KINGSTON
ENGINEERING COLLEGE

College code:5113

E. Chanikya-(511321104013) – Team Head

chanikyaeddula@gmail.com

E. Manoj-(511321104053)

manojerragopula@gmail.com

R.Bharath kumar-(511321104010)

bk7428891@gmail.com

V.Mohith Manindranath-(511321104055)
v.mohitmanindranath162003@gmail.com

Data Analytics with Cognos

PHASE 5

PROJECT:

Product Sales Analysis

IBM Cognos logo:

Visualisation:

OBJECTIVES:

Analyzing product sales data is crucial for businesses to make informed decisions, optimize strategies, and drive growth. Here is a structured approach with steps for conducting a product sales analysis project:

Step 1: Define Objectives and Scope

Clearly define the objectives of your product sales analysis. Identify what specific aspects of sales you want to analyze (e.g., overall sales performance, product-specific performance, market segmentation, or sales forecasting).

Step 2: Data Collection and Preparation

Collect relevant sales data, which may include transaction records, product details, customer information, and market data. Ensure data quality by cleaning and preprocessing the data:

- Handle missing data by imputing values or removing incomplete records.
- Standardize and clean product and customer names for consistency.
- Convert data types and formats as needed.

Step 3: Data Exploration and Descriptive Analysis

Conduct exploratory data analysis (EDA) to understand the dataset's characteristics:

- Calculate basic statistics like mean, median, and standard deviation.
- Create visualizations (e.g., histograms, bar charts, scatter plots) to identify trends and patterns in sales data.
- Segment data by time (e.g., monthly, quarterly, annually) to analyze seasonality and trends.

Step 4: Product Performance Analysis

Analyze product-level performance to identify top-selling products, slow-moving items, and underperforming products:

- Calculate metrics like total revenue, quantity sold, profit margins, and growth rates for each product.
- Identify product categories or SKUs that contribute significantly to overall sales.

Step 5: Customer Segmentation

Segment customers based on various criteria such as demographics, purchase behavior, or customer lifetime value:

- Identify high-value customers, returning customers, and potential target segments.
- Analyze customer purchase patterns and preferences.

Step 6: Market Analysis

Analyze market data to understand external factors that influence sales:

- Study market trends, economic indicators, and competitor performance.
- Assess the impact of marketing campaigns, promotions, and pricing strategies.

Step 7: Time Series Analysis

Perform time series analysis to understand sales patterns over time:

- Apply forecasting models (e.g., moving averages, exponential smoothing, ARIMA) to predict future sales.
- Evaluate forecast accuracy using appropriate metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

Step 8: Sales Funnel Analysis (if applicable)

For businesses with multi-step sales processes (e.g., e-commerce sites), analyze the sales funnel:

- Monitor conversion rates at each stage of the funnel (e.g., website visits, product views, cart additions, checkout).
- Identify bottlenecks and areas for optimization.

Step 9: Root Cause Analysis

Investigate the factors contributing to fluctuations or changes in sales:

- Use statistical methods or hypothesis testing to identify the root causes of sales variations.
- Assess the impact of internal and external factors (e.g., product launches, economic downturns) on sales.

Step 10: Visualization and Reporting

Create visual reports and dashboards to communicate insights effectively to stakeholders:

- Use tools like Tableau, Power BI, or Python libraries (e.g., Matplotlib, Seaborn) to visualize data.
- Prepare a comprehensive report summarizing key findings, trends, and recommendations.

Step 11: Recommendations and Action Plan

Based on the analysis, provide actionable recommendations to improve sales performance:

- Propose pricing adjustments, marketing strategies, or product enhancements.

- Create a prioritized action plan with clear objectives and timelines.

Step 12: Implementation and Monitoring

Implement the recommended actions and closely monitor their impact on sales:

- Track sales performance after implementing changes.
- Adjust strategies as needed and continue monitoring over time.

Step 13: Documentation and Knowledge Sharing

Document the entire analysis process, methodologies, and results for future reference and knowledge sharing within the organization.

Dataset:

The dataset is comprised of hundreds of thousands of electronics store purchases broken down by product type, prices, order date, purchase address, etc., corresponding to the following coloumns:

Column	Description
Order ID	Unique IDs that are used to track orders.
Product	Names of the products.
Quantity Ordered	Total quantity ordered of a particular item.
Price Each	Prices of the products ordered.
Order Date	Dates and time at which a customer made an order.
Purchase Address	Addresses the orders were delivered to.

https://www.kaggle.com/datasets/beekiran/sales-data-analysis

Project:

The study will make use of finding best month for sales, how much earned on that month. This complete evaluation will provide a clear picture of the analysis of the sales based on order ID, Quality ordered, price each, order date, purchase address, Month, Sales, City, Hours which involves in the best sales month, forecasting profit, and the demand of the product.

Importing the required libraries:

In this step we are going to import the required python libraries and modules to work with our data and perform various data processing and machine learning tasks.

```
import pandas as pd
import pathlib
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter("ignore")
```

Loading the dataset:

This step involves loading our dataset into memory. We use libraries like pandas to read data from a CSV file or other formats.

```
all_data = pd.read_csv("/kaggle/input/sales-data-analysis/Sales Data.csv")
```

Preprocessing the data:

Preprocessing data in air quality analysis is a crucial step to ensure that the data is clean, reliable, and ready for in-depth analysis.

Data Cleaning:

Missing data handling:

Identify and address missing data points, which can result from sensor malfunctions or communication issues. Options include imputing missing values or removing affected data points if necessary.

Outlier Detection:

Detect and handle outliers, which can skew the analysis. Outliers may result from equipment malfunction or unusual events. You can choose to filter out extreme values or apply statistical techniques like Z-score analysis to identify them.

This step is vital for accurate comparisons and correlations between different datasets.

Data Transformation:

Feature Scaling:

Normalize or standardize numerical features to bring them to a similar scale. This is important for algorithms sensitive to feature scales.

Feature Encoding:

Convert categorical variables into a numerical format using techniques like one-hot encoding or label encoding.

Feature Engineering:

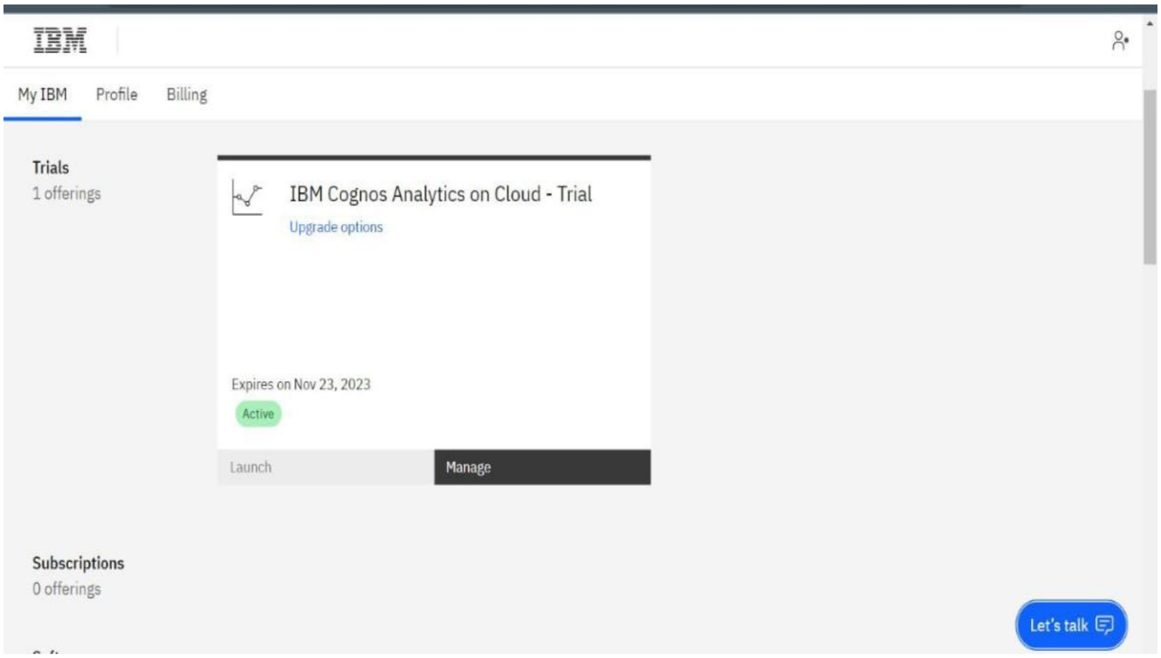
Create new features or modify existing ones to capture relevant information and patterns in the data.

Binning:

Group continuous data into bins or categories to simplify analysis.

Log Transformation:

Apply logarithmic transformations to features when necessary to make their distribution more normal.



Data Reduction:

Dimensionality Reduction:

Reduce the number of features, often using techniques like Principal Component Analysis (PCA) or feature selection to select the most relevant variables.

Outlier Detection and Handling:

Identify and deal with outliers, which can distort analysis and modeling results.

Data Integration:

Merge data from multiple sources or datasets to create a consolidated dataset for analysis.

Exploratory Data Analysis:

It focuses on Exploratory Data Analysis (EDA). It involves exploring and visualizing the data to gain insights. In this example, a simple time series plot is created using Matplotlib to visualize the sales.

Question 1: What was the best month for sales? How much was earned that month?

To answer this question, first, we need to extract only the months from the 'Order Date' coloumn and store each separately in a new coloumn, 'Months'. Second, we need to get the total sales amounts per order by multiplying the quantity ordered with the price of each individual product, and creating and storing the results in a 'Sales' coloumn. Finally, I will group the data by month, calculate the total sum of sales per month, and, lastly, visualize the data to get a better view of how sales changed from one month to the next.

```
Months_col = pd.to_datetime(df['Order Date']).dt.month_name().str[:3]

df.insert(loc=5, column='Months', value=Months_col)

Sales_col = df['Quantity Ordered'] * df['Price Each']

df.insert(loc=4, column='Sales', value=Sales_col)

sales_per_month = df.groupby(['Months']).sum()[['Quantity Ordered', 'Sales']]

sort_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

sales_per_month = sales_per_month.reindex(sort_order)

sales_per_month_USD = sales_per_month.copy()

sales_per_month_USD['Sales'] = sales_per_month_USD['Sales'].apply(lambda sale: '${:,.2f}'.format(sale))
```

```
print('The following table displays the total sales amount (and quantities ordered) for each month:')

sales_per_month_USDsales_per_month_sorted=sales_per_month.sort_values(by='Sales', ascending=False)

best_month = sales_per_month_sorted.index[0]

best_month = pd.to_datetime(best_month, format='%b').month_name()

print('The best month for sales was:', best_month)

maxsale = sales_per_month_sorted['Sales'].iloc[0]

print('The total sales amount earned that month was: ${:,.2f}'.format(maxsale))

months = sales_per_month.index.values

sales = sales_per_month['Sales']

plt.figure(figsize=(12,7))

plt.bar(months, sales,color='#407bbf', linewidth=1,edgecolor='k')

plt.title('Sales Amount Per Month', fontsize=1

plt.xlabel('Month', fontsize=13)

plt.ylabel('Sales Amount in USD ($)', fontsize=13)

plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)

plt.gcf().axes[0].yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('${x:,.0f}'))

plt.tight_layout()

plt.show()
```

Question 2: Which city sold the most products?

