DV QP Sessional Answers

1.a) What is Data Visualization? Differentiate between Descriptive and Inferential Statistics.

Ans.1.a)

Data Visualization

- **Data visualization is actually a set of data points and information that are represented graphically to make it easy and quick for user to understand.**
- **Data visualization is good if it has a clear meaning, purpose, and is very easy to interpret, without requiring context.**
- **Tools of data visualization provide an accessible way to see and understand trends, outliers, and patterns in data by using visual effects or elements such as a chart, graphs, and maps.**

Characteristics of Effective Graphical Visual:

- **It shows or visualizes data very clearly in an understandable manner.**
- **It encourages viewers to compare different pieces of data.**
- **It closely integrates statistical and verbal descriptions of data sets.**
- **It grabs our interest, focuses our mind, and keeps our eyes on message as the human brain tends to focus on visual data more than written data.**
- **It also helps in identifying area that need more attention and improvement.**
- **Using graphical representation, a story can be told more efficiently. Also, it requires less time to understand picture than it takes to understand textual data.**

Categories of Data Visualization :

Data visualization is very critical to market research where both numerical and categorical data can be visualized that helps in an increase in impacts of insights and helps in reducing risk of analysis paralysis.

So, data visualization is categorized into following categories:

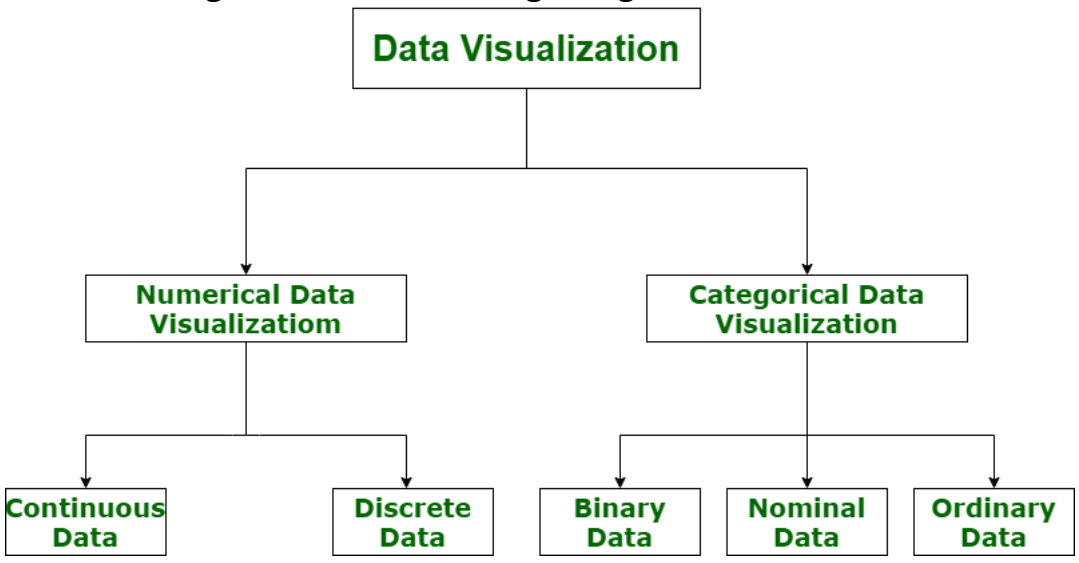


Figure – Categories of Data Visualization

1. Numerical Data :

Numerical data is also known as Quantitative data. Numerical data is any data where data generally represents amount such as height, weight, age of a person, etc. Numerical data visualization is easiest way to visualize data. It is generally used for helping others to digest large data sets and raw numbers in a way that makes it easier to interpret into action. Numerical data is categorized into two categories:

- **Continuous Data –**
  It can be narrowed or categorized (Example: Height measurements).
- **Discrete Data –**
  This type of data is not "continuous" (Example: Number of cars or children's a household has).

The type of visualization techniques that are used to represent numerical data visualization is Charts and Numerical Values. Examples are Pie Charts, Bar Charts, Averages, Scorecards, etc.

2. Categorical Data :

Categorical data is also known as Qualitative data. Categorical data is any data where data generally represents groups. It simply consists of categorical variables that are used to represent characteristics such as a person's ranking, a person's gender, etc. Categorical data visualization is all about depicting key themes, establishing connections, and lending context. Categorical data is classified into three categories:

- **Binary Data –**
  In this, classification is based on positioning (Example: Agrees or Disagrees).
- **Nominal Data –**
  In this, classification is based on attributes (Example: Male or Female).
- **Ordinal Data –**
  In this, classification is based on ordering of information (Example: Timeline or processes).

The type of visualization techniques that are used to represent categorical data is Graphics, Diagrams, and Flowcharts. Examples are Word clouds, Sentiment Mapping, Venn Diagram, etc.

S.No.	Descriptive Statistics	Inferential Statistics
1.	It gives information about raw data which describes the data in some manner.	It makes inferences about the population using data drawn from the population.
2.	It helps in organizing, analyzing, and to present data in a meaningful manner.	It allows us to compare data and make hypotheses and predictions.
3.	It is used to describe a situation.	It is used to explain the chance of occurrence of an event.
4.	It explains already known data and is limited to a sample or population having a small size.	It attempts to reach the conclusion about the population.
5.	It can be achieved with the help of charts, graphs, tables, etc.	It can be achieved by probability.

