PRACTICAL 6

AIM: To implement movie recommendation systems using K-means clustering algorithm.

CONCEPT:

- K-means clustering is a technique used to organize data into groups based on their similarity.
- For example online store uses K-Means to group customers based on purchase frequency and spending creating segments like Budget Shoppers, Frequent Buyers and Big Spenders for personalised marketing.
- The algorithm works by first randomly picking some central points called centroids and each data point is then assigned to the closest centroid forming a cluster.
- After all the points are assigned to a cluster the centroids are updated by finding the average position of the points in each cluster.
- This process repeats until the centroids stop changing forming clusters. The goal of clustering is to divide the data points into clusters so that similar data points belong to same group.

LIBRARIES USED:

- Panda
- Warning
- Sklearn
- KMean
- NumPy

CODE:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
movies = pd.read_csv('/content/movies.csv')
ratings = pd.read_csv('/content/ratings.csv')
data = pd.merge(ratings,movies,on='movieId')
```

```
user_item_matrix = data.pivot_table(index='userId', columns='title',
values='rating').fillna(0)
scaler = StandardScaler()
user_item_matrix_scaled = scaler.fit_transform(user_item_matrix)
kmeans = KMeans(n_clusters=10, random_state=42)
user_clusters = kmeans.fit_predict(user_item_matrix_scaled)
user_item_matrix['cluster'] = user_clusters
import numpy as np
def
recommend_movies(user_id,user_item_matrix,data,movies,num_recommendations
=5):
  user_cluster = user_item_matrix.loc[user_id,'cluster']
  similar_users =
user_item_matrix[user_item_matrix['cluster']==user_cluster].index
  similar_users_data = data[data['userId'].isin(similar_users)]
  movie_ratings = similar_users_data.groupby('title')['rating'].mean()
  user_rated_movies = data[data['userId'] == user_id]['title'].tolist()
  user_top_movies = data[data['userId']==user_id].groupby('title')['rating'].mean()
  user_top_movies = user_top_movies[user_top_movies>=4.0]
  user_top_genres =
movies[movies['title'].isin(user_top_movies.index)]['genres'].str.split('|').explode().
value_counts().index[:3]
  movies['is_relevant_genre'] = movies['genres'].apply(lambda x: any(genre in x for
genre in user_top_genres))
  recommended_movies =
movie_ratings[~movie_ratings.index.isin(user_rated_movies)].sort_values(ascendi
ng=False)
  recommended_movies = recommended_movies.to_frame().merge(movies,
on='title')
  recommended_movies=
recommended_movies[recommended_movies['is_relevant_genre']]
  return recommended_movies[['title', 'rating']].head(num_recommendations)
recommendations = recommend_movies(user_id=32,
user_item_matrix=user_item_matrix, data=data, movies=movies)
print(recommendations)
```

OUTPUT:

```
                                      title  rating
0                     Che: Part Two (2008)     5.0
1                  Monster Squad, The (1987)     5.0
2             What We Do in the Shadows (2014)     5.0
4        Red Sorghum (Hong gao liang) (1987)     5.0
6  Into the Forest of Fireflies' Light (2011)     5.0
```

PRACTICAL 7

AIM : To implement tf-idf(term frequency-inverse document frequency) analysis.

CONCEPT :

- Information retrieval and text mining both employ TF-IDF, or term frequency-inverse document frequency, as a numerical statistic.
- Its purpose is to represent a word's significance inside a document about a group of texts (referred to as a corpus).
- The TF-IDF value rises in direct proportion to the number of times a word occurs in the text; however, the frequency of the term in the corpus counteracts this increase, helping to account for the fact that certain words occur more frequently than others.
- The assessment of a term's relevance over a corpus of texts is called Inverse Document Frequency (IDF).
- It is useful to determine a term's frequency or rarity across the corpus. A term with a high IDF score is considered uncommon across the papers, whereas a term with a low IDF value is considered frequent.

LIBRARIES USED :

- Pandas
- Numpy
- Sklearn
- NLTK

CODE :

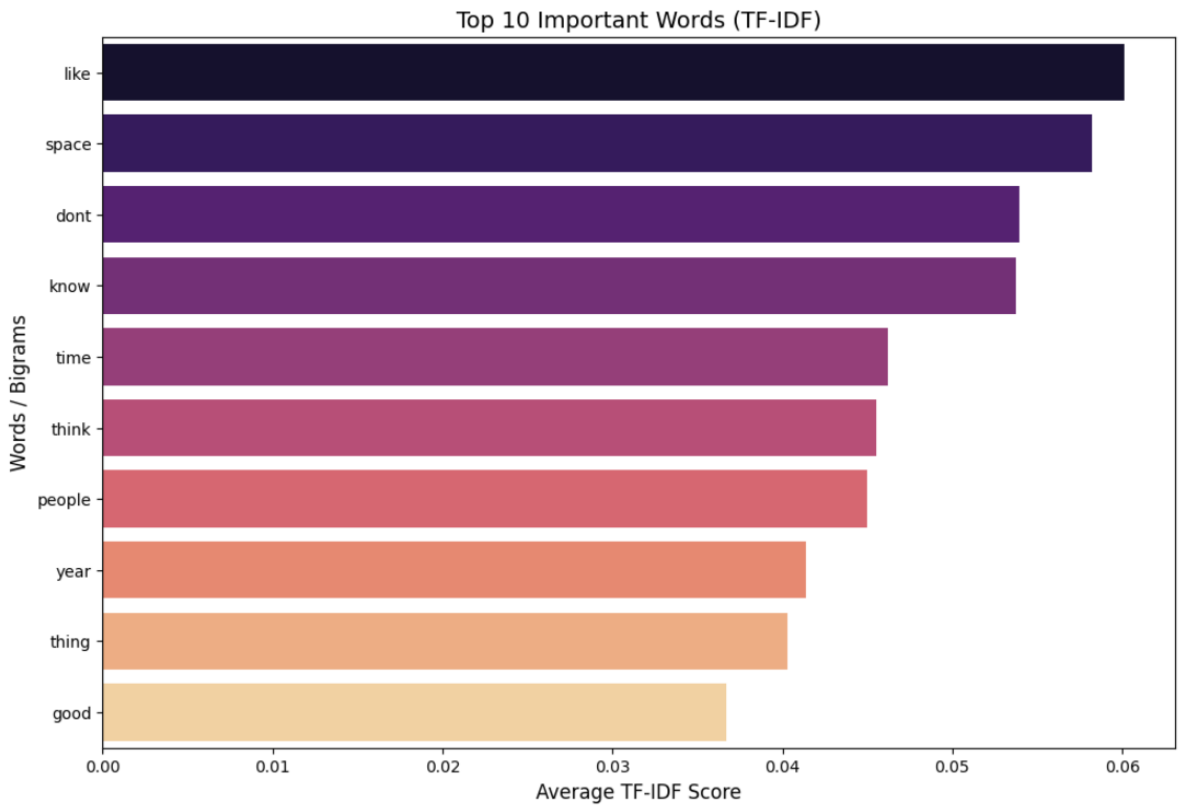
```
!pip install nltk scikit-learn matplotlib seaborn pandas numpy
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.datasets import fetch_20newsgroups
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

```
from nltk.stem import WordNetLemmatizer
import nltk
import string
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
categories = ['sci.space', 'sci.med']
newsgroups = fetch_20newsgroups(subset='train', categories=categories,
remove=('headers', 'footers', 'quotes'))
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans("", "", string.punctuation))
    words = word_tokenize(text)  # Tokenization
    words = [lemmatizer.lemmatize(word) for word in words if word not in
stop_words and len(word) > 2]
    return ' '.join(words)
documents = [preprocess_text(doc) for doc in newsgroups.data]  # Apply
preprocessing
vectorizer = TfidfVectorizer(ngram_range=(1,2), max_features=100,
stop_words="english")  # Use unigrams & bigrams
tfidf_matrix = vectorizer.fit_transform(documents)
df = pd.DataFrame(tfidf_matrix.toarray(),
columns=vectorizer.get_feature_names_out())
word_scores = df.mean().sort_values(ascending=False)
import textwrap
def wrap_labels(labels, width=15):
    return ["\n".join(textwrap.wrap(label, width)) for label in labels]
top_n = 10
wrapped_labels = wrap_labels(word_scores.index[:top_n], width=15)
plt.figure(figsize=(12, 8))
sns.barplot(x=word_scores.values[:top_n], y=wrapped_labels, palette="magma")
plt.xlabel("Average TF-IDF Score", fontsize=12)
```

```
plt.ylabel("Words / Bigrams", fontsize=12)
plt.title(f"Top {top_n} Important Words (TF-IDF)", fontsize=14)
plt.yticks(fontsize=10)
plt.show()
```