To compare cities, first we'll have to extract the city corresponding to each order from the 'Purchase Address' coloumn and store them in a separate coloumn 'City'. Thereafter we can group the data by city and calculate the total sum of sales for each city separately.

```
def get_city_state(address):
    address = address.split(', ')
    city = address[1]
    state = address[2][0:2]
    return "{} ({})".format(city, state)
city_col = df['Purchase Address'].apply(lambda add: get_city_state(add))
df.insert(loc=8, column='City', value=city_col)
sales_per_city = df.groupby(['City']).sum()['Sales']
sales_per_city_USD = sales_per_city.apply(lambda sale:
'${:,.2f}'.format(sale)).to_frame(name='Total Sales Amount')
print('The following table displays the total sales amount for each city:')
sales_per_city_USD
sales_per_city_sorted = sales_per_city.sort_values(ascending=False)
best_city = sales_per_city_sorted.index[0]
print('The city that sold the most products is:', best_city)
cities = sales_per_city.index.values
plt.figure(figsize=(10,7))
plt.bar(cities,sales_per_city,color='#44749d',width=0.6,linewidth=1,edgecolor=
'k')
plt.title('Sales Amount Per City', fontsize=15)
plt.xlabel('City', fontsize=13)
plt.ylabel('Sales Amount in USD ($)', fontsize=13)
plt.xticks(rotation=60)
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)
plt.gcf().axes[0].yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('${
x:,.0f}'))
plt.tight_layout()
```

```
plt.show()
```

Question 3: Which product sold the most? And why do you think it sold the most?

To answer this question, we would have to group the data based on product purchases and then calculate the total amount of quantities ordered for each product to determine which one sold the most amount of quantities.

```
products_sold = df.groupby(['Product']).sum()['Quantity Ordered']

products_sold = products_sold.sort_values(ascending=False)

most_sold_pro`duct = products_sold.index[0]

print('The product that was sold the most is:', most_sold_product)
```

Question 4: Is there a relationship between how much a product costs and the quantity sold?

One way to answer this question is to create a dual-axis line chart displaying the prices of each product and the quantity sold in order to compare them. First, we will have to create two groups, the first representing the prices of each product, the second representing the total quantity sold for each product.

```
products_quantity = df.groupby(['Product']).sum()['Quantity Ordered']

products = products_quantity.index.values

products_prices = df.groupby(['Product'])['Price Each'].apply(lambda price:
float(np.unique(price)))

fig, ax1 = plt.subplots(figsize=(12,7))

ax1.plot(products,products_quantity,marker='o',c='#407bbf',lw=2,

label='Quantities')

ax2 = ax1.twinx()

ax2.plot(products, products_prices,marker='o',c='#bf4040',lw=2,label='Prices')
```

```
ax1.set_title('The Relationship Between Product Price and Quantity Sold',fontsize=15)
ax1.set_xlabel('Product Name', fontsize=13)
ax1.set_ylabel('Total Quantity Sold', fontsize=12, color='#407bbf')
ax2.set_ylabel('Prices in USD ($)', fontsize=12, color='#cc3333')
ax1.set_xticklabels(products, rotation='vertical')
ax1.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax2.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('${x:,.0f}'))
ax1.legend(loc='upper left')
x2.legend(loc='upper right')
plt.grid()
plt.show()
```

Question 5: Which products are most often sold together?

For starters, we can filter data based on whether there are duplicates in the 'Order ID' coloumn, indicating that the same person made multiple product purchases, and then join the multiple products sold together and count the instances of particular products being sold together in order to extract those that most often ordered together.

```
order_filter = df['Order ID'].duplicated(keep=False)
df_multiple_orders = df[order_filter][['Order ID', 'Product']]
orders_per_person = df_multiple_orders.groupby(['Order ID'])['Product'].transform(lambda product: ", ".join(product))
df_orders_per_person = orders_per_person.to_frame(name='Products Sold Together').reset_index(drop=True)
orders_frequency = orders_per_person.value_counts()
df_orders_frequency = orders_frequency.to_frame(name='Frequency of products sold together')
df_orders_frequency
```

```
most_sold_together = orders_frequency.index[0]

print('The two products sold together the most often are: {}'.format(' and '.join(most_sold_together.split(', '))))
```

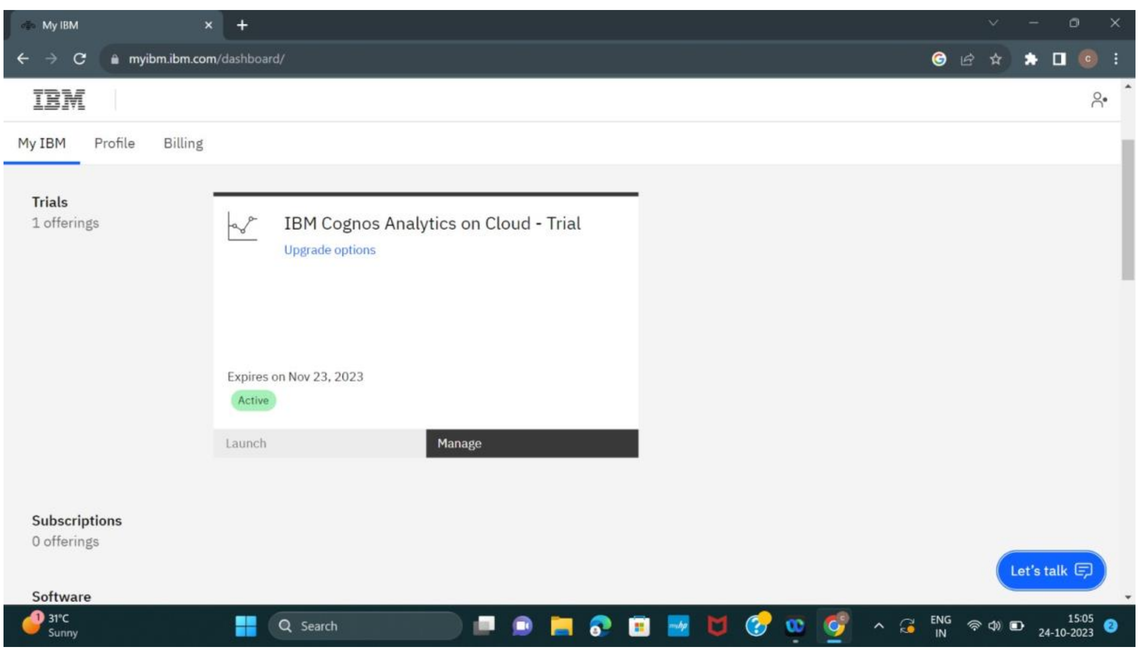
Question 6: Which time of the day should we display advertisments to maximize the likelihood of customer's purchasing products?

One way to answer this question is to extract the time of the day from the 'Order Date' coloumn, and then grouping the data based on the time of the day (hour) in which a product was purchased to determine which times are associated with the most product purchases.

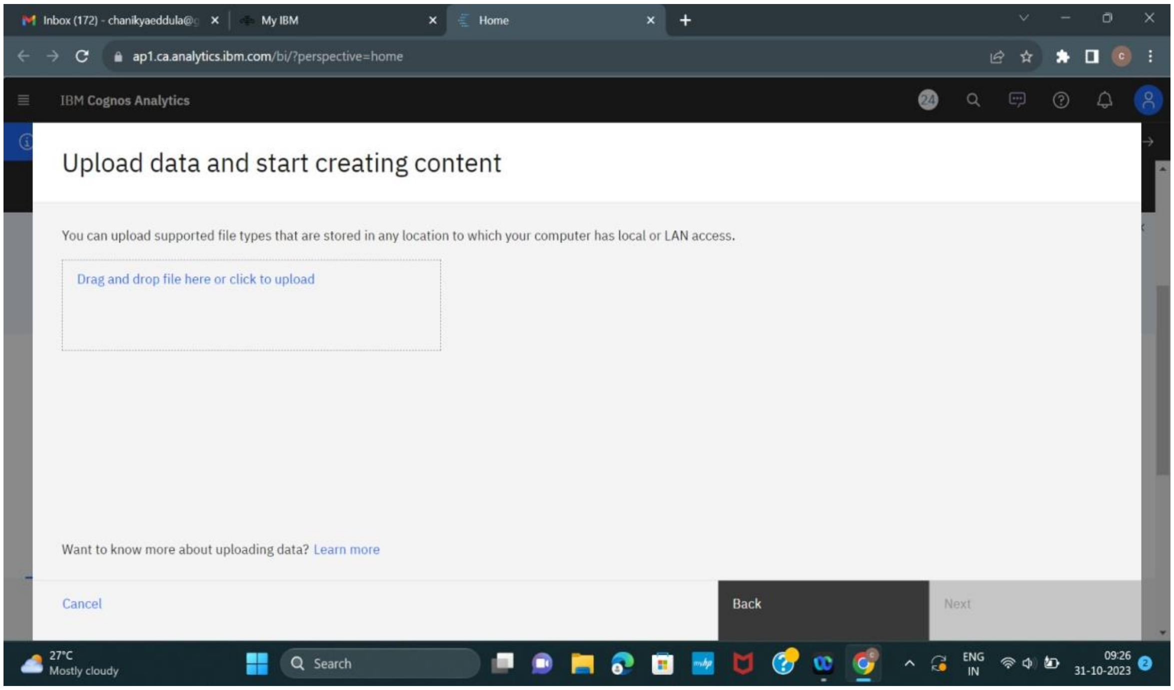
```
Time_col = pd.to_datetime(df['Order Date'], format='%d/%m/%y %H:%M').dt.strftime('%I %p')

df.insert(loc=6, column='Time of Purchase', value=Time_col)

purchases_per_hour = df.groupby(['Time of Purchase']).sum()['Quantity Ordered']

time_sort_order = ['12 AM', '01 AM', '02 AM', '03 AM', '04 AM', '05 AM', '06 AM', '07 AM', '08 AM', '09 AM', '10 AM', '11 AM', '12 PM', '01 PM', '02 PM', '03 PM', '04 PM', '05 PM', '06 PM', '07 PM', '08 PM', '09 PM', '10 PM', '11 PM']

purchases_per_hour = purchases_per_hour.reindex(time_sort_order)

df_purchases_per_hour = purchases_per_hour.to_frame(name='Total Quantity Sold')

print('The following table displays the total sum of quantities ordered for each hour of the day:')

df_purchases_per_hour

purchases_per_hour_sorted = purchases_per_hour.sort_values(ascending=False)

best_hour = purchases_per_hour_sorted.index[0]

print('The best time of day for displaying advertisements is:', best_hour)

time_of_purchase = purchases_per_hour.index.values
```