1.b) Explain data types of the R-object with example.

Ans.1.b)

There are 5 basic data types of objects in the R language:

1. **Vectors**
   Atomic vectors are one of the basic types of objects in R programming. Atomic vectors can store homogeneous data types such as character, doubles, integers, raw, logical, and complex. A single element variable is also said to be vector.

2. **Lists**
   List is another type of object in R programming. List can contain heterogeneous data types such as vectors or another lists.

3. **Matrices**
   To store values as 2-Dimensional array, matrices are used in R. Data, number of rows and columns are defined in the matrix() function.

4. **Factors**
   Factor object encodes a vector of unique elements (levels) from the given data vector.

5. **Arrays**
   array() function is used to create n-dimensional array. This function takes dim attribute as an argument and creates required length of each dimension as specified in the attribute.

2.a) Explain the statement of Hypothesis in detail.

Ans.2.a)

Statement of Hypothesis

A statistical hypothesis is defined as a statement, which may or may not be true about the population parameter or about the probability distribution of the parameter that we wish to validate on the basis of sample information. Most times, experiments are performed with random samples instead of the entire population and inferences drawn from the observed results are then generalised over to the entire population. But before drawing inferences about the population, it should be always kept in mind that the observed results might have come due to chance factor. In order to have an accurate or more precise inference, the chance factor should be ruled out.

Null Hypothesis

The probability of chance occurrence of the observed results is examined by the null hypothesis (H0). Null hypothesis is a statement of no differences. The other way to state null hypothesis is that the two samples came from the same population. Here, we assume that population is normally distributed, and both the groups have equal means and standard deviations. Since the null hypothesis is a testable proposition, there is counter proposition to it known as alternative hypothesis and denoted by H1 . In contrast to null hypothesis, the alternative hypothesis (H1) proposes that

i) the two samples belong to two different populations,

ii) their means are estimates of two different parametric means of the respective population,

iii) there is a significant difference between their sample means.

2.b) Explain in short the following term.

i) Random Variables ii) Normal Probability Distribution

Ans.2.b)

i) Random Variables

A random variable, X, is a variable whose possible values are numerical outcomes of a random phenomenon. There are two types of random variables, discrete and continuous.

Example of Random variable

- A person's blood type
- Number of leaves on a tree
- Number of times a user visits LinkedIn in a day
- Length of a tweet.

Types of Random Variables

a. Discrete Random Variables :

A discrete random variable is one which may take on only a countable number of distinct values such as 0,1,2,3,4,..... Discrete random variables are usually counts. If a random variable can take only a finite number of distinct values, then it must be discrete. Examples of discrete random variables include the number of children in a family, the Friday night attendance at a cinema, the number of patients in a doctor's surgery, the number of defective light bulbs in a box of ten.

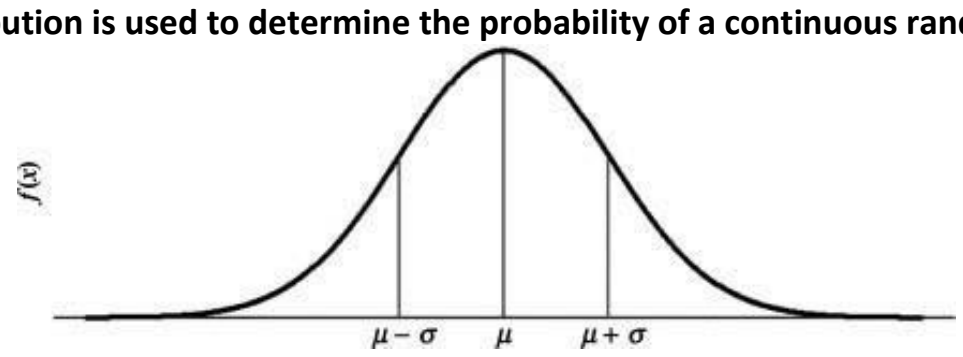
b. Continuous Random Variables:

A continuous random variable is one which takes an infinite number of possible values. Continuous random variables are usually measurements. Examples include height, weight, the amount of sugar in an orange, the time required to run a mile. A continuous random variable is not defined at specific values. Instead, it is defined over an interval of values, and is represented by the area under a curve (known as an integral). The probability of observing any single value is equal to 0, since the number of values which may be assumed by the random variable is infinite.

ii) Normal Probability Distribution

The Bell-Shaped Curve

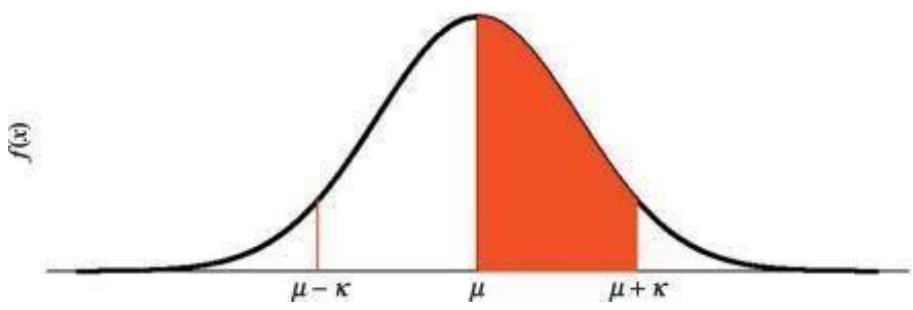
The Bell-shaped Curve is commonly called the normal curve and is mathematically referred to as the Gaussian probability distribution. Unlike Bernoulli trials which are based on discrete counts, the normal distribution is used to determine the probability of a continuous random variable.