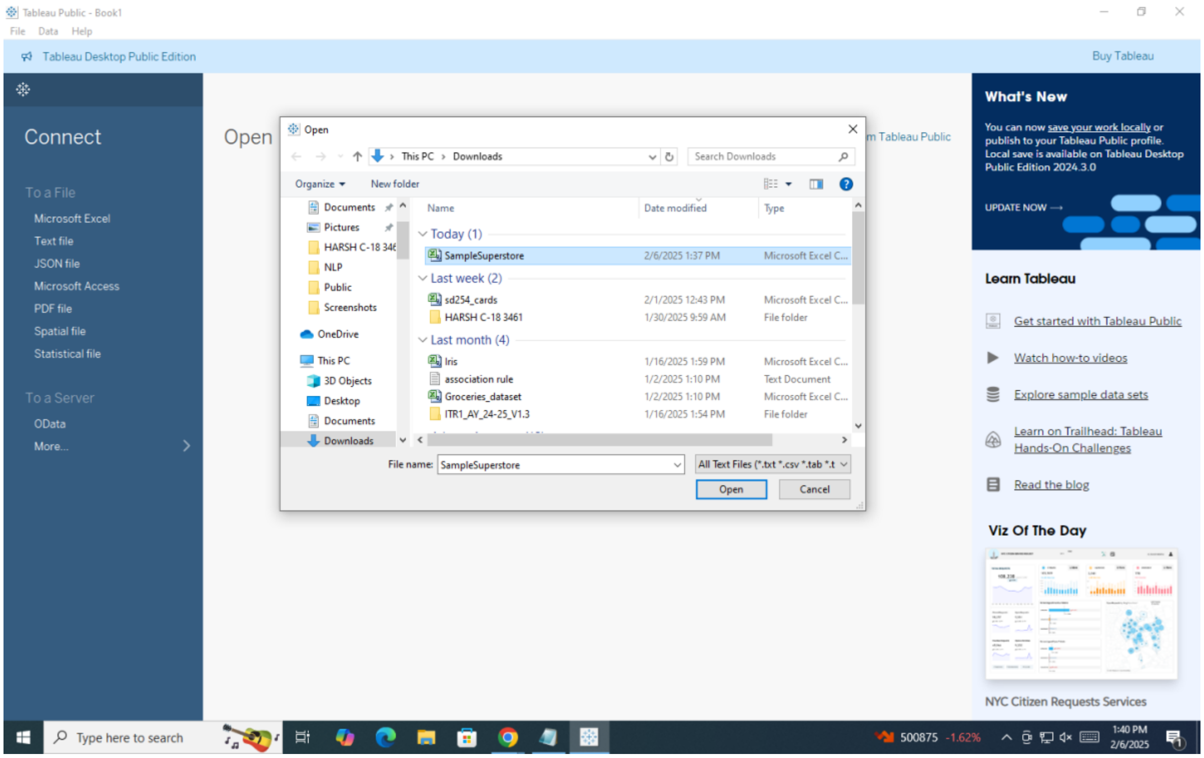
OUTPUT :

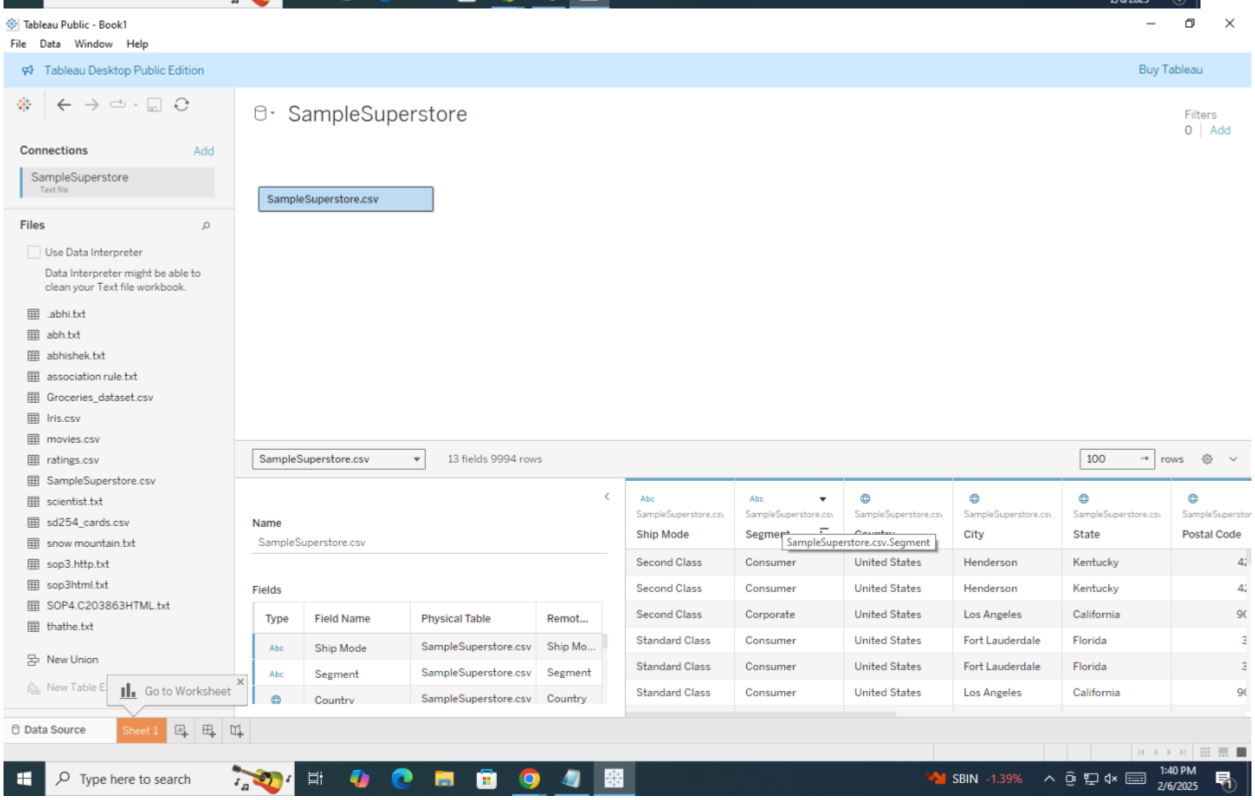
```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.55.7)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
<ipython-input-2-6d004d40d289>:63: FutureWarning:
```


Top 10 Important Words (TF-IDF)
like
space
dont
know
time
think
people
year
thing
good
Words / Bigrams
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Average TF-IDF Score

PRACTICAL 8

AIM : To implement data visualization using tableau and to generate summary of the data and create dashboards.