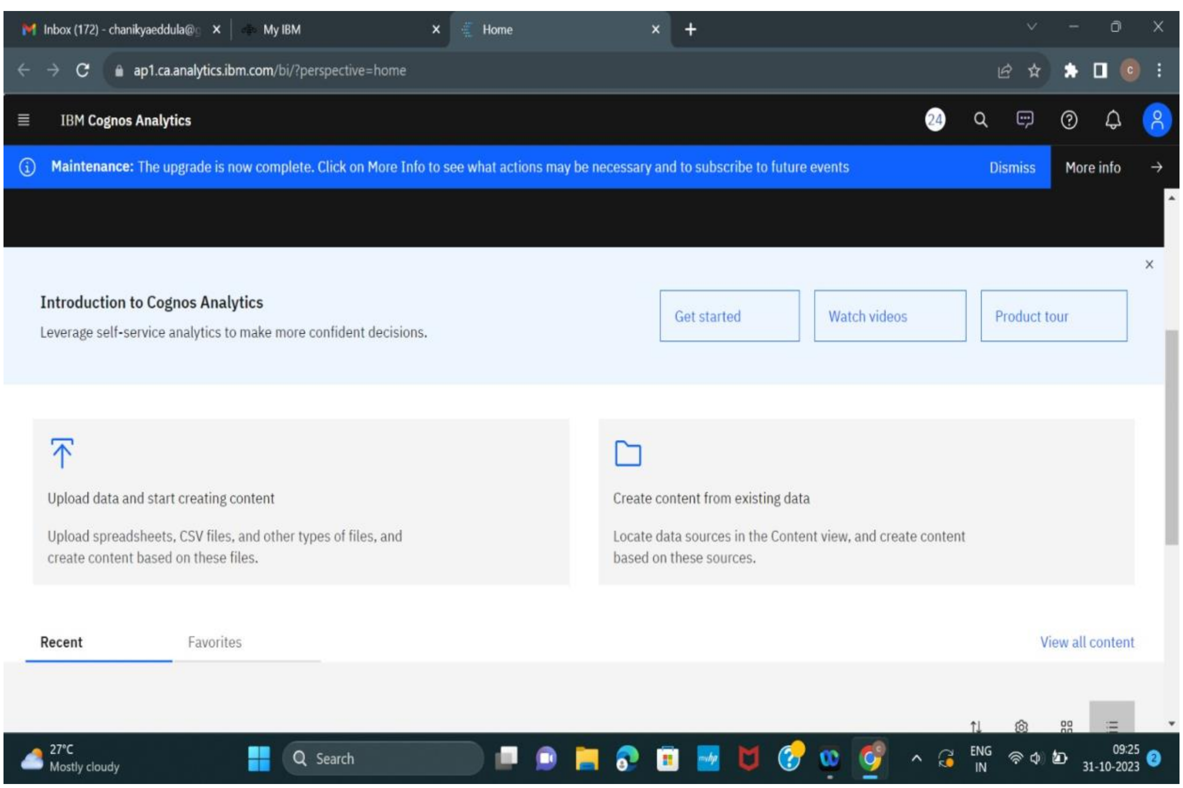
```
plt.figure(figsize=(12,7))
plt.bar(time_of_purchase,purchases_per_hour,color='#4169e1',linewidth=1,
edgecolor='k')
plt.title('Quantities Sold Per Hour', fontsize=15)
plt.xlabel('Time of Day', fontsize=13)
plt.ylabel('Amount of Quantities Sold', fontsize=13)
plt.xticks(rotation=90)
plt.gcf().axes[0].yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x
:,.0f}'))
plt.tight_layout()
plt.show()
```