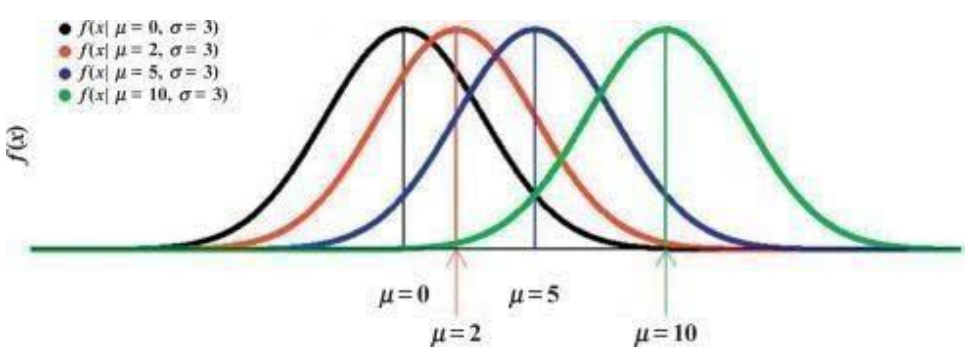
The normal or Gaussian Probability Distribution is most popular and important because of its unique mathematical properties which facilitate its application to practically any physical problem in the real world. The constants μ and σ^2 are the parameters:

- **"μ" is the population true mean (or expected value) of the subject phenomenon characterized by the continuous random variable, X,**
- **"σ^2" is the population true variance characterized by the continuous random variable, X.**
- **Hence, "σ" the population standard deviation characterized by the continuous random variable X;**
- **The points located at μ−σ and μ+σ are the points of inflection; that is, where the graph changes from cupping up to cupping down.**

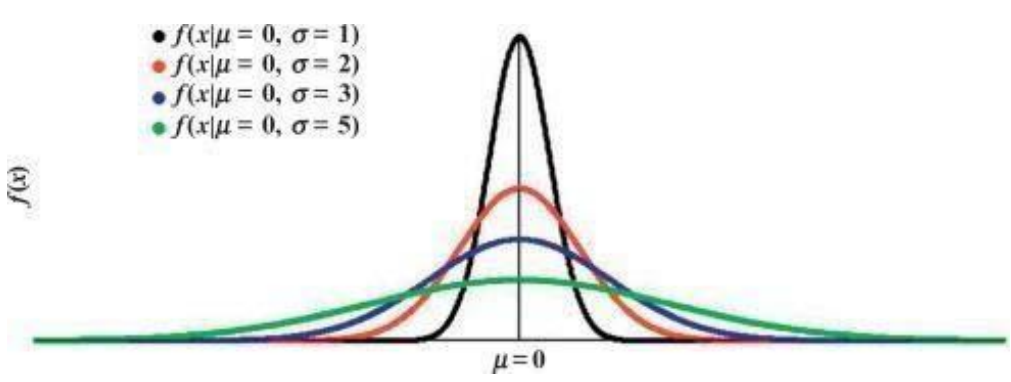
The normal curve graph of the normal probability distribution) is symmetric with respect to the mean μ as the central position. That is, the area between μ and κ units to the left of μ is equal to the area between μ and κ units to the right of μ.



There is not a unique normal probability distribution. The figure below is a graphical representation of the normal distribution for a fixed value of σ2 with μ varying.



The figure below is a graphical representation of the normal distribution for a fixed value of μ with varying σ2 .



3.a) Explain Scatter Plot ? Also explain the advantage and limitation of scatter plot.

Ans.3.a)

Scatter Plot

A Scatter Plot is a graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present. With scatter plots we can explain how the variables relate to each other. Which is defined as correlation. Positive, Negative, and None (no correlation) are the three types of correlation.

Advantages of a Scatter Plot

- **Relationship between two variables can be viewed.**
- **For non-linear pattern, this is the best method.**
- **Maximum and minimum value can be easily determined.**
- **Observation and reading are easy to understand**
- **Plotting the diagram is very simple**

Limitations of a Scatter Plot

- **With Scatter diagrams we cannot get the exact extent of correlation.**
- **Quantitative measure of the relationship between the variable cannot be viewed.**
- **Only shows the quantitative expression.**
- **The relationship can only show for two variables.**

3.b) What is data frame and how it is created in R ? Explain with suitable example.