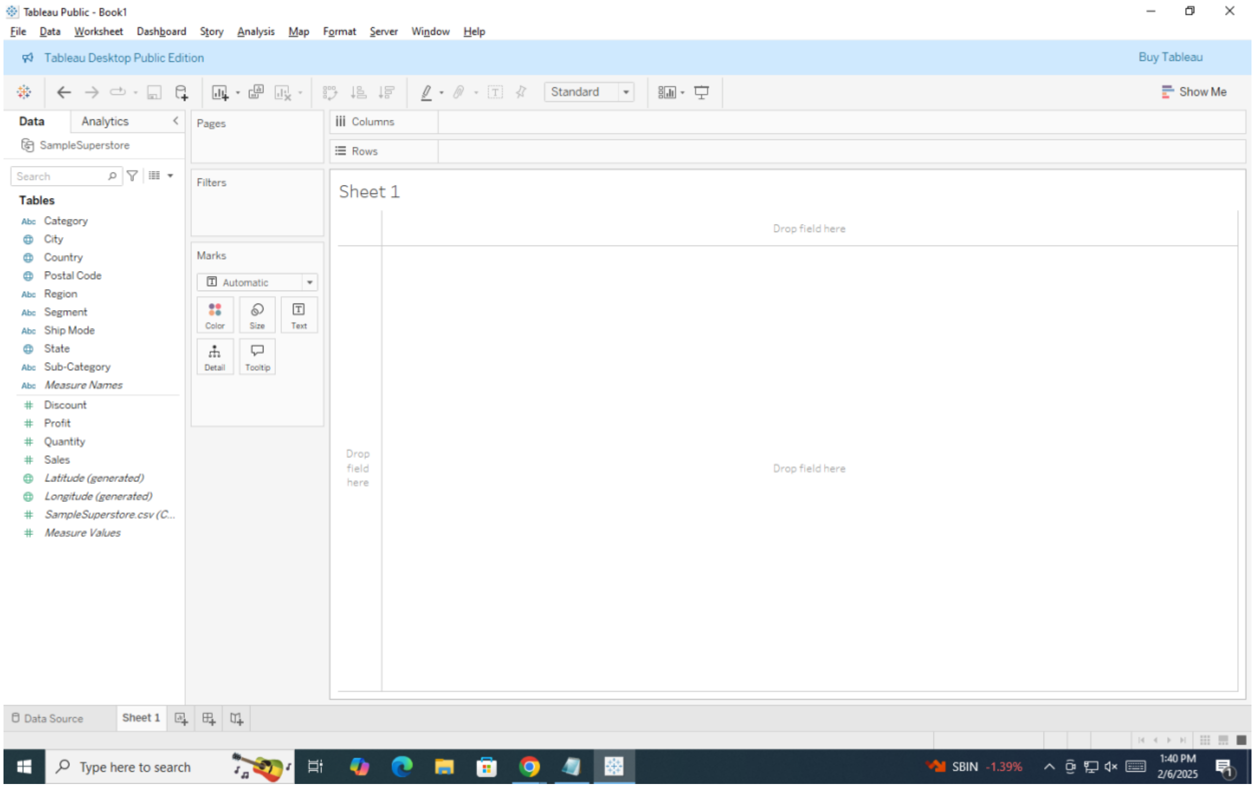


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Standard
Show Me
Data
Analytics
SampleSuperstore
Pages
Columns
Rows
Filters
Marks
Automatic
Color
Size
Text
Detail
Tooltip
Tables
Category
City
Country
Postal Code
Region
Segment
Ship Mode
State
Sub-Category
Measure Names
Discount
Profit
Quantity
Sales
Latitude (generated)
Longitude (generated)
SampleSuperstore.csv (C...
Measure Values
Sheet 1
Drop field here
Drop field here
Drop field here
Data Source
Sheet 1
Type here to search
SBIN -1.39%
1:40 PM
2/6/2025

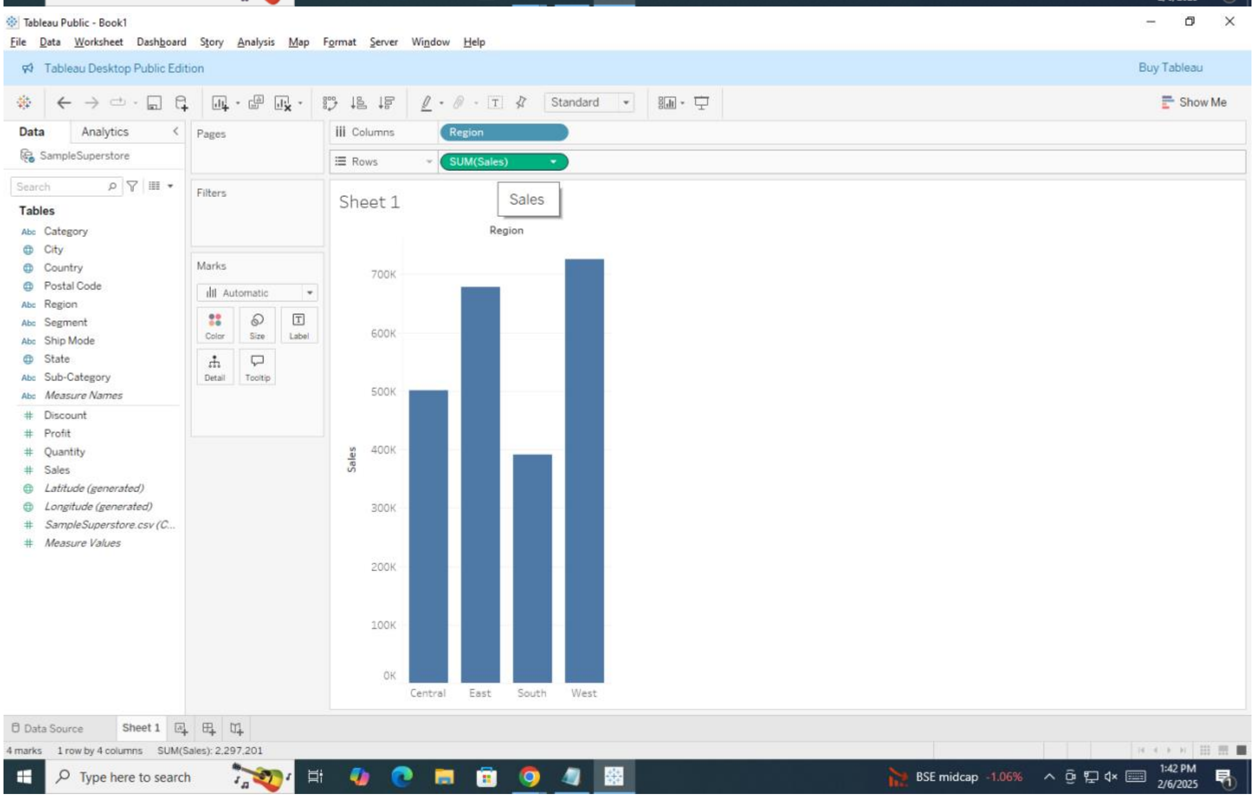


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Standard
Show Me
Data
Analytics
SampleSuperstore
Pages
Columns
Region
Rows
SUM(Sales)
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Tables
Category
City
Country
Postal Code
Region
Segment
Ship Mode
State
Sub-Category
Measure Names
Discount
Profit
Quantity
Sales
Latitude (generated)
Longitude (generated)
SampleSuperstore.csv (C...
Measure Values
Sheet 1
Sales
Region
700K
600K
500K
400K
300K
200K
100K
0K
Sales
Central
East
South
West
Data Source
Sheet 1
4 marks
1 row by 4 columns
SUM(Sales): 2,297,201
Type here to search
BSE midcap -1.06%
1:42 PM
2/6/2025