Step1: login IBM cognos account on cloud

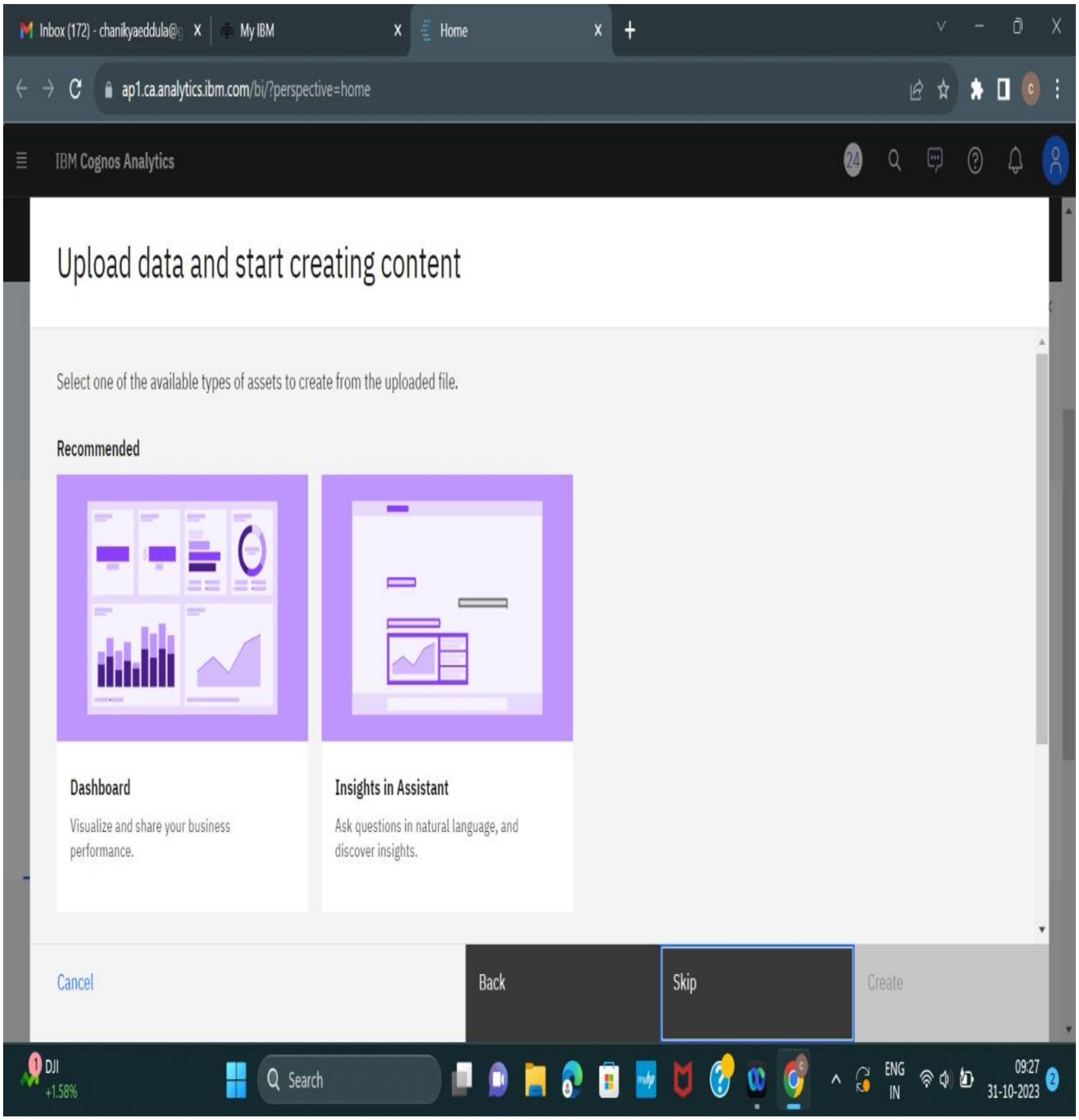


Step 2: Upload data in csv file

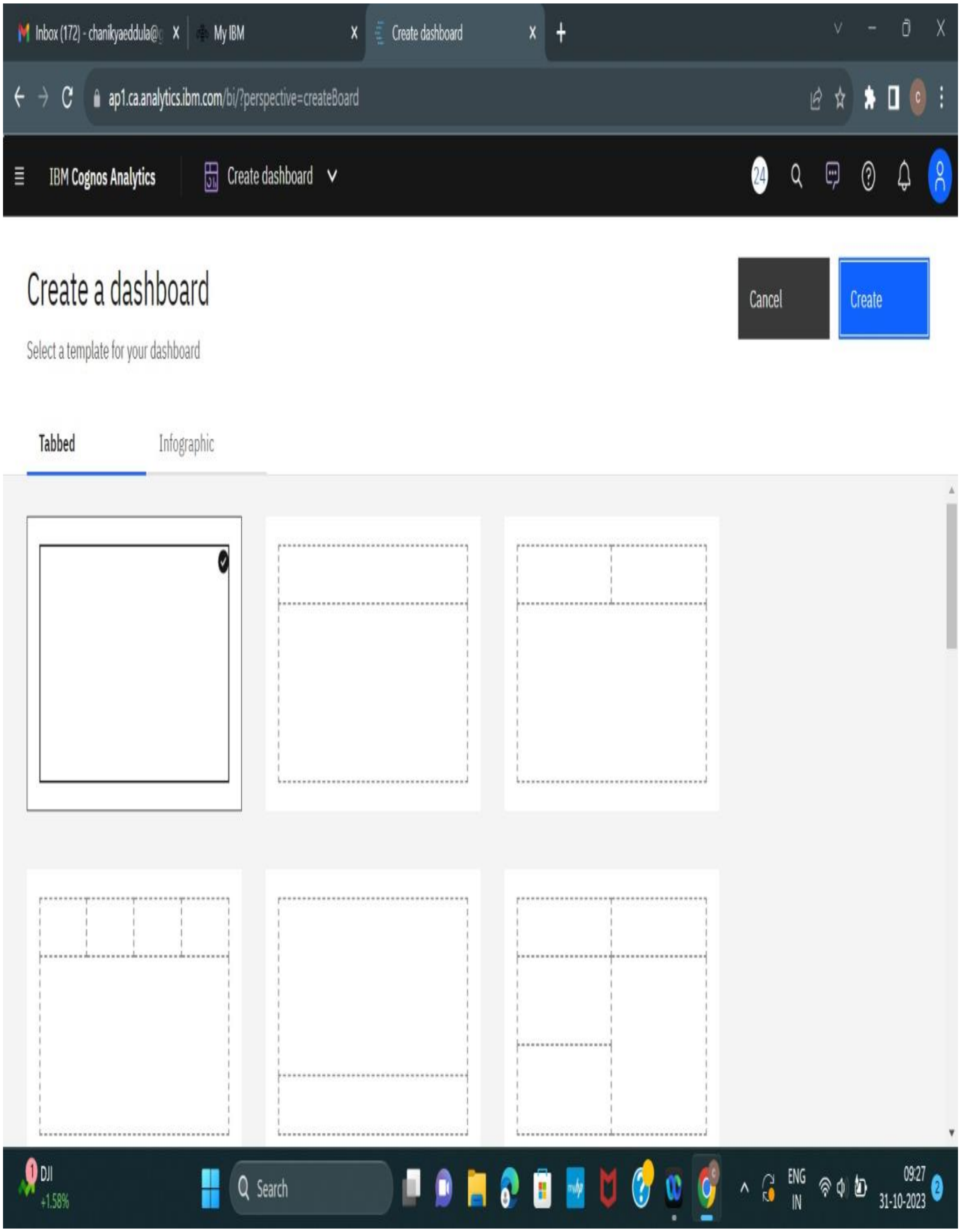




Step 3:drag and drop the file

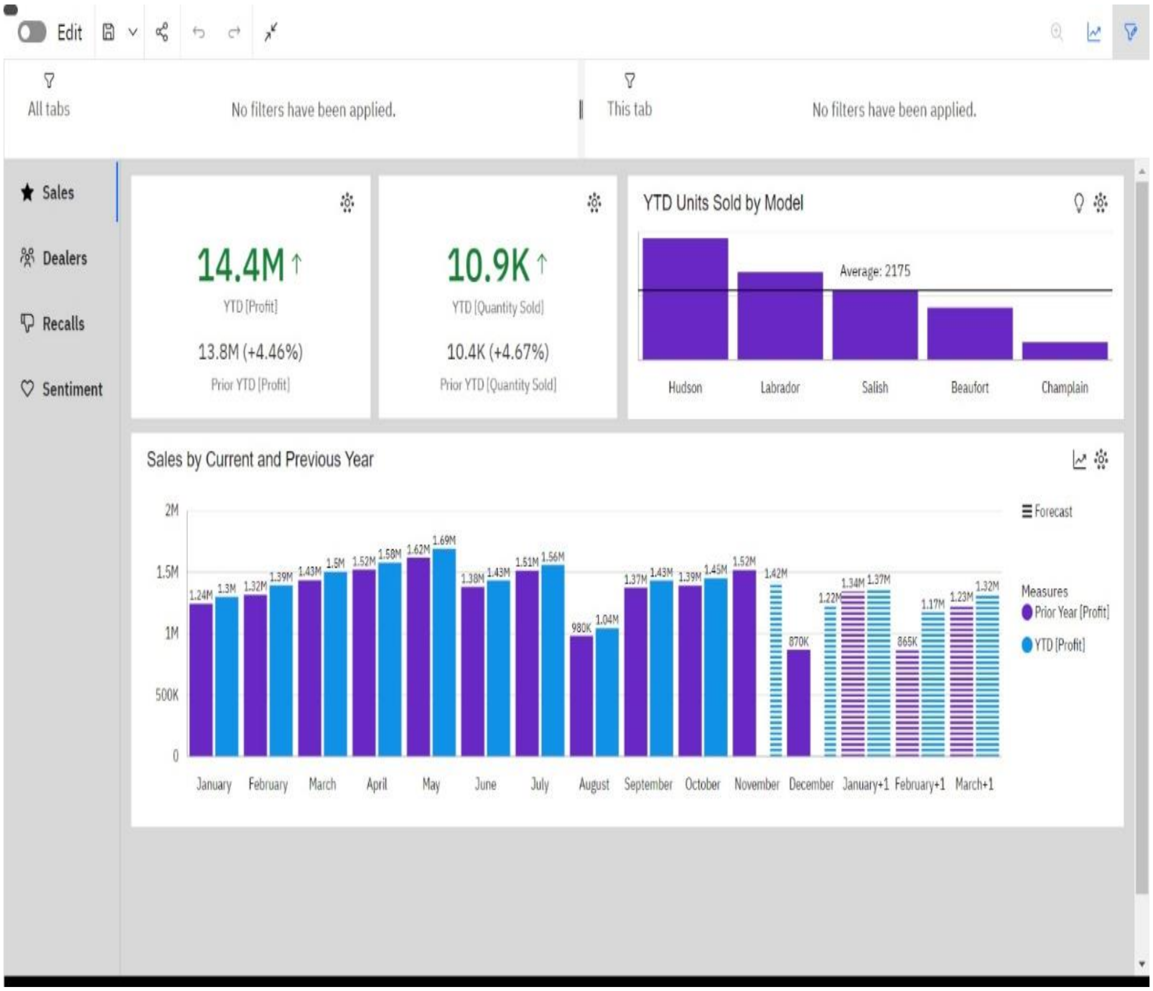


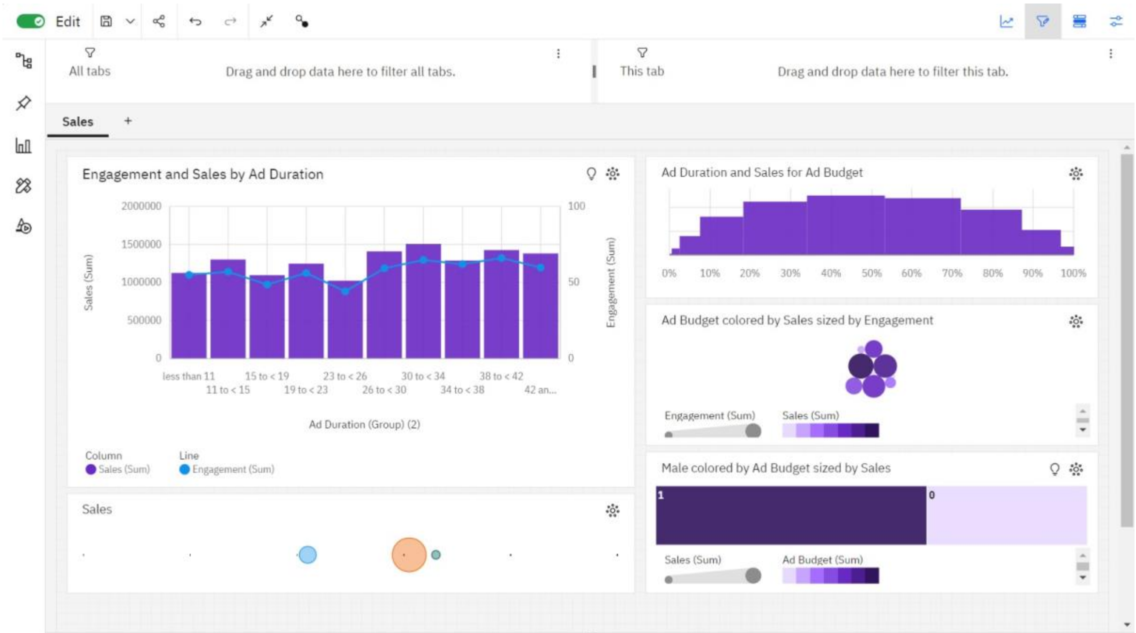
step 4: creat a dash board template

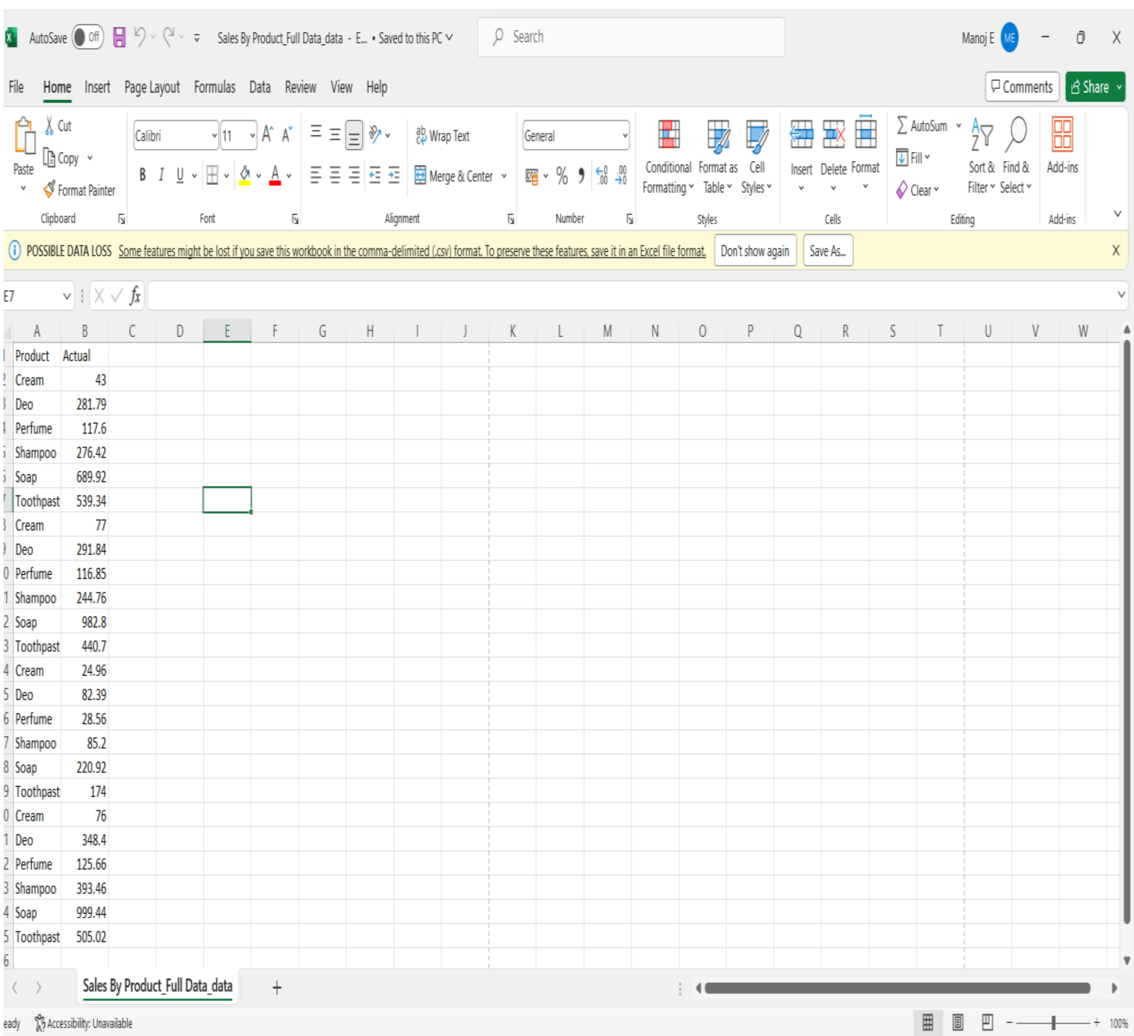


Dash board of sales analysis of product

blob



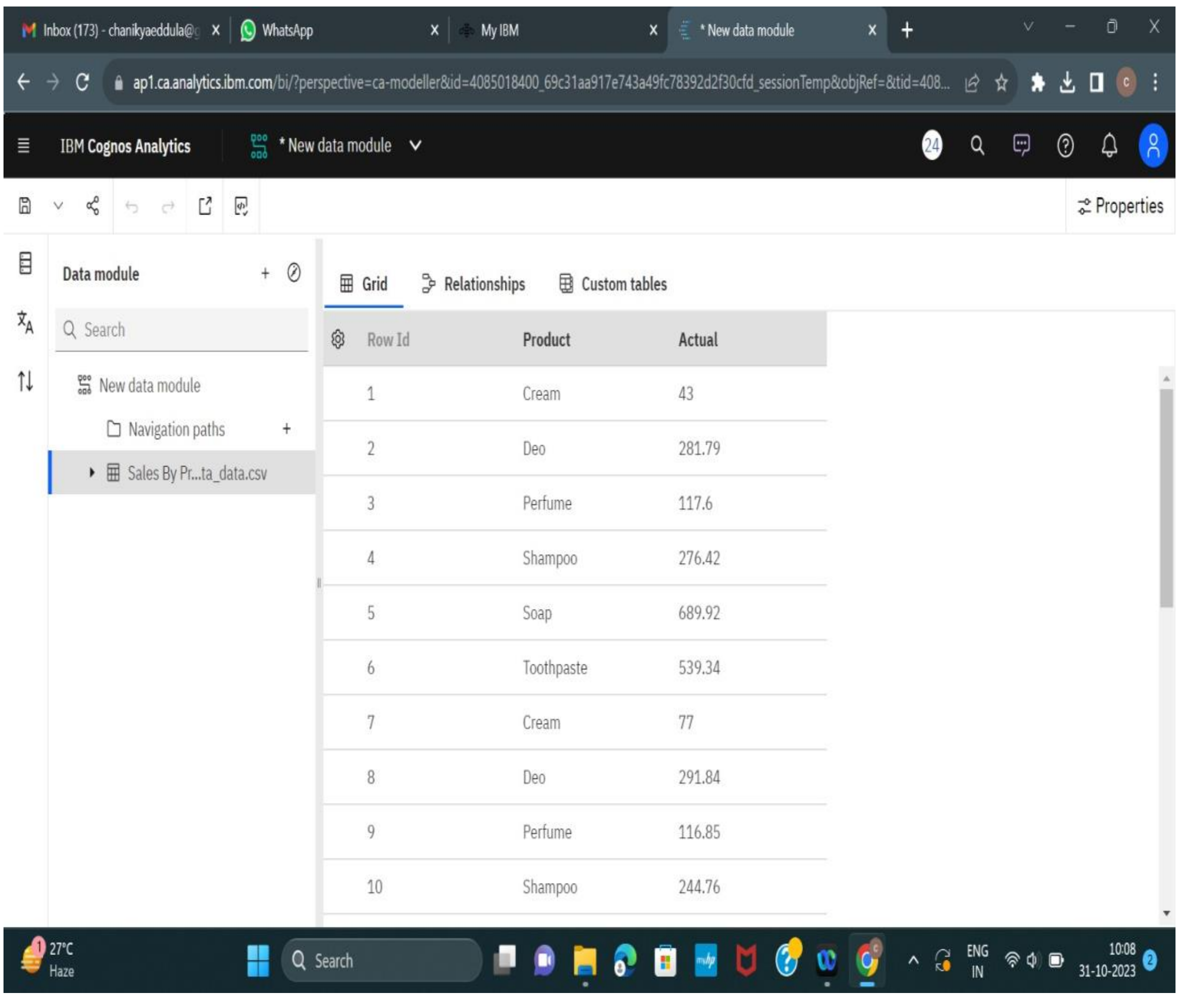




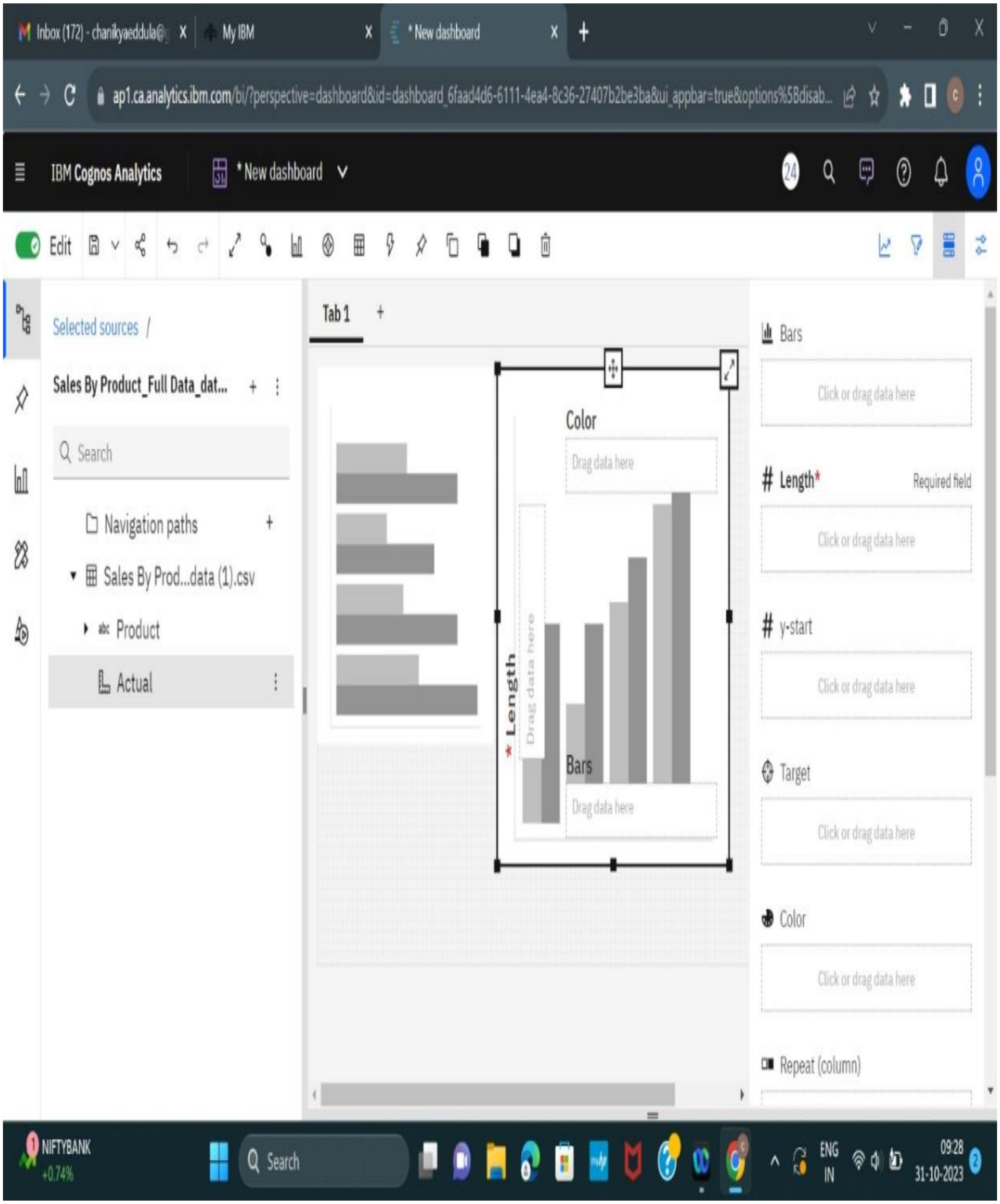
Product	Actual
Cream	43
Deo	281.79