Ans.3.b)

Data Frame

- **A Data Frame is a way to represent and work with tabular data.**
- **Tabular data has rows and columns, just like our csv file.**
- **In order to read in the data, we'll need to use the pandas.read_csv function.**
- **This function will take in a csv file and return a Dataframe.**
- **Data Frames are data displayed in a format as a table.**
- **Data Frames can have different types of data inside it.**
- **While the first column can be character, the second and third can be numeric or logical.**
- **However, each column should have the same type of data.**

Create DataFrame in R :

Use the data.frame() function to create a data frame:

Example:

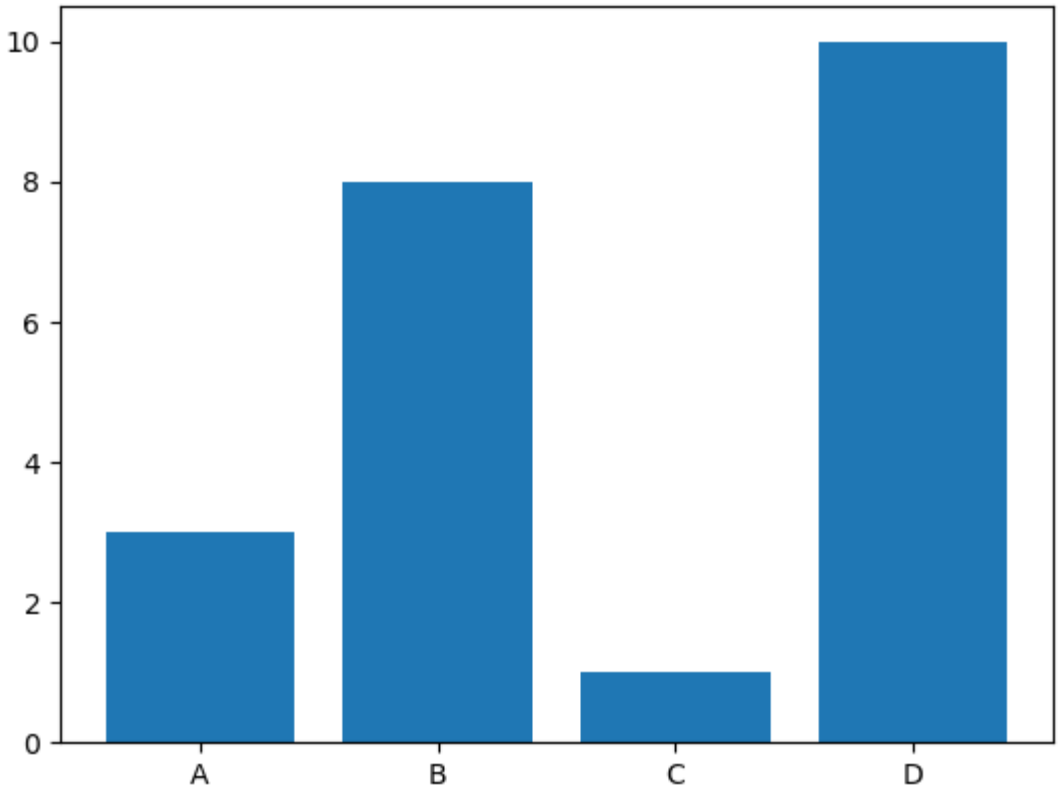
```
# Create a data frame
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

# Print the data frame
Data_Frame
```

4.a) Explain Bar plot? Write a code in R to plot bar plot by taking suitable parameters.

Ans.4.a)

Bar Plot

- **A barplot (or barchart) is one of the most common type of graphic. It shows the relationship between a numeric variable and a categoric variable.**
- **Bar Plot are classified into four types of graphs - bar graph or bar chart, line graph, pie chart, and diagram.**



Limitations of Bar Plot:

- **When we try to display changes in speeds such as acceleration, Bar graphs won't help us.**

Advantages of Bar plot:

- **Bar charts are easy to understand and interpret.**
- **Relationship between size and value helps for in easy comparison.**
- **They're simple to create.**
- **They can help in presenting very large or very small values easily.**

In R, you can create a bar plot using the ggplot2 package.
Here is an example code to plot a bar plot in R:

```
library(ggplot2)

# Create a data frame
df <- data.frame(
  category = c("A", "B", "C", "D"),
  frequency = c(10, 15, 8, 12)
)

# Plot the bar plot
ggplot(df, aes(x = category, y = frequency)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  xlab("Category") +
  ylab("Frequency") +
  ggtitle("Bar Plot")
```

In this code, we first create a data frame with two columns: "category" and "frequency". The "category" column represents the different categories, and the "frequency" column represents the frequency of each category.
Then, we use the ggplot() function to initialize the plot and specify the data frame and aesthetic mappings. We use the geom_bar() function with the stat = "identity" parameter to create the bar plot. The fill parameter sets the color of the bars.
Finally, we add labels to the x-axis, y-axis, and a title to the plot using the xlab(), ylab(), and ggtitle() functions, respectively.
Please note that you need to have the ggplot2 package installed in R to run this code.

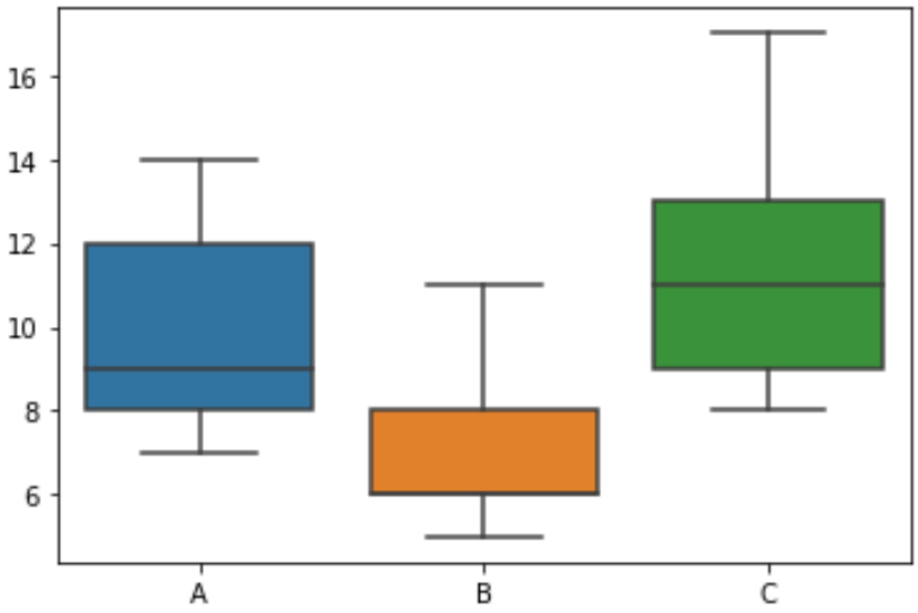
4.b) Explain Box and Histogram plot with advantage and limitation.