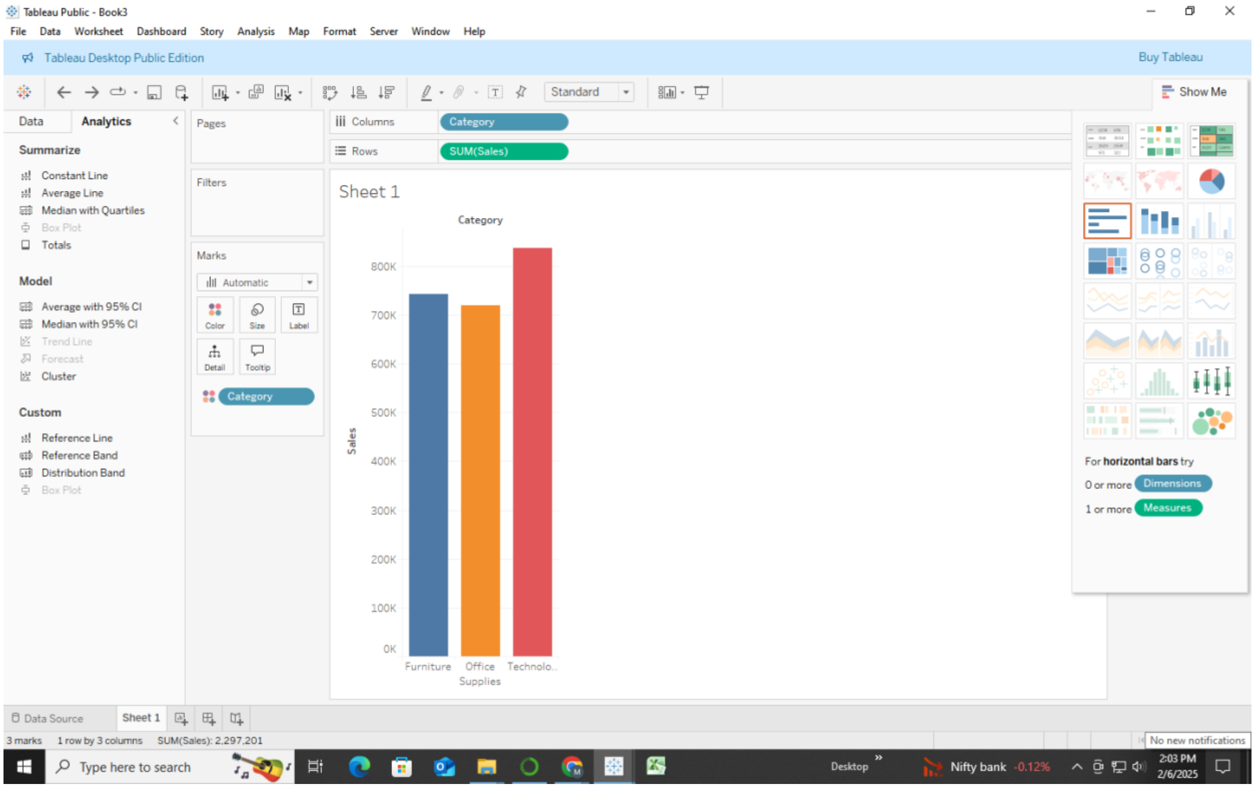


Tableau Public - Book3
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Standard
Show Me
Data
Analytics
Summarize
Constant Line
Average Line
Median with Quartiles
Box Plot
Totals
Model
Average with 95% CI
Median with 95% CI
Trend Line
Forecast
Cluster
Custom
Reference Line
Reference Band
Distribution Band
Box Plot
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Category
Columns
Category
Rows
SUM(Sales)
Sheet 1
Category
800K
700K
600K
500K
400K
300K
200K
100K
0K
Sales
Furniture
Office Supplies
Technolo..
For horizontal bars try
0 or more Dimensions
1 or more Measures
Data Source
Sheet 1
3 marks
1 row by 3 columns
SUM(Sales): 2,297,201
No new notifications
Type here to search
Desktop
Nifty bank -0.12%
2:03 PM
2/6/2025

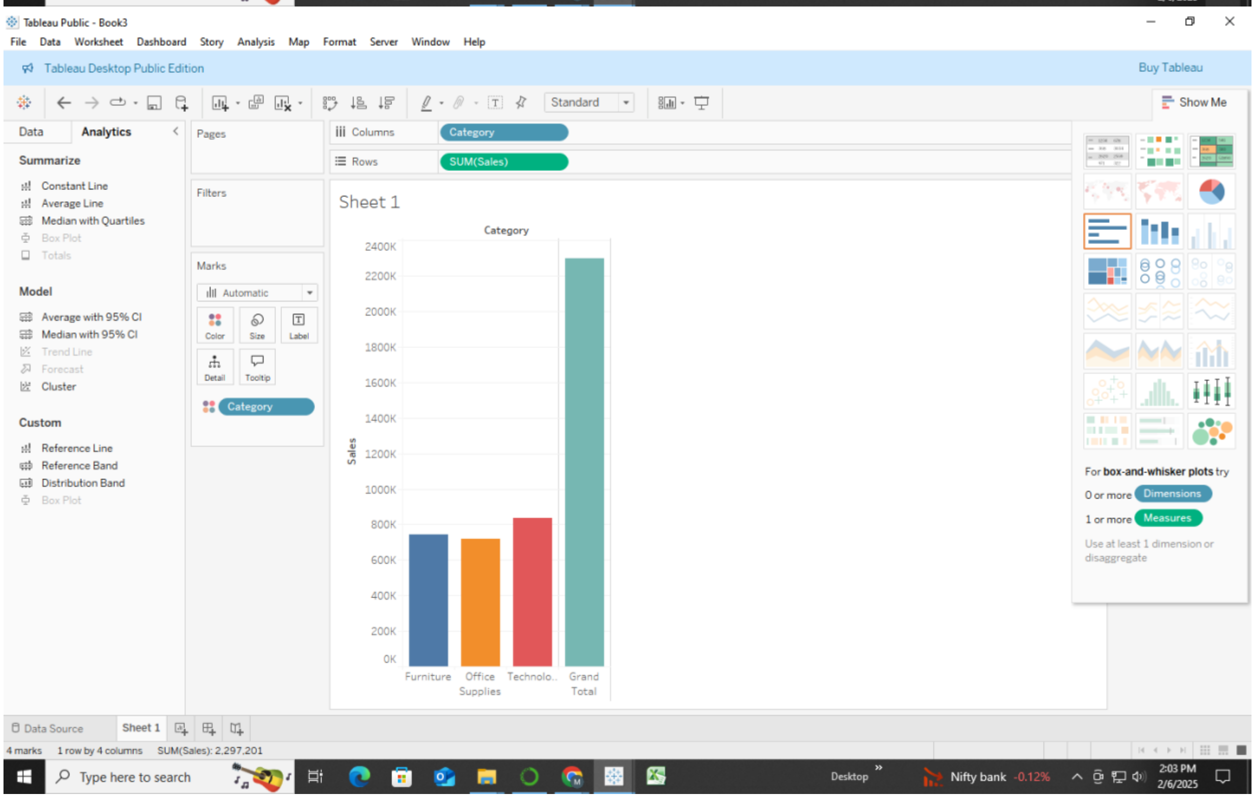


Tableau Public - Book3
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Standard
Show Me
Data
Analytics
Summarize
Constant Line
Average Line
Median with Quartiles
Box Plot
Totals
Model
Average with 95% CI
Median with 95% CI
Trend Line
Forecast
Cluster
Custom
Reference Line
Reference Band
Distribution Band
Box Plot
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Category
Columns
Category
Rows
SUM(Sales)
Sheet 1
Category
2400K
2200K
2000K
1800K
1600K
1400K
1200K
1000K
800K
600K
400K
200K
0K
Sales
Furniture
Office Supplies
Technolo..
Grand Total
For box-and-whisker plots try
0 or more Dimensions
1 or more Measures
Use at least 1 dimension or disaggregate
Data Source
Sheet 1
4 marks
1 row by 4 columns
SUM(Sales): 2,297,201
Type here to search
Desktop
Nifty bank -0.12%
2:03 PM
2/6/2025