Perfume	117.6
Shampoo	276.42
Soap	689.92
Toothpast	539.34
Cream	77
Deo	291.84
Perfume	116.85
Shampoo	244.76
Soap	982.8
Toothpast	440.7
Cream	24.96
Deo	82.39
Perfume	28.56
Shampoo	85.2
Soap	220.92
Toothpast	174
Cream	76
Deo	348.4
Perfume	125.66
Shampoo	393.46
Soap	999.44
Toothpast	505.02

step 5:here we to create visualizations of sales

Visualization of bar and column

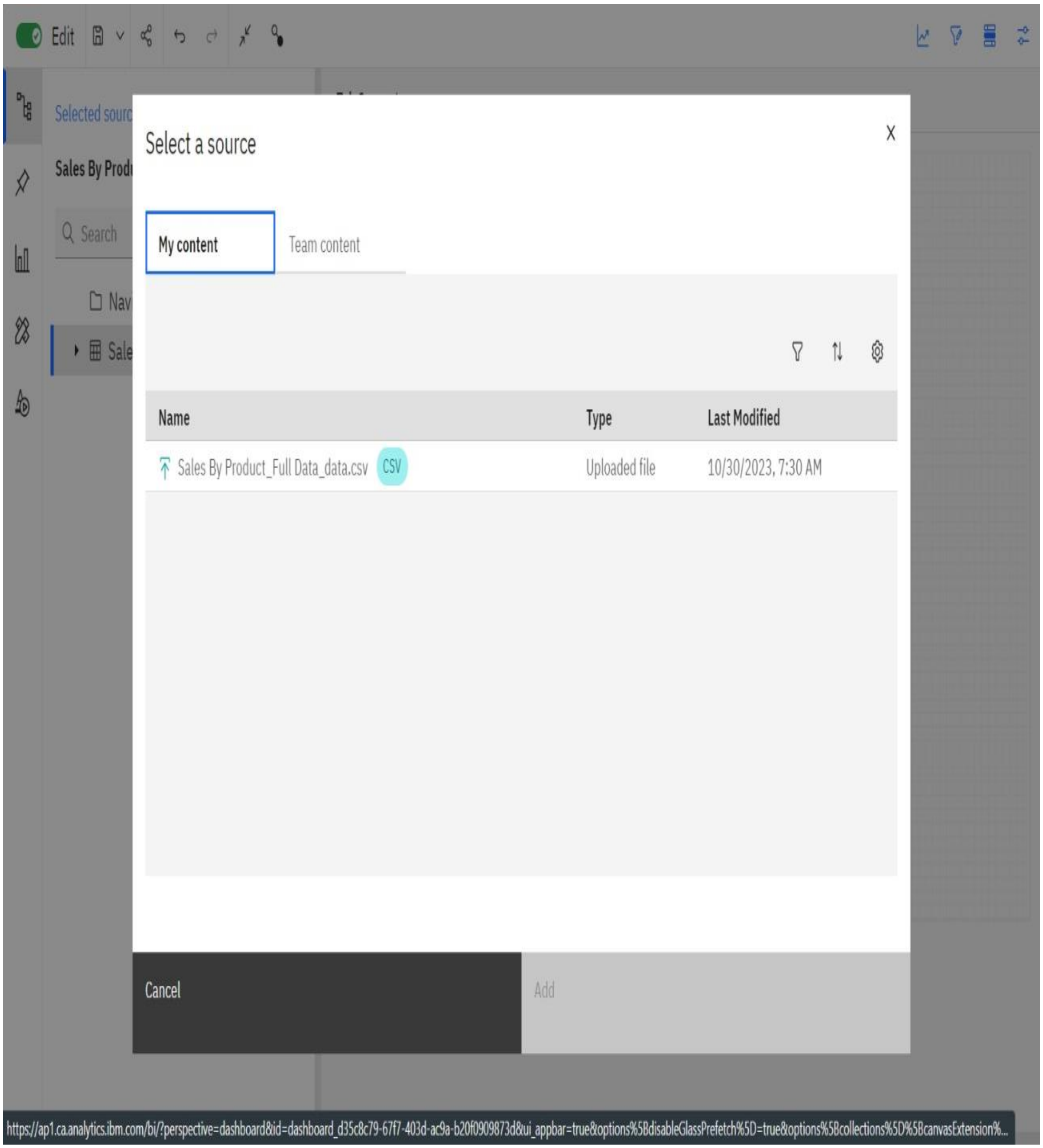


Step 6:to explore data module of sales analysis



after add the file in my contact source

one month to the next.