Ans.4.b)

Box Plot

A box plot is a way of statistically representing the distribution of the data through five main dimensions :

- **Minimum: Smallest number in the dataset.**
- **First quartile: Middle number between the minimum and the median.**
- **Second quartile (Median): Middle number of the (sorted) dataset.**
- **Third quartile: Middle number between median and maximum.**
- **Maximum: Highest number in the dataset.**



Advantages:

- **The box plot organizes large amounts of data and visualizes outlier values.**

Disadvantages:

- **The box plot is not relevant for detailed analysis of the data as it deals with a summary of the data distribution.**

Histogram

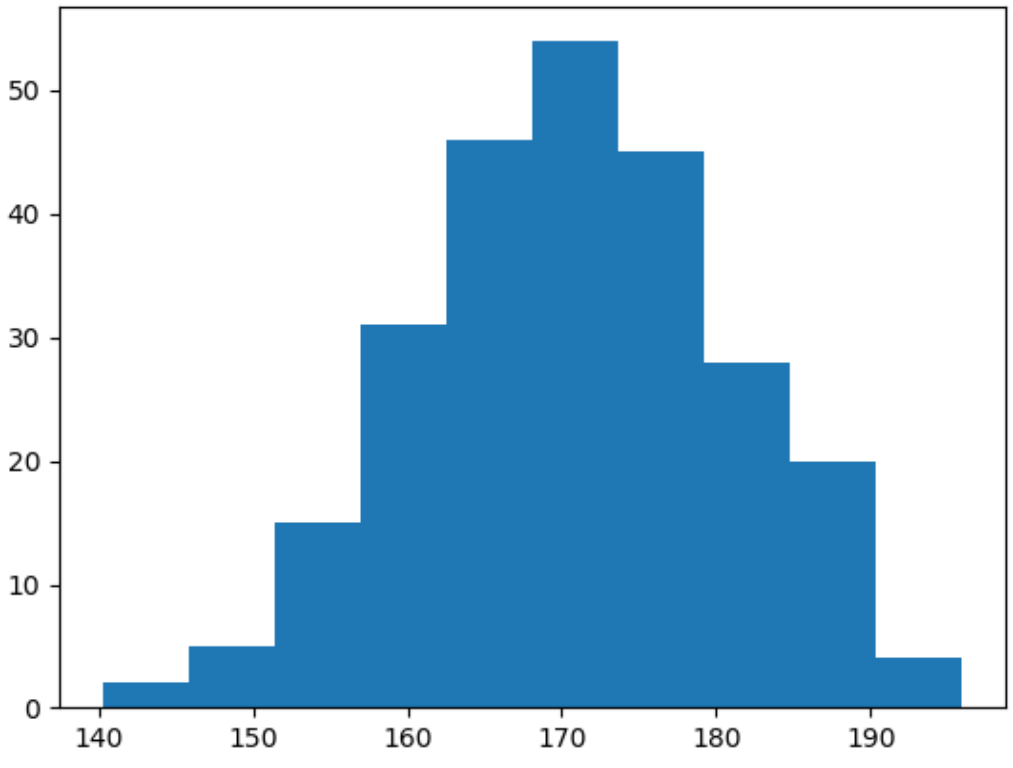
A histogram represents the frequency distribution of continuous variables. While, a bar graph is a diagrammatic comparison of discrete variables. Histogram presents numerical data whereas bar graph shows categorical data. The histogram is drawn in such a way that there is no gap between the bars.

Limitations of Histogram:

- **A histogram can present data that is misleading as it has many bars.**
- **Only two sets of data are used, but to analyze certain types of statistical data, more than**
- **two sets of data are necessary**

Advantages of Histogram:

- **Histogram helps to identify different data, the frequency of the data occurring in the dataset and categories which are difficult to interpret in a tabular form.**
- **It helps to visualize the distribution of the data.**



5.a) What is Conditional Statements? Explain with example.

Ans.5.a)

Conditional statements, also known as decision-making statements, are used in programming languages to control the flow of execution based on certain conditions. These statements allow us to execute a block of code when a given condition is true or false.

In Python, we have several types of conditional statements:

1. If statements: The if statement is the most commonly used conditional statement. It checks for a given condition and executes a block of code only if the condition is true.

Example:

```
name = "Srikar"
if name == "Srikar":
   print("Hello, Srikar!")
```

2. If-else statements: The if-else statement executes a block of code inside the if block if the condition is true, and executes a block of code inside the else block if the condition is false.