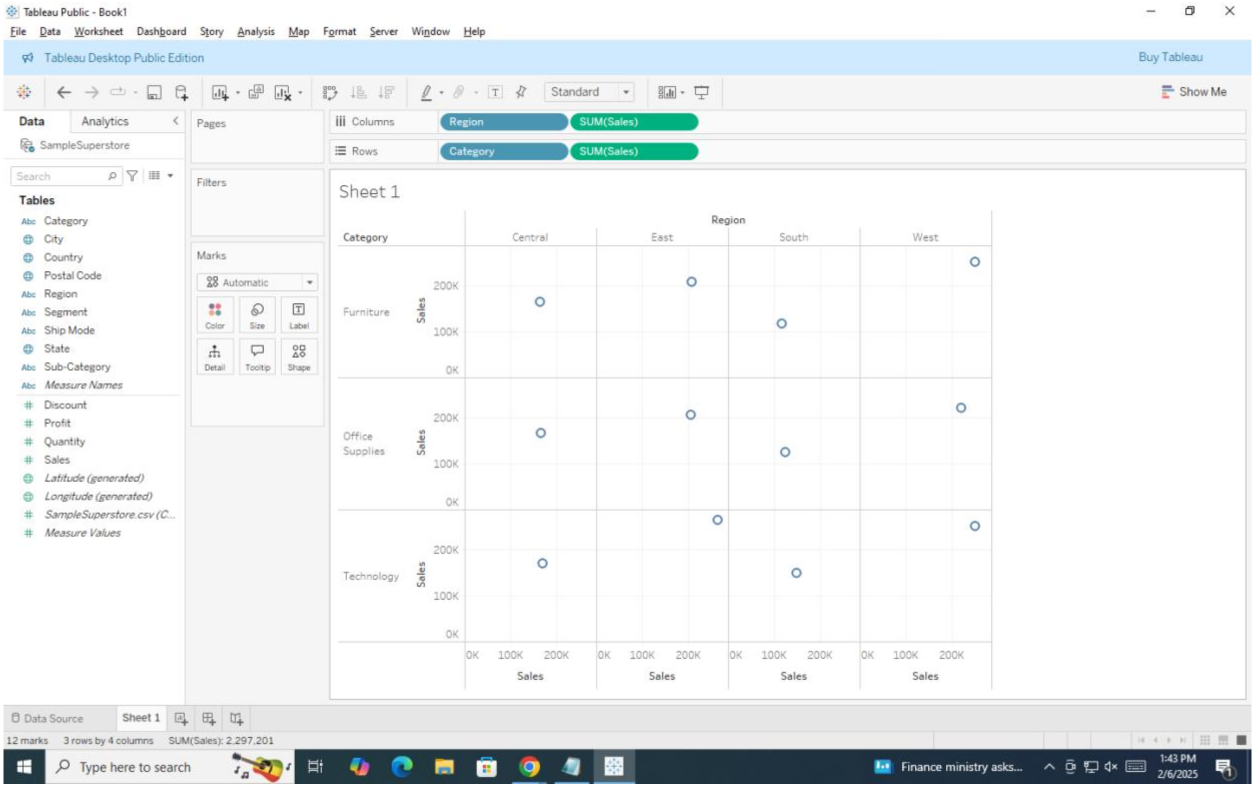


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Data Analytics
SampleSuperstore
Pages
Columns Region SUM(Sales)
Rows Category SUM(Sales)
Filters
Marks
Automatic
Color Size Label Detail Tooltip Shape
Tables
Category City Country Postal Code Region Segment Ship Mode State Sub-Category Measure Names
Discount Profit Quantity Sales Latitude (generated) Longitude (generated) SampleSuperstore.csv (C... Measure Values
Sheet 1
Region
Category
Central East South West
Furniture
Office Supplies
Technology
Sales
0K 100K 200K
Data Source Sheet 1
12 marks 3 rows by 4 columns SUM(Sales): 2,297,201
Type here to search
Finance ministry asks...
1:43 PM 2/6/2025

