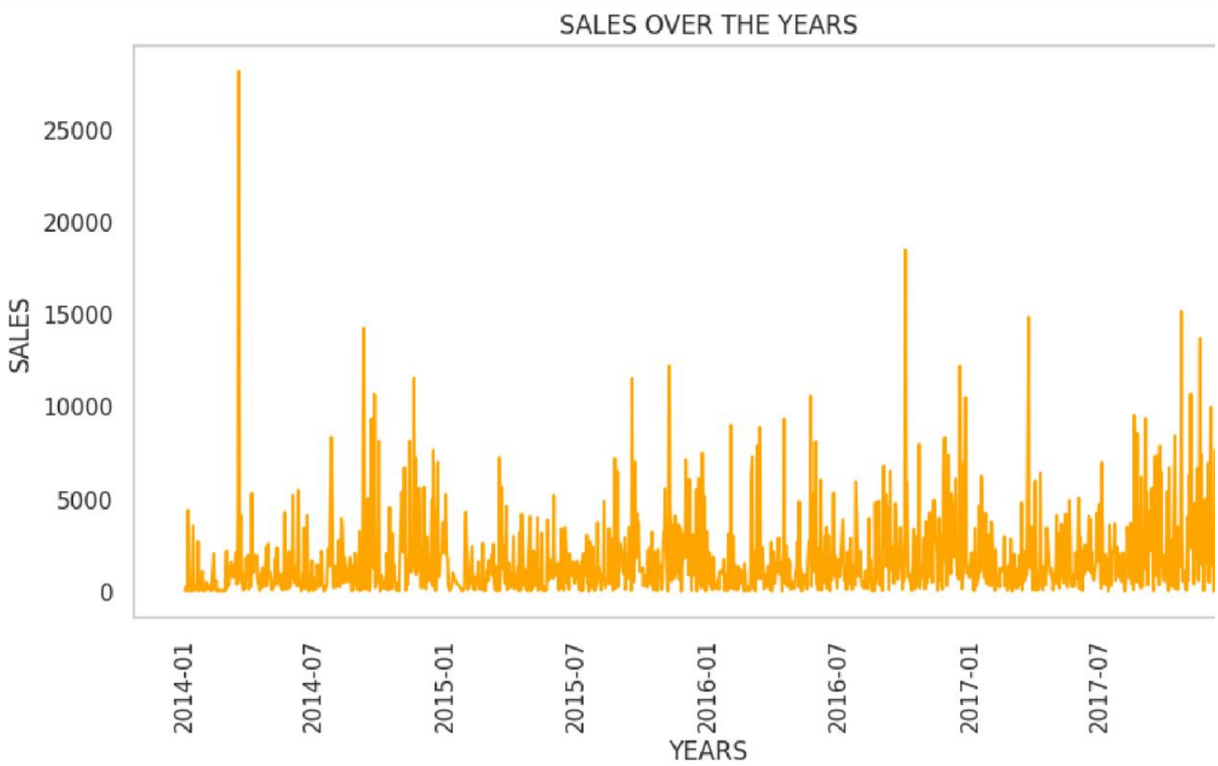
6. profit vs discount - scatter plot
7. order placed distribution - histogram

```
In [10]: #sales by region
         region_sales=df.groupby('Region')['Sales'].sum().sort_values(ascending=F
         al plt.figure(figsize=(7,5))
         sns.barplot(data=region_sales,x='Region',y='Sales',palette='viridis',hue
         =' plt.title('SALES BY REGION') plt.show()
```




```
In [12]: #sales by catagories
         catagory_sales=df.groupby('Category')['Sales'].sum().round().reset_index
         () plt.figure(figsize=(7,5))
         sns.barplot(data=catagory_sales,x='Category',y='Sales',palette="muted",h
         ue plt.title('SALES BY CATAGORIES') plt.show()
In [11]: #sales over a time time_sales=df.groupby(df['Order
         Date'])['Sales'].sum().round().reset_index plt.figure(figsize=(10,5))
         plt.plot(time_sales['Order Date'],time_sales['Sales'],c="orange")
```






```
plt.grid(False) plt.title('SALES OVER THE YEARS') plt.xlabel('YEARS')
plt.xticks(rotation=90) plt.ylabel('SALES') plt.show()
```

In [13]:
```
#profit by sub_catagories sub_catagory=df.groupby('Sub-
Category')['Profit'].sum().round().reset_inde
```