Example:

```
name = "John"
if name == "Srikar":
   print("Hello, Srikar!")
else:
   print("Hello, stranger!")
```

3. Elif statements: The elif statement is used to check multiple conditions only if the previous if condition is false. It is similar to an if-else statement, but it allows us to evaluate multiple conditions.

Example:

```
name = "Jane"
if name == "Srikar":
   print("Hello, Srikar!")
elif name == "John":
   print("Hello, John!")
else:
   print("Hello, stranger!")
```

4. Nested if-else statements: Nested if-else statements refer to the situation where an if statement or if-else statement is present inside another if or if-else block. This feature allows us to check multiple conditions in a program. In Python, we can have an if statement inside another if statement, which can be inside another if statement, and so on.

Example:

```
name = "Jane"
if name == "Srikar":
   print("Hello, Srikar!")
elif name == "John":
   print("Hello, John!")
else:
   print("Hello, stranger!")
```

These conditional statements help us make decisions in our code based on certain conditions, allowing for more dynamic and flexible programming.

5.b) Write a code in python with numpy module to create an array and find the sum of all elements.

Ans.5.b)

Program:

```
# numpy array - sum of all elements
import numpy as np

# Create a numpy array
arr = np.array([1, 2, 3, 4, 5])

# Find the sum of all elements in the array
sum_of_elements = np.sum(arr)

print("Array:", arr)
print("Sum of elements:", sum_of_elements)
```

Output:

```
Array: [1 2 3 4 5]
Sum of elements: 15
```

6.a) What is dictionary? Explain the methods available in dictionary.

Ans.6.a)

- **Dictionaries are used to store data values in key:value pairs.**
- **A dictionary is a collection which is ordered*, changeable and do not allow duplicates.**
- **Dictionaries are written with curly brackets and have keys and values.**

The methods available in dictionary are

Functions Name	Descriptions
clear()	Removes all items from the dictionary
copy()	Returns a shallow copy of the dictionary
fromkeys()	Creates a dictionary from the given sequence
get()	Returns the value for the given key
items()	Return the list with all dictionary keys with values
keys()	Returns a view object that displays a list of all the keys in the dictionary in order of insertion
pop()	Returns and removes the element with the given key
popitem()	Returns and removes the key-value pair from the dictionary
setdefault()	Returns the value of a key if the key is in the dictionary else inserts the key with a value to the dictionary
values()	Updates the dictionary with the elements from another dictionary
update()	Returns a list of all the values available in a given dictionary

6.b) Write a code in python to read the csv file using pandas module.

Ans.6.b)

To read a CSV file using the pandas module in Python, you can follow these steps:

1. Import the pandas library:

```
import pandas as pd
```

2. Use the `read_csv()` function to read the CSV file:

```
df = pd.read_csv('path/to/your/file.csv')
```

Replace `'path/to/your/file.csv'` with the actual path to your CSV file.

3. Now, you can work with the data stored in the DataFrame `df`.

Here's an example of how the code would look like:

```
import pandas as pd

# Read the CSV file
df = pd.read_csv('path/to/your/file.csv')

# Perform operations on the DataFrame
# For example, print the first 5 rows
print(df.head(5))
```

Make sure to replace `'path/to/your/file.csv'` with the actual path to your CSV file.

7.a) Explain Matplotlib in detail with suitable example.

Ans.7.a)

Matplotlib is the most popular data visualization library for Python. This provides control over every aspect of an image. Matplotlib is designed to provide end users with an experience similar to MATLAB graphical plotting. Matplotlib consists of three main layers: the back-end layer, the artist layer, and the scripting layer.

The back-end layer is the lowest layer in the Matplotlib architecture. It includes three abstract interface classes: FigureCanvas, Renderer, and Event. FigureCanvas defines and covers the area on which the figure is drawn. The renderer is an object that knows how to draw on the image canvas. Events handle user input such as keyboard keys and mouse clicks.

The artist layer consists of a main object called Artist. Artist is an object that knows how to use the Renderer to place ink on the canvas. Everything visible in the Matplotlib image is an Artist example. There are two types of Artist objects. The first type is a primitive type, such as line, rectangle, circle, or text. The second type is a composite type, such as an image or axis. The top-level Artist object that contains and manages all the elements in a given graph is the figure Artist object, and the most important composite objects are the axes because this is where most of the plotting methods of the Matplotlib API are defined.

The scripting layer is a lighter interface provided by Matplotlib for everyday purposes. It is designed to simplify common tasks and to produce graphs and plots quickly and easily. This scripting layer is mainly used by scientists who are not professional programmers.

Here is an example of using Matplotlib to read a CSV file and produce a line plot:

```
import matplotlib.pyplot as plt
import pandas as pd

# Membaca file CSV
data = pd.read_csv('data.csv')

# Menghasilkan plot garis
plt.plot(data['x'], data['y'])
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot')
plt.show()
```

In this example, we use Matplotlib to read a CSV file and generate a line plot. The data was read using pandas, then a line plot was created using the plot function from Matplotlib. Axis labels and titles are also added to provide additional information on the plot. Finally, the plot is displayed using the show function.

7.b) Write a python code to generate a Line Plot with matplotlib.