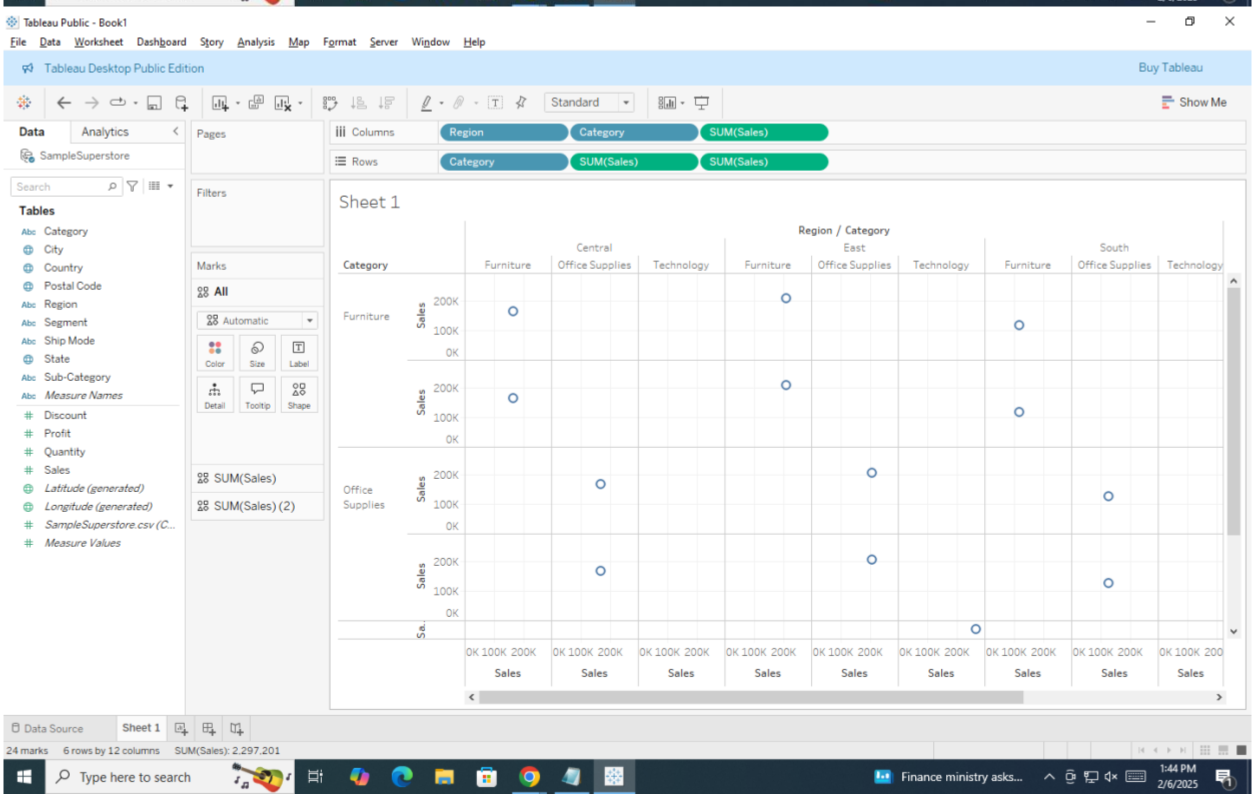
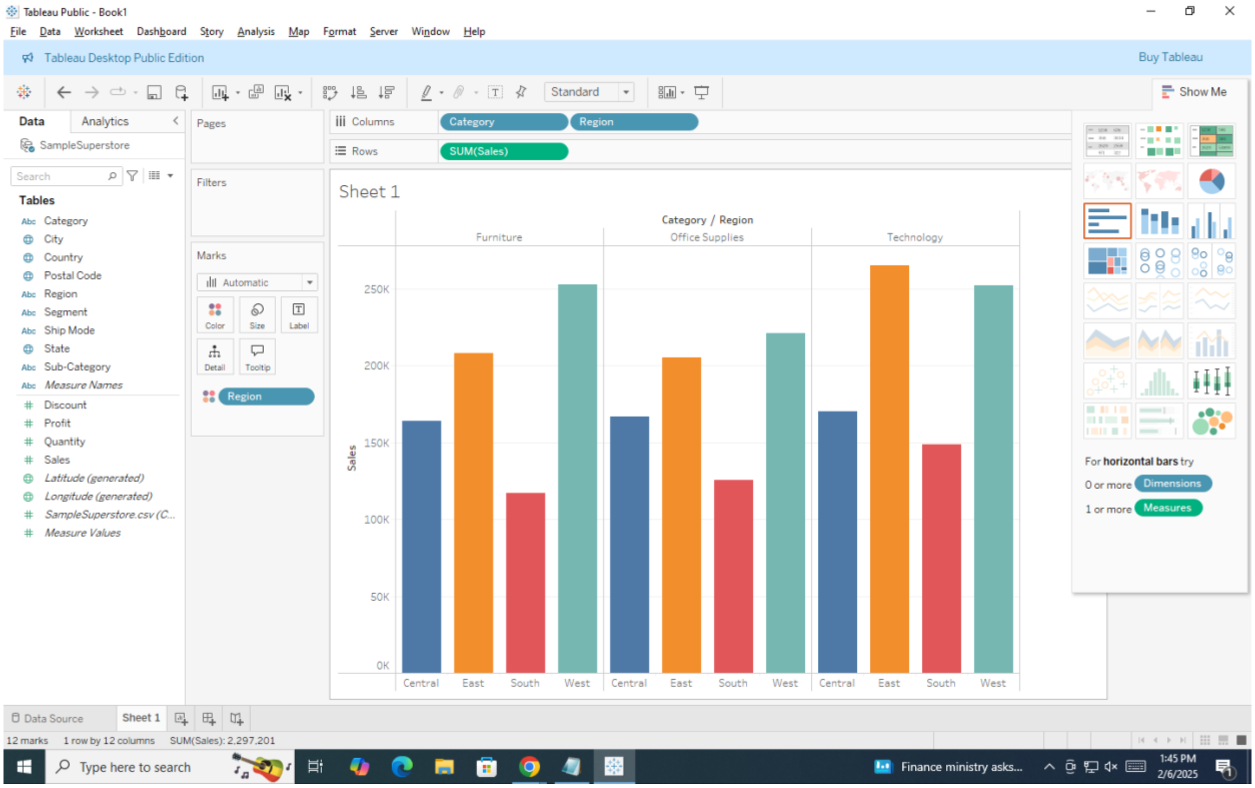
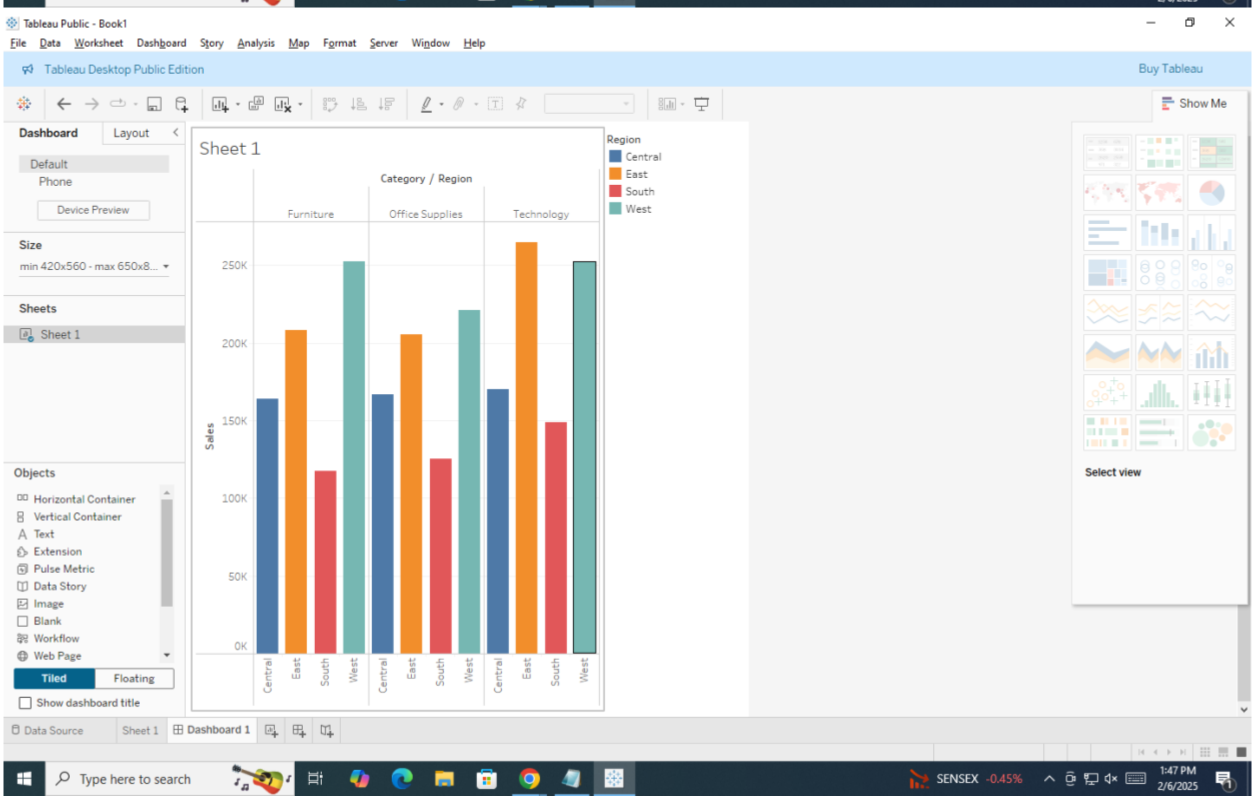


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Data Analytics
SampleSuperstore
Pages
Columns Region Category SUM(Sales)
Rows Category SUM(Sales) SUM(Sales)
Filters
Marks
All
Automatic
Color Size Label Detail Tooltip Shape
SUM(Sales)
SUM(Sales) (2)
Tables
Category City Country Postal Code Region Segment Ship Mode State Sub-Category Measure Names
Discount Profit Quantity Sales Latitude (generated) Longitude (generated) SampleSuperstore.csv (C... Measure Values
Sheet 1
Region / Category
Central East South
Category
Furniture Office Supplies Technology
Furniture
Office Supplies
Sales
0K 100K 200K
Data Source Sheet 1
24 marks 6 rows by 12 columns SUM(Sales): 2,297,201
Type here to search
Finance ministry asks...
1:44 PM 2/6/2025





PRACTICAL 9

AIM: To Implement Arima Model For Time Series Forecasting.