Exploring the Data

Question 1: What was the best month for sales? How much was earned that month?

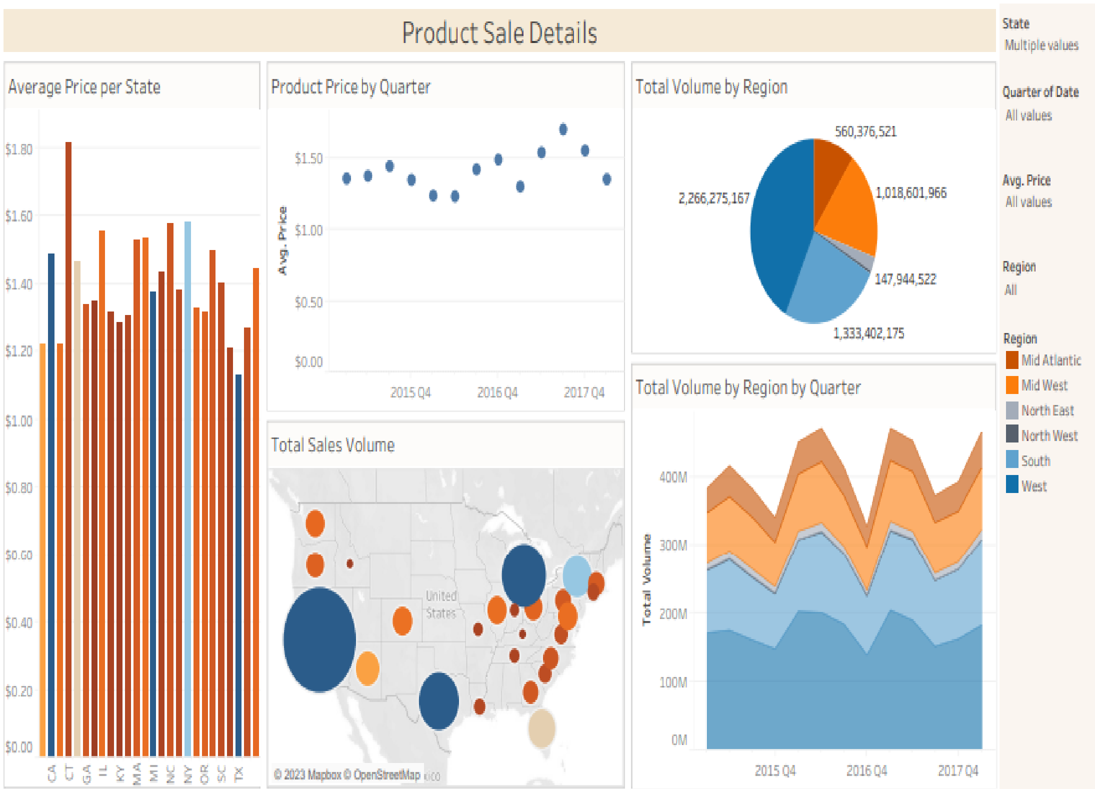
Out[14]:

	Order ID	Product	Quantity Ordered	Price Each	Sales	Order Date	Months	Purchase Address
0	176558	USB-C Charging Cable	2	11.950000	23.900000	19/04/19 08:46	Apr	917 1st St, Dallas, TX 75001
1	176559	Bose SoundSport Headphones	1	99.989998	99.989998	07/04/19 22:30	Jul	682 Chestnut St, Boston, MA 02215
2	176560	Google Phone	1	600.000000	600.000000	12/04/19 14:38	Dec	669 Spruce St, Los Angeles, CA 90001
3	176560	Wired Headphones	1	11.990000	11.990000	12/04/19 14:38	Dec	669 Spruce St, Los Angeles, CA 90001
4	176561	Wired Headphones	1	11.990000	11.990000	30/04/19 09:27	Apr	333 8th St, Los Angeles, CA 90001

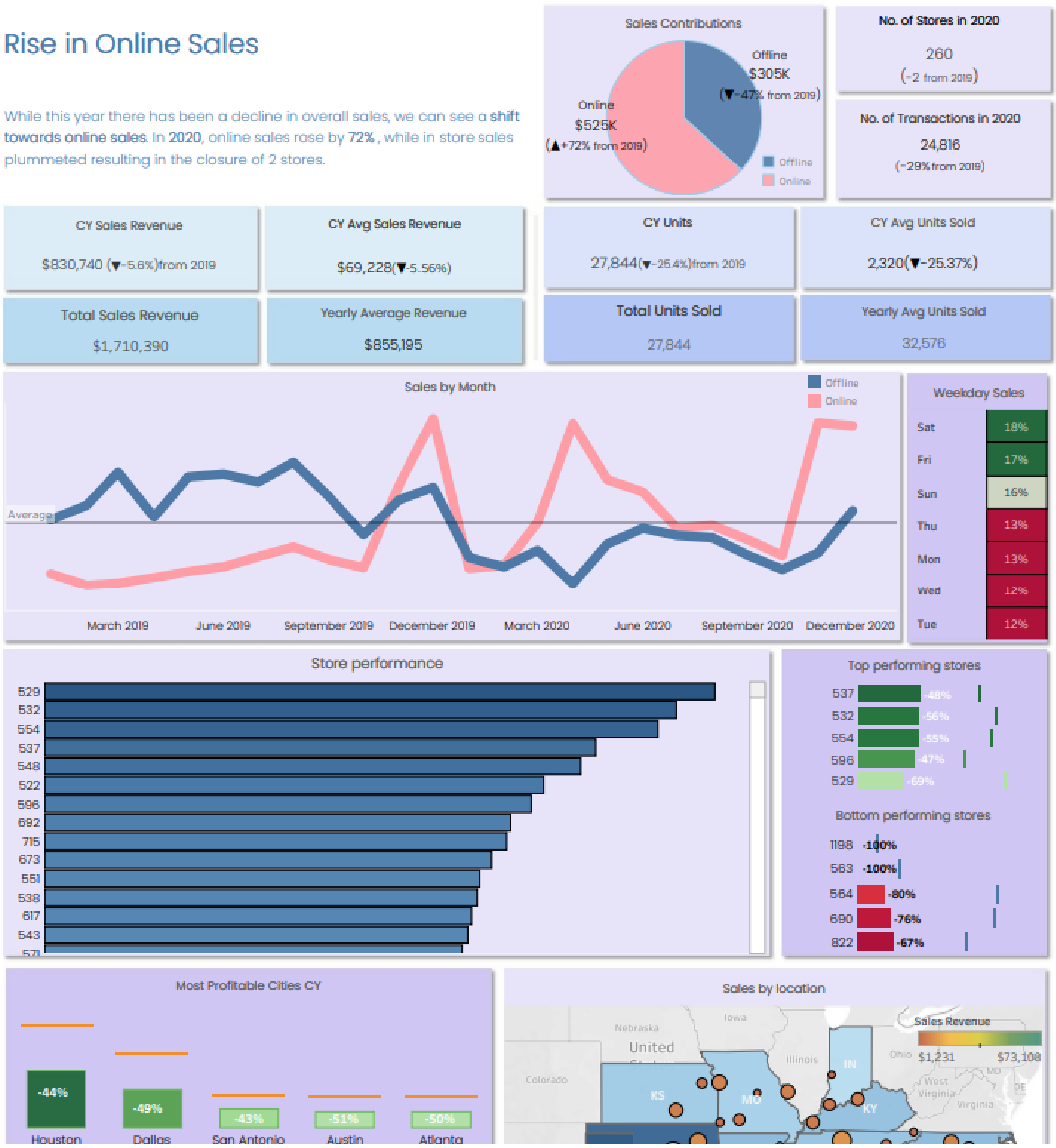
In [15]:

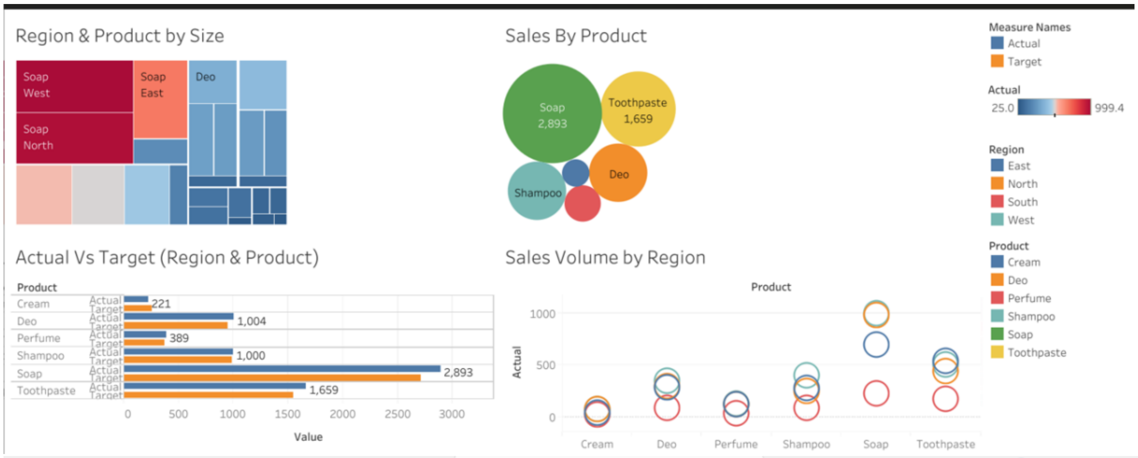
```
#Third, grouping the data by month and calculating the total sales (and quantities) amount for each month
sales_per_month = df.groupby(['Months']).sum()[['Quantity Ordered', 'Sales']]

#sorting the data in alphabetical order by month name
#specifying the sorting order
sort_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

#sorting the data
sales_per_month = sales_per_month.reindex(sort_order)

#to present the sales in USD and display them in a reader-friendly manner
sales_per_month_USD = sales_per_month.copy()
sales_per_month_USD['Sales'] = sales_per_month_USD['Sales'].apply(lambda sale: '${:,.2f}'.format(sale))

#displaying the results
print('The following table displays the total sales amount (and quantities ordered) for each month:')
sales_per_month_USD
```