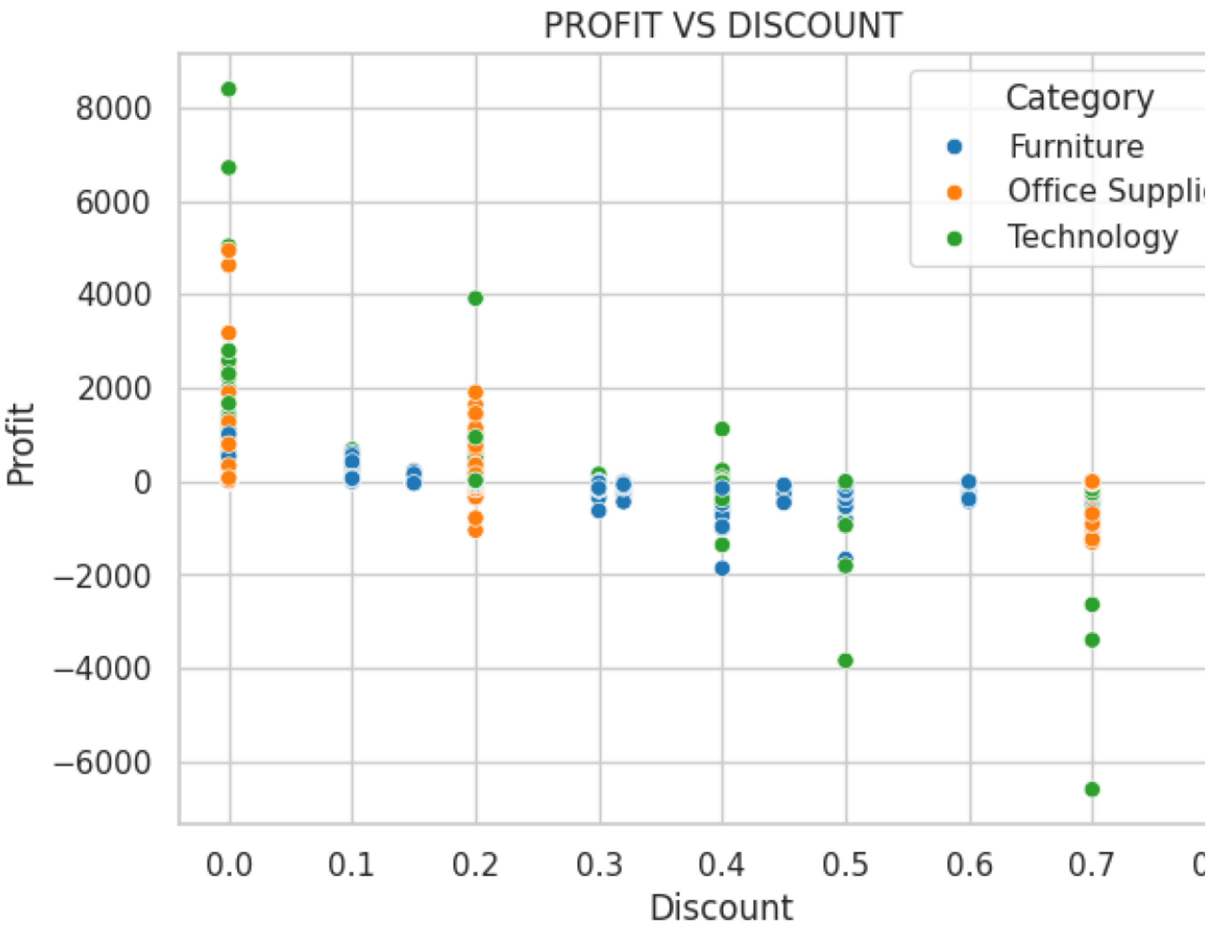
```
fig=px.bar(sub_catagory,x='Sub-
Category',y='Profit',color='Profit',title='
fig.update_layout(xaxis_tickangle=-45) fig.show()
```

In [14]: #correlation plt.figure(figsize=(7,5))