CONCEPT:

- ARIMA is one of the most widely used approaches to time series forecasting and it can be used in two different ways depending on the type of time series data that you're working with.
- In the first case, we have create a Non-seasonal ARIMA model that doesn't require accounting for seasonality in your time series data.
- We predict the future simply based on patterns in the past data. In the second case, we account for seasonality which is regular cycles that affect the time series.
- These cycles can be daily, weekly, or monthly and they help define patterns in the past data of the time series that can be used to forecast future values.
- Like much of data science, the foundation of forecasting is having good time series data with which to train your models.
- A time series is an ordered sequence of measurements of a variable at equally spaced time intervals.

LIBRARIES USED:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Statsmodel
- ARIMA Model
- Sklearn

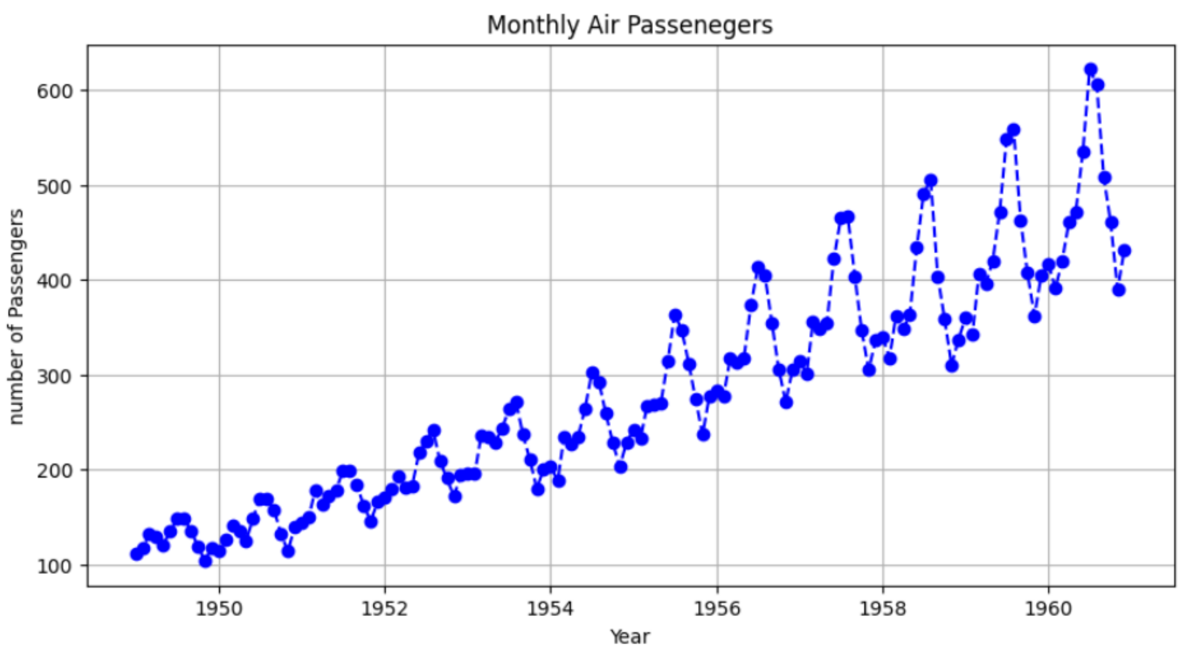
CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
```

```
from pmdarima import auto_arima
from sklearn.metrics import mean_absolute_error
import warnings
warnings.filterwarnings("ignore")
df=pd.read_csv("https://raw.githubusercontent.com/jbrownlee/Datasets/master/airli
ne-passengers.csv", parse_dates=['Month'], index_col='Month')
df.columns = ['Passengers']
plt.figure(figsize=(10,5))
plt.plot(df,color='blue',marker='o',linestyle='dashed')
plt.title('Monthly Air Passenegers')
plt.xlabel('Year')
plt.ylabel('number of Passengers')
plt.grid()
plt.show()
def test_stationarity(timeseries):
  result = adfuller(timeseries)
  print('ADF Statistic: %f' % result[0])
  print('p-value: %f' % result[1])
  if result[1] <= 0.05:
    print('Timeseries is Stationary')
  else:
    print('Timeseries is Non-Stationary')
test_stationarity(df['Passengers'])
df['Passengers_diff2'] = df['Passengers'] - df['Passengers'].shift(2)
df.dropna(inplace=True)
test_stationarity(df['Passengers_diff2'])
best_model =
auto_arima(df['Passengers'],seasonal=True,m=12,trace=True,stepwise=True,suppr
ess_warnings=True)
print(best_model.summary())
optimal_order = best_model.order
optimal_seasonal_order = best_model.seasonal_order
model =
ARIMA(df['Passengers'],order=optimal_order,seasonal_order=optimal_seasonal_o
rder)
```

```
model_fit = model.fit()
print(model_fit.summary())
predictions = model_fit.forecast(steps=12)
plt.figure(figsize=(10,5))
plt.plot(df.index, df['Passengers'], label='Actual', color='blue')
plt.plot(pd.date_range(df.index[-1], periods=13, freq='M')[1:], predictions,
label='Forecast', color='red')
plt.title('Optimized ARIMA Forecast')
plt.xlabel('Year')
plt.ylabel('Number of Passengers')
plt.legend()
plt.show()
mae = mean_absolute_error(df['Passengers'][-12:], predictions[:12])
print(f'Optimized Mean Absolute Error: {mae}')
```

OUTPUT:




```
ADF Statistic: 0.815369
p-value: 0.991880
Timeseries is Non-Stationary
```

```
ADF Statistic: -2.961695
p-value: 0.038630
Timeseries is Stationary
```

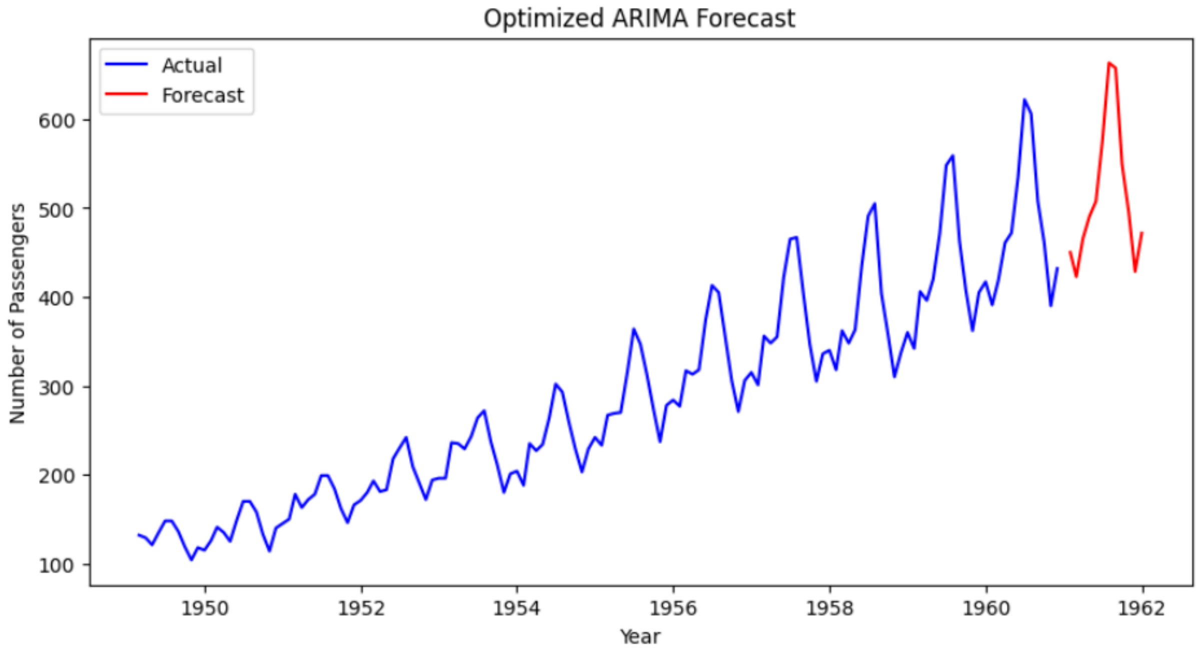
```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,1,1)[12]             : AIC=inf, Time=1.86 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1017.606, Time=0.04 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1006.696, Time=0.14 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=1007.313, Time=0.22 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=1006.660, Time=0.07 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=1007.212, Time=0.19 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=1006.521, Time=0.54 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=4.43 sec
 ARIMA(1,1,0)(1,1,2)[12]             : AIC=1000.994, Time=2.51 sec
 ARIMA(1,1,0)(0,1,2)[12]             : AIC=1005.827, Time=0.69 sec
 ARIMA(1,1,0)(2,1,2)[12]             : AIC=inf, Time=3.33 sec
 ARIMA(0,1,0)(1,1,2)[12]             : AIC=1020.207, Time=0.77 sec
 ARIMA(2,1,0)(1,1,2)[12]             : AIC=1001.896, Time=3.26 sec
 ARIMA(1,1,1)(1,1,2)[12]             : AIC=1001.428, Time=3.74 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=999.436, Time=1.93 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=1005.838, Time=0.78 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=inf, Time=0.67 sec
 ARIMA(0,1,1)(2,1,2)[12]             : AIC=inf, Time=3.84 sec
 ARIMA(0,1,1)(2,1,1)[12]             : AIC=inf, Time=4.24 sec
 ARIMA(0,1,2)(1,1,2)[12]             : AIC=1001.432, Time=2.19 sec
 ARIMA(1,1,2)(1,1,2)[12]             : AIC=inf, Time=3.73 sec
 ARIMA(0,1,1)(1,1,2)[12] intercept   : AIC=1001.383, Time=2.53 sec
```

```
Best model:  ARIMA(0,1,1)(1,1,2)[12]
Total fit time: 41.815 seconds
                                      SARIMAX Results
============================================================================================
Dep. Variable:                                  y   No. Observations:                  142
Model:             SARIMAX(0, 1, 1)x(1, 1, [1, 2], 12)   Log Likelihood            -494.718
Date:                            Thu, 13 Feb 2025   AIC                            999.436
Time:                                    07:41:08   BIC                           1013.735
Sample:                                03-01-1949   HQIC                          1005.246
                                     - 12-01-1960
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.4267      0.064     -6.638      0.000      -0.553      -0.301
ar.S.L12       0.9757      0.086     11.398      0.000       0.808       1.143
ma.S.L12      -1.2942      0.237     -5.469      0.000      -1.758      -0.830
ma.S.L24       0.3951      0.132      2.985      0.003       0.136       0.655
sigma2       114.3814     16.374      6.985      0.000      82.289     146.474
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 7.83
Prob(Q):                              0.95   Prob(JB):                         0.02
Heteroskedasticity (H):               2.83   Skew:                             0.08
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.20
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
                                      SARIMAX Results
============================================================================================
Dep. Variable:                         Passengers   No. Observations:                  142
Model:             ARIMA(0, 1, 1)x(1, 1, [1, 2], 12)   Log Likelihood              -494.718
Date:                            Thu, 13 Feb 2025   AIC                            999.436
Time:                                    07:48:35   BIC                           1013.735
Sample:                                03-01-1949   HQIC                          1005.246
                                     - 12-01-1960
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.4267      0.064     -6.638      0.000      -0.553      -0.301
ar.S.L12       0.9757      0.086     11.398      0.000       0.808       1.143
ma.S.L12      -1.2942      0.237     -5.469      0.000      -1.758      -0.830
ma.S.L24       0.3951      0.132      2.985      0.003       0.136       0.655
sigma2       114.3814     16.374      6.985      0.000      82.289     146.474
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 7.83
Prob(Q):                              0.95   Prob(JB):                         0.02
Heteroskedasticity (H):               2.83   Skew:                             0.08
Prob(H) (two-sided):                  0.00   Kurtosis:                         4.20
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```




```
Optimized Mean Absolute Error: 38.62313865694666
```

PRACTICAL 10

AIM: To implement word count using map-reduce technique in big data analytics.

CONCEPT:

- In Big Data Analytics (BDA), "MapReduce" is a programming model used to process large datasets in parallel across a cluster of computers, where data is split into smaller chunks, processed individually in the "Map" phase, and then combined in the "Reduce" phase to produce a final result, typically utilizing a distributed file system like Hadoop Distributed File System (HDFS) for storage and efficient processing.
- The core strength of MapReduce is its ability to distribute data processing across multiple nodes in a cluster, significantly speeding up computations on large datasets.
- In this step, the input data is divided into smaller parts, and each part is processed independently by a "mapper" function which generates key-value pairs based on the specific operation needed.
- After the Map phase, the key-value pairs are shuffled and sorted based on their keys to ensure that data with the same key is sent to the same reducer.

LIBRARIES USED:

- Regular Expression

CODE:

```
import re
import os
def read_file(filename):
  with open(filename,"r", encoding="utf-8") as file:
    return file.readlines()
def mapper(lines):
  word_list = []
  for line in lines:
    words =re.findall(r'\b\w+\b', line.lower())
    word_list.extend([(word, 1) for word in words])
  return word_list
```

```
def shuffle_sort(mapped_data):
  word_dict = {}
  for word, count in mapped_data:
    if word in word_dict:
      word_dict[word].append(count)
    else:
      word_dict[word] = [count]
    return word_dict
def reducer(shuffled_data):
  return {word: sum(counts) for word, counts in shuffled_data.items()}
def sequential_map_reduce(lines):
  print("Staring map...")
  mapped_results = mapper(lines)
  print(f"Mapped results: {len(mapped_results)} items")
  shuffled_data = shuffle_sort(mapped_results)
  print(f"Shuffled data: {len(shuffled_data)} words")
  reduced_data = reducer(shuffled_data)
  print(f"Reduced data: {len(reduced_data)} unique words")
  return reduced_data
if os.path.exists('/content/war_and_peace.txt'):
   print("File found!")
   text_lines = read_file('/content/war_and_peace.txt')
   print(f"TotalLines read: {len(text_lines)}")
   word_count_result = sequential_map_reduce(text_lines[:1000])
   sorted_results = sorted(word_count_result.items(), key = lambda x: x[1],
reverse=True)
   for word, count in sorted_results[:10]:
        print(f"{word}: {count}")
else:
   print("File not found. Please check the file path.")
```

OUTPUT:

```
File found!
TotalLines read: 66032
Staring map...
Mapped results: 2430 items
Shuffled data: 1 words
Reduced data: 1 unique words
the: 1
```