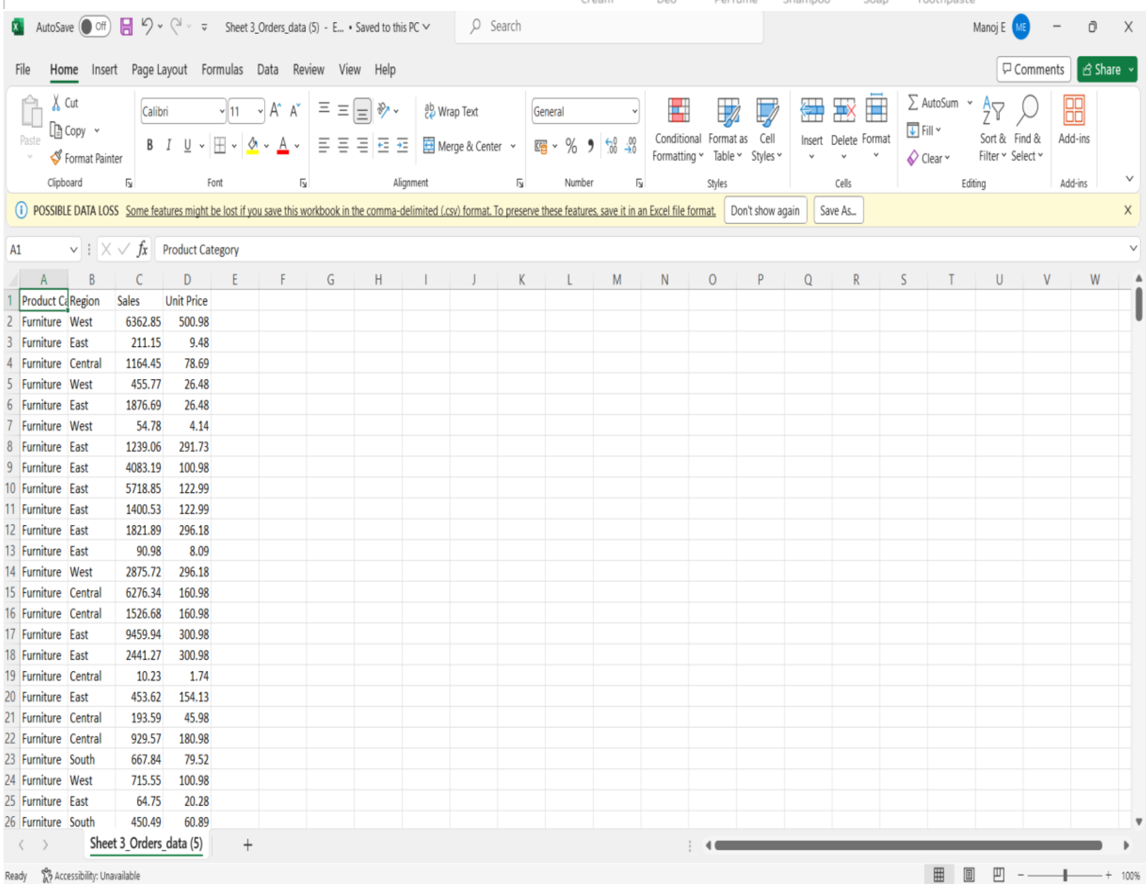
```
The following table displays the total sales amount (and quantities ordered) for each month:
```



Product Sale Details
Average Price per State
$1.80
$1.60
$1.40
$1.20
$1.00
$0.80
$0.60
$0.40
$0.20
$0.00
CA
CT
GA
IL
KY
MA
MI
NC
NY
OR
SC
TX
Product Price by Quarter
Avg. Price
$1.50
$1.00
$0.50
$0.00
2015 Q4
2016 Q4
2017 Q4
Total Sales Volume
United States
© 2023 Mapbox © OpenStreetMap
Total Volume by Region
560,376,521
1,018,601,966
147,944,522
1,333,402,175
2,266,275,167
Total Volume by Region by Quarter
Total Volume
400M
300M
200M
100M
0M
2015 Q4
2016 Q4
2017 Q4
State
Multiple values
Quarter of Date
All values
Avg. Price
All values
Region
All
Region
Mid Atlantic
Mid West
North East
North West
South
West


Rise in Online Sales
While this year there has been a decline in overall sales, we can see a shift towards online sales. In 2020, online sales rose by 72% , while in store sales plummeted resulting in the closure of 2 stores.
Sales Contributions
Offline
$305K
(▼-47% from 2019)
Online
$525K
(▲+72% from 2019)
Offline
Online
No. of Stores in 2020
260
(-2 from 2019)
No. of Transactions in 2020
24,816
(-29% from 2019)
CY Sales Revenue
$830,740 (▼-5.6%)from 2019
CY Avg Sales Revenue
$69,228(▼-5.56%)
CY Units
27,844(▼-25.4%)from 2019
CY Avg Units Sold
2,320(▼-25.37%)
Total Sales Revenue
$1,710,390
Yearly Average Revenue
$855,195
Total Units Sold
27,844
Yearly Avg Units Sold
32,576
Sales by Month
Offline
Online
Average
March 2019
June 2019
September 2019
December 2019
March 2020
June 2020
September 2020
December 2020
Weekday Sales
Sat 18%
Fri 17%
Sun 16%
Thu 13%
Mon 13%
Wed 12%
Tue 12%
Store performance
529
532
554
537
548
522
596
692
715
673
551
538
617
543
Top performing stores
537 -48%
532 -56%
554 -55%
596 -47%
529 -69%
Bottom performing stores
1198 -100%
563 -100%
564 -80%
690 -76%
822 -67%
Most Profitable Cities CY
-44% Houston
-49% Dallas
-43% San Antonio
-51% Austin
-50% Atlanta
Sales by location
Sales Revenue
$1,231
$73,108
Nebraska
Iowa
Illinois
Colorado
Ohio
West Virginia
Virginia
KS
MO
IN
KY

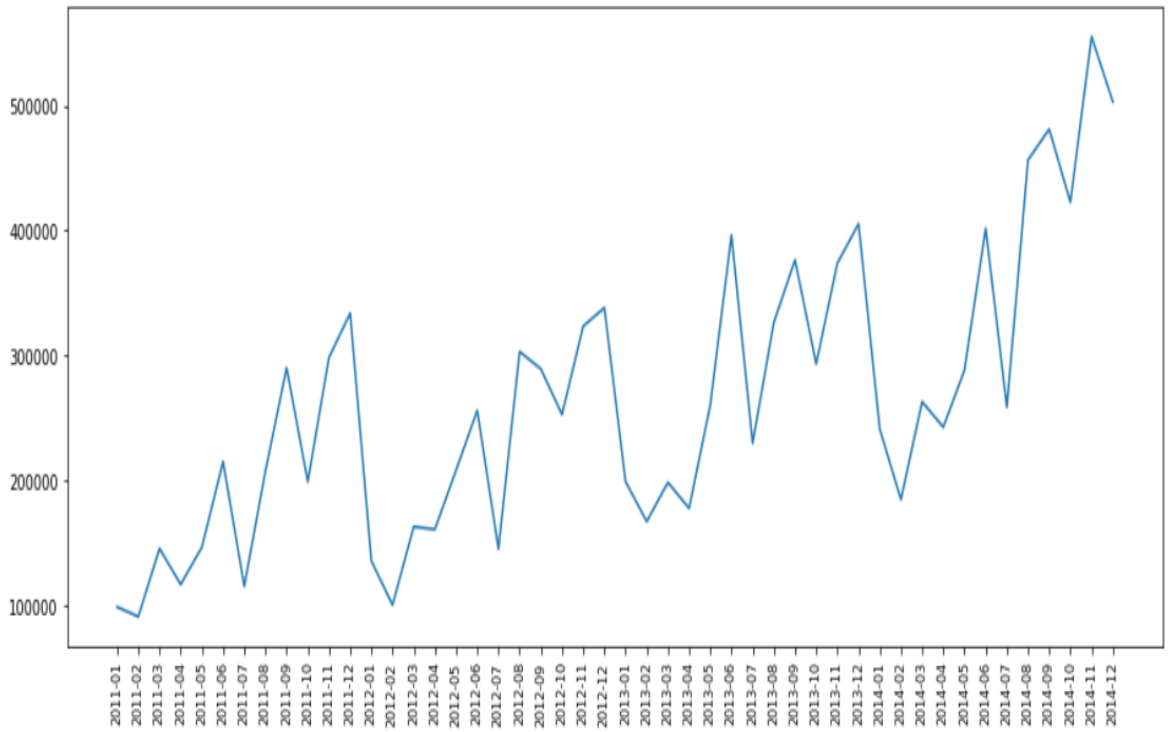




Product Category	Region	Sales	Unit Price
Furniture	West	6362.85	500.98
Furniture	East	211.15	9.48
Furniture	Central	1164.45	78.69
Furniture	West	455.77	26.48
Furniture	East	1876.69	26.48
Furniture	West	54.78	4.14
Furniture	East	1239.06	291.73
Furniture	East	4083.19	100.98
Furniture	East	5718.85	122.99
Furniture	East	1400.53	122.99
Furniture	East	1821.89	296.18
Furniture	East	90.98	8.09
Furniture	West	2875.72	296.18
Furniture	Central	6276.34	160.98
Furniture	Central	1526.68	160.98
Furniture	East	9459.94	300.98
Furniture	East	2441.27	300.98
Furniture	Central	10.23	1.74
Furniture	East	453.62	154.13
Furniture	Central	193.59	45.98
Furniture	Central	929.57	180.98
Furniture	South	667.84	79.52
Furniture	West	715.55	100.98
Furniture	East	64.75	20.28
Furniture	South	450.49	60.89

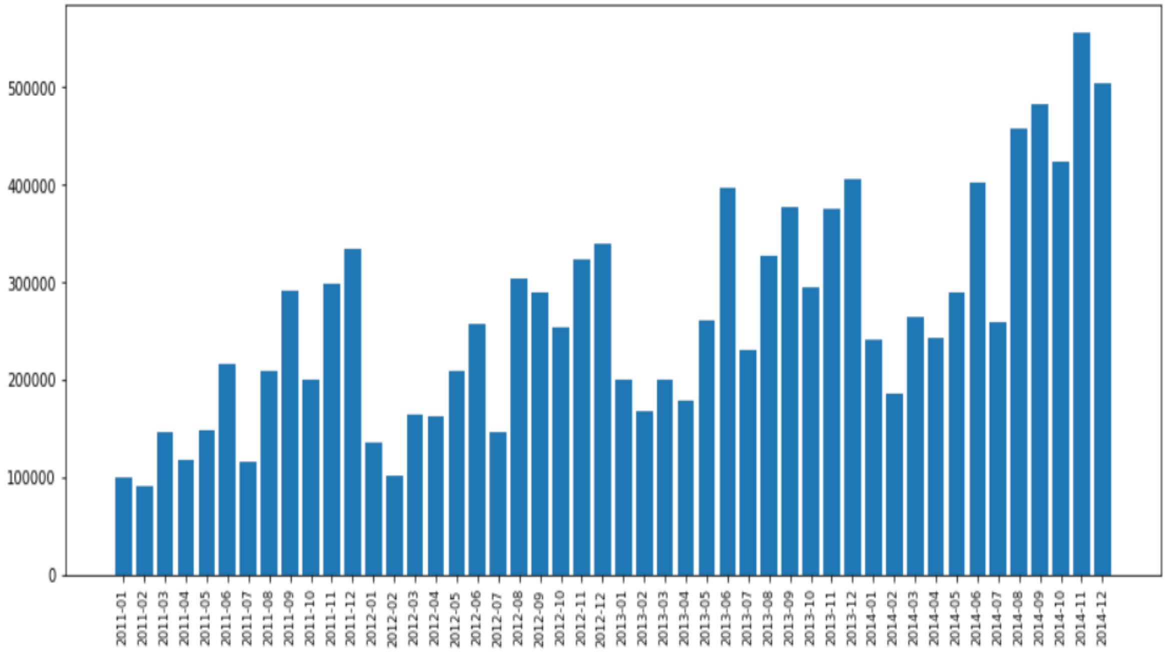
VISUALIZE SALES TRAND BY MONTHS

```
plt.figure(figsize=(15,6))
plt.plot(sales_by_month['month_year'],sales_by_month['sales'])
plt.xticks(rotation='vertical',size=8)
plt.show()
```