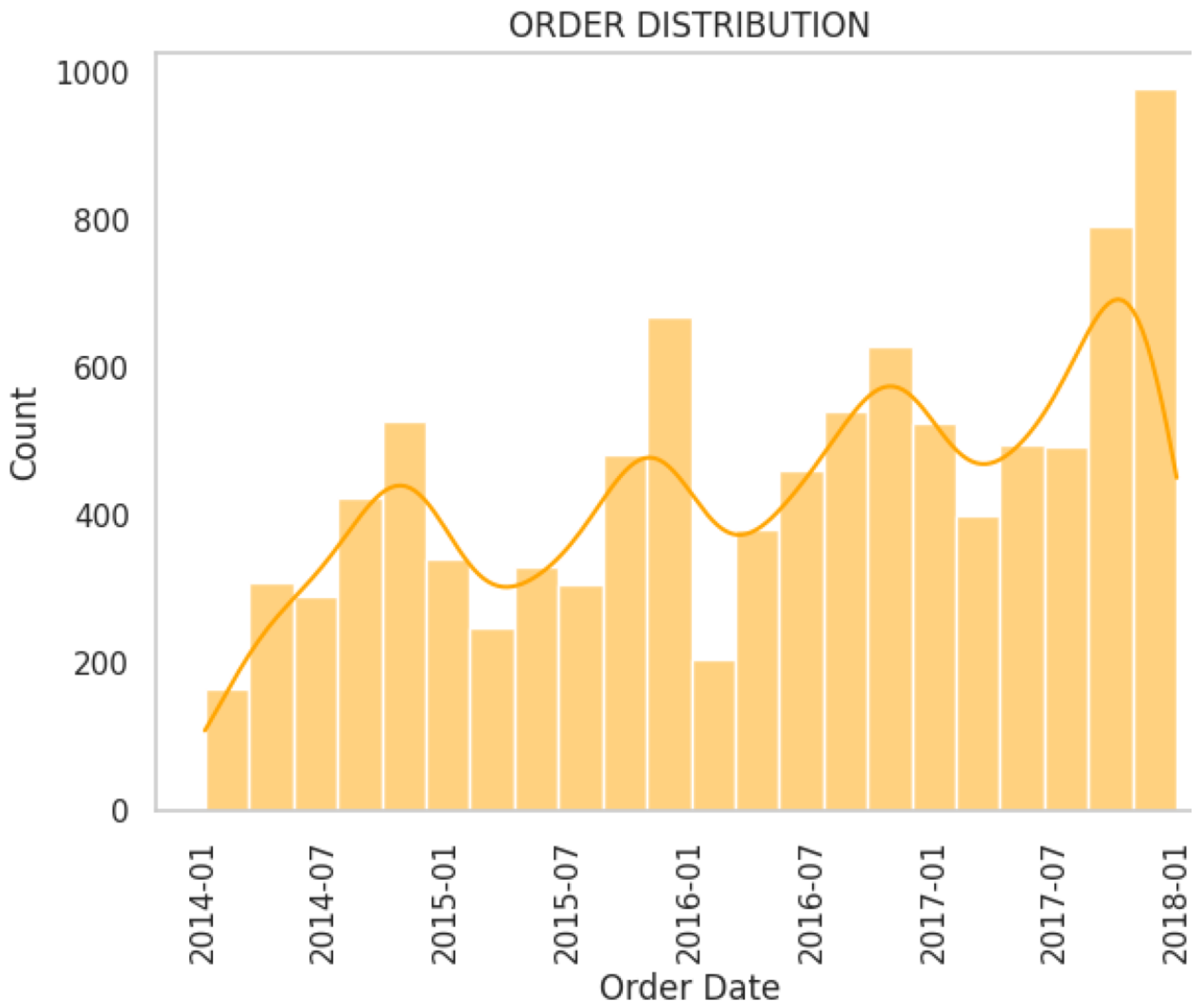
```
sns.heatmap(df[['Sales','Quantity','Discount','Profit']].corr(),annot=Tr
ue plt.title('CORRELATION MATRIX') plt.show()
```

In [15]:
```
#profit vs discount plt.figure(figsize=(7,5))
sns.scatterplot(data=df,x='Discount',y='Profit',hue='Category',palette='
ta plt.title('PROFIT VS DISCOUNT') plt.show()
```

In [20]:
```
#order distribution plt.figure(figsize=(7,5))
sns.histplot(data=df,x='Order
Date',kde=True,color='orange') plt.grid(False)
```




```
plt.title('ORDER DISTRIBUTION') plt.xticks(rotation=90)
plt.show()
```



CONCLUSION:

In this data analytics project, we conducted an in-depth analysis using **seven key visualizations**, each designed to explore different dimensions of the business dataset. Here's what each visualization revealed:

1. **Sales by Region (Bar Chart)**:
   This visualization highlighted regional performance differences. It was observed that certain regions significantly outperformed others in total sales, helping identify high-value areas and regions needing attention.
2. **Sales by Categories (Bar Chart)**:
   A breakdown of sales across product categories showed which product lines contributed most to revenue. This aids in inventory planning and marketing focus.
3. **Sales Over Time (Line Plot)**:
   The time-series plot revealed trends and seasonality in sales. Peak sales periods were identified, offering valuable insight for promotional planning and demand forecasting.
4. **Profit by Sub-Categories (Bar Chart using Plotly)**:
   A deeper dive into sub-categories showed that some had high sales but lower profits, while others were consistently profitable. This helps in optimizing product strategy.
5. **Correlation Matrix (Heatmap)**:
   The heatmap provided a visual overview of relationships between numerical variables. It showed strong correlations (positive and negative), helping identify variables that move together or inversely.

6. **Profit vs. Discount (Scatter Plot)**:
   This plot revealed that higher discounts often led to lower profits, indicating diminishing returns on discount strategies. It emphasized the need to balance discounts with profit margins.
7. **Order Placed Distribution (Histogram)**:
   The histogram showed how orders were distributed in terms of frequency. It identified whether the order activity was concentrated around specific values or spread out, helping understand customer behavior patterns.

🔍 Key Takeaways:

- High-performing regions and categories can be prioritized for growth.
- Discount strategies should be evaluated to avoid profit loss.
- Time-based and sub-category trends offer insights for future planning.
- Correlation insights can guide feature selection and deeper statistical modeling.

✅ Final Thought:

These seven visualizations collectively provided a clear and actionable understanding of the dataset. They transformed raw data into business intelligence, enabling more informed and strategic decision-making.