Ans.7.b)

```
#matplotlib - line plot

import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a figure and axis
fig, ax = plt.subplots()

# Plot the data
ax.plot(x, y)

# Customize the plot
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_title('Line Plot')

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

This code creates a simple line plot using the `plot()` function from Matplotlib. It defines the x and y coordinates of the data points, creates a figure and axis, plots the data, and customizes the plot with labels and a title. Finally, it displays the plot using `plt.show()`.

Please note that you may need to install Matplotlib if you haven't already. You can do so by running `pip install matplotlib` in your Python environment.

8.a) List out all Specialized Visualization Tools using Matplotlib? Explain any one in detail.

Ans.8.a)

Some examples of specialized visualization tools available in Matplotlib are:

1. **Pie Charts: A circular graphic that displays numeric proportions by dividing a circle into proportional slices.**
2. **Box Plot: A box plot is a way of statistically representing the distribution of the data through five main dimensions**
3. **Scatter Plots: A plot that displays the relationship between two variables by using dots on a two-dimensional plane.**
4. **Bubble Plots: A bubble plot is a variation of the scatter plot that displays three dimensions of data (x, y, z).**
5. **Waffle Chart: A waffle chart is an interesting visualization that is normally created to display progress toward goals.**
6. **Word Clouds Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud.**

1. Pie Charts:

Purpose: Pie charts are used to represent the distribution of categorical data. Each slice of the pie represents a category, and the size of each slice corresponds to the proportion of that category in the whole.

- Matplotlib Example:

```
import matplotlib.pyplot as plt
labels = ['Category A', 'Category B', 'Category C']
sizes = [30, 45, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

2. Box Plot:

Purpose: Box plots (box-and-whisker plots) provide a visual summary of the distribution of a dataset. They show the median, quartiles, and potential outliers.

- Matplotlib Example:

```
import matplotlib.pyplot as plt
import numpy as np
data = np.random.randn(100)
plt.boxplot(data)
plt.show()
```

3. Scatter Plots:

Purpose: Scatter plots display the relationship between two continuous variables. Each point represents an observation, and the position of the point is determined by the values of the two variables.

Matplotlib Example:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(100)
y = 2 * x + np.random.randn(100)
plt.scatter(x, y)
plt.show()
```

4. Bubble Plots:

Purpose: Bubble plots extend scatter plots by introducing a third dimension. In addition to x and y, the size of each point (bubble) represents a third variable.

- Matplotlib Example:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(100)
y = 2 * x + np.random.randn(100)
size = np.random.rand(100) * 100
plt.scatter(x, y, s=size)
plt.show()
```

5. Waffle Chart:

Purpose: Waffle charts are used to represent progress toward a goal. They consist of a grid, and each cell in the grid represents a unit of the goal. Filled cells indicate completed units.

Matplotlib Example: Creating a waffle chart in Matplotlib can be a bit involved. It often involves creating a grid of subplots and selectively filling them based on the desired representation.

6. Word Clouds:

Purpose: Word clouds visually represent the frequency of words in a given text. The more frequently a word appears, the larger and bolder it appears in the cloud.

Matplotlib Example: Creating a word cloud typically involves using a specialized library like `wordcloud`. Here's a basic example:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
text = "Data visualization is a powerful tool for exploring and communicating insights from data."
wordcloud = WordCloud().generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

These examples demonstrate how Matplotlib can be used to create a diverse range of visualizations for different types of data and purposes.

8.b) Explain the following term.

i) Bubble Plots ii) Waffle Chart

Ans.8.b)

i) Raw Code

Raw Code in data visualization typically refers to the actual programming code used to create visual representations of data. In Python, the term "raw code" typically refers to the source code as it was originally written. It's the code that you write in your Python script before it's interpreted or compiled by the Python interpreter.

Here's an example of raw Python code:

```
def greet(name):
    return f"Hello, {name}!"
print(greet("World"))
```

In this raw code:

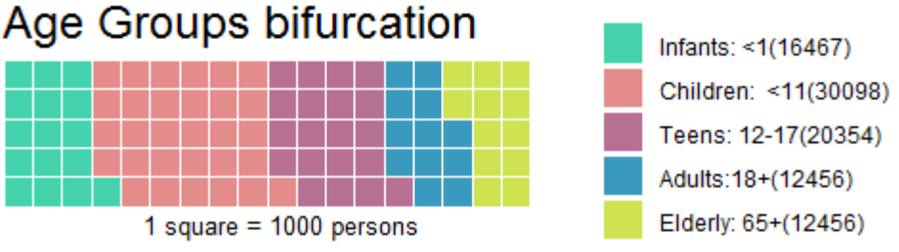
We define a function greet that takes one argument name.
The function returns a string that says "Hello, {name}!" where {name} is replaced with the argument passed to the function.
We then call this function with the argument "World" and print the result.
This raw code would output: Hello, World!

Raw Python code can include all constructs of the Python language, such as variables, functions, classes, and more. It's called "raw" because it's the code as you originally wrote it, without any changes made by interpreters or compilers.

When you run this raw code, the Python interpreter reads and executes it line by line. If the code is syntactically correct and doesn't have any runtime errors, you'll see the expected output. If there are any errors, the interpreter will stop and show an error message.

ii) Waffle Chart

A waffle chart shows progress towards a target or a completion percentage. Waffle Charts are a great way of visualizing data in relation to a whole, to highlight progress against a given threshold, or when dealing with populations too varied for pie charts. A lot of times, these are used as an alternative to the pie charts. It also has a niche for showing parts-to-whole contribution. It doesn't misrepresent or distort a data point (which a pie chart is sometimes guilty of doing).