]:

```
plt.figure(figsize=(15,6))
plt.bar(sales_by_month['month_year'],sales_by_month['sales'])
plt.xticks(rotation='vertical',size=8)
plt.show()
```



DISPLAY MOST SELLING PRODCUTS

In [22]:

```
products_sales = pd.DataFrame(sales.groupby('product_name').sum()['sales'])
products_sales = products_sales.sort_values('sales',ascending=False)
```

TOP 10 MOST SALES PRODUCTS

In [23]:

```
products_sales[:10]
```

Out[23]:

product_name	sales
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306
Motorola Smart Phone, Full Size	73156.3030
Nokia Smart Phone, Full Size	71904.5555
Canon imageCLASS 2200 Advanced Copier	61599.8240
Hon Executive Leather Armchair, Adjustable	58193.4841
Office Star Executive Leather Armchair, Adjustable	50661.6840
Harbour Creations Executive Leather Armchair, Adjustable	50121.5160
Samsung Smart Phone, Cordless	48653.4600
Nokia Smart Phone, with Caller ID	47877.7857

```
products_by_quantity = pd.DataFrame(sales.groupby('product_name').sum()['quantity'])
products_by_quantity_sorted = products_by_quantity.sort_values('quantity',ascending=False)
```

TOP 10 MOST QUANTITY SELLING PRODUCTS ITEMS

```
products_by_quantity_sorted[:10]
```

product_name	quantity
Staples	876
Cardinal Index Tab, Clear	337
Eldon File Cart, Single Width	321
Rogers File Cart, Single Width	262
Sanford Pencil Sharpener, Water Color	259
Stockwell Paper Clips, Assorted Sizes	253
Avery Index Tab, Clear	252
Ibico Index Tab, Clear	251
Smead File Cart, Single Width	250
Stanley Pencil Sharpener, Water Color	242

VISUALIZE MOST USED SHIP MODS

In [14]:

```
sales.isnull().sum()
```

Out[14]:

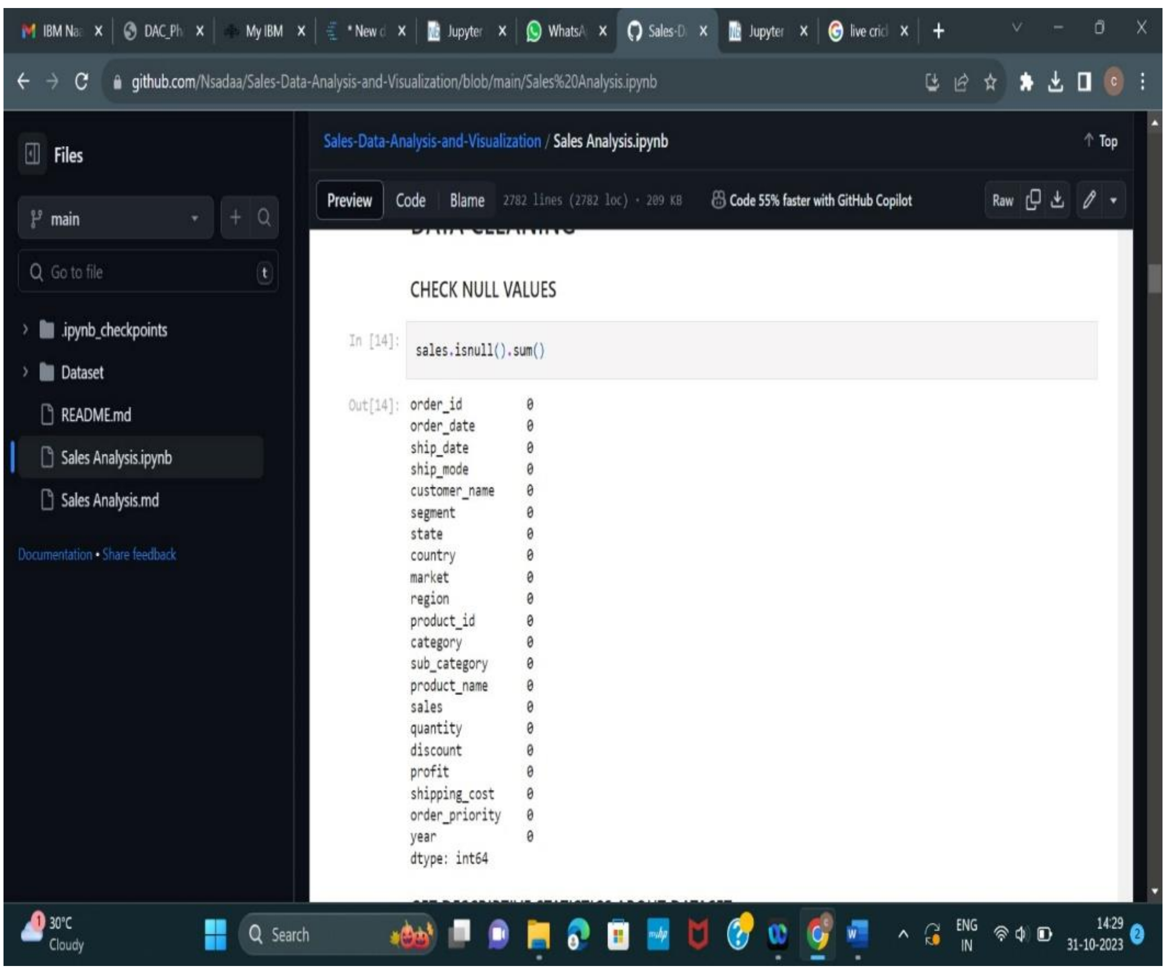
```
order_id          0
order_date        0
ship_date         0
ship_mode         0
customer_name     0
segment           0
state             0
country           0
market            0
region            0
product_id        0
category          0
sub_category      0
product_name      0
sales             0
quantity          0
discount          0
profit            0
shipping_cost     0
order_priority    0
year              0
dtype: int64
```

GET INFORMATIONS ABOUT DATASET

[13]:
```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   order_id        51290 non-null  object
 1   order_date      51290 non-null  datetime64[ns]
 2   ship_date       51290 non-null  datetime64[ns]
 3   ship_mode       51290 non-null  object
 4   customer_name   51290 non-null  object
 5   segment         51290 non-null  object
 6   state           51290 non-null  object
 7   country         51290 non-null  object
 8   market          51290 non-null  object
 9   region          51290 non-null  object
 10  product_id      51290 non-null  object
 11  category        51290 non-null  object
 12  sub_category    51290 non-null  object
 13  product_name    51290 non-null  object
 14  sales           51290 non-null  float64
 15  quantity        51290 non-null  int64
 16  discount        51290 non-null  float64
 17  profit          51290 non-null  float64
 18  shipping_cost   51290 non-null  float64
 19  order_priority  51290 non-null  object
 20  year            51290 non-null  int64
dtypes: datetime64[ns](2), float64(4), int64(2), object(13)
memory usage: 8.2+ MB
```

DATA CLEANING



GETTING KNOW ABOUT DATSET SHAPE & COLUMNS

In [11]:
```
sales.shape
```

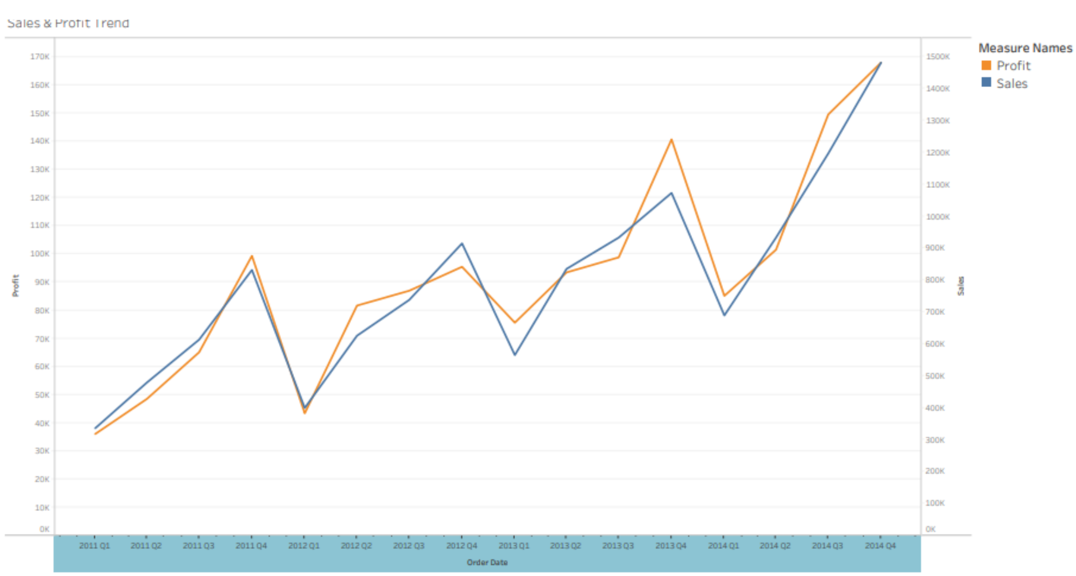
Out[11]: (51290, 21)

In [12]:
```
for columns in sales.columns:
        print(columns)
```

```
order_id
order_date
ship_date
ship_mode
customer_name
segment
state
country
market
region
product_id
category
sub_category
product_name
sales
quantity
discount
profit
shipping_cost
order_priority
year
```

GET INFORMATIONS ABOUT DATASET

Sales and profit trend



Conclusion:

In conclusion, the utilization of IBM cognos for visualizing product sales data has been instrumental in providing a historic view of sales performance.The insights derived from these visualizations are pivotal for formulating data driven strategies.These conclusions can serve as a foundation for future business decisions and actions aimed at driving sales growth.These document shows how to login IBM cognos and how to create visualizations for sales and dervie for insights