Waffle Charts are mainly used when composing parts of a whole, or when comparing progress against a goal. These charts usually follow other kinds of data visualization for helping the understanding of the audience. For example, you might want a Waffle Chart when plotting how the expenses of a company are composed by each type of expense, or when classifying percentages of a population at a given moment. Waffle charts are also known as Squared Pie Charts. The individual values will be summed up and each that will be the total number of squares in the grid.

9.a) Explain seaborn with functionalities and usage.

Ans.9.a)

Seaborn is a statistical data visualization library in Python that is built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn comes with several built-in themes and color palettes to make it easy to create aesthetically pleasing visualizations. Here are some key functionalities and common usage patterns of Seaborn:

1. Data Visualization Functions:
- **Seaborn provides functions to create a variety of statistical visualizations, including:**
  - **Histograms and Kernel Density Plots: `sns.histplot()`, `sns.kdeplot()`**
  - **Box Plots: `sns.boxplot()`**
  - **Violin Plots: `sns.violinplot()`**
  - **Scatter Plots: `sns.scatterplot()`**
  - **Pair Plots: `sns.pairplot()`**

2. Statistical Estimation:
- **Seaborn includes functions for statistical estimation and visualization, such as:**
  - **Regression Plots: `sns.regplot()`, `sns.lmplot()`**
  - **Correlation Heatmaps: `sns.heatmap()`**
  - **Residual Plots: `sns.residplot()`**

3. Categorical Plots:
- **Seaborn excels at visualizing categorical data with functions like:**
  - **Bar Plots: `sns.barplot()`**
  - **Count Plots: `sns.countplot()`**
  - **Categorical Scatter Plots: `sns.stripplot()`, `sns.swarmplot()`**

4. Themes and Aesthetics:
- **Seaborn makes it easy to control the aesthetics of your plots with themes and color palettes. You can use functions like `sns.set_theme()` to set the overall appearance of your plots, and `sns.color_palette()` to choose custom color palettes.**

5. Faceting:
- **Seaborn supports faceting, which involves creating multiple plots that display different subsets of the data. The `sns.FacetGrid` class allows you to create a grid of subplots based on the values of one or more categorical variables.**

6. Distribution Plots:
- **Seaborn includes functions to visualize the distribution of a single variable, such as:**
  - **Distplot: `sns.distplot()`**
  - **Rug Plots: `sns.rugplot()`**

7. Matrix Plots:
- **For visualizing matrices of data, Seaborn provides functions like:**
  - **Clustermap: `sns.clustermap()`**
  - **Pair Grid: `sns.PairGrid()`**

8. Time Series Visualization:
- **Seaborn includes functionality for visualizing time series data with functions like:**
  - **Time Series Plot: `sns.lineplot()`**
  - **Time Series Heatmap: `sns.heatmap()`**

Usage Example:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load example dataset
tips = sns.load_dataset("tips")

# Create a scatter plot with regression line
sns.lmplot(x="total_bill", y="tip", data=tips)

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

In this example, we use Seaborn to create a scatter plot (`lmplot`) of the "total_bill" versus "tip" from the "tips" dataset. Seaborn's high-level interface allows us to create a visually appealing plot with minimal code.

Seaborn is widely used for exploratory data analysis and creating publication-quality visualizations in a concise manner. Its integration with Pandas DataFrames and its ability to handle complex statistical visualizations make it a popular choice for data scientists and analysts.

9.b) Describe in detail the following term.

i) Matrix Plots ii) Regression plot

Ans.9.b)

i) Matrix Plots

A plot of matrix data is called a matrix plot. A matrix plot is a color-coded figure with values, data in the rows, and data in the columns. You can create matrix plots in seaborn by using either heatmap() or clustermap() functions.

Heatmap() is used to produce rectangular data as a color-coded matrix and clustermap() is used to plot a dataset as a hierarchically clustered heat map.

To plot any data into a map, we need to import the data. You can either use datasets available in the seaborn library or import them as per your choice from elsewhere.

ii) Regression Plots

The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. Regression plots as the name suggests creates a regression line between 2 parameters and helps to visualize their linear relationships. This article deals with those kinds of plots in seaborn and shows the ways that can be adapted to change the size, aspect, ratio etc. of such plots.

Seaborn is not only a visualization library but also a provider of built-in datasets. Here, we will be working with one of such datasets in seaborn named 'tips'. The tips dataset contains information about the people who probably had food at the restaurant and whether they left a tip. It also provides information about the gender of the people, whether they smoke, day, time and so on.

10. Explain Spatial Visualizations and Analysis in Python with Folium?

Ans.10.a)

Spatial visualizations and analysis involve representing and analyzing data in the context of geographic locations. Python provides several libraries for spatial data visualization and analysis, and one popular choice for creating interactive maps is Folium.

Folium:

Folium is a Python library that makes it easy to visualize geospatial data interactively. It builds on the capabilities of Leaflet.js, a widely used JavaScript library for interactive maps. Folium allows you to create Leaflet maps with Python and embed them in Jupyter notebooks or display them on a web page.

Key Functionalities and Usage:

1. Map Creation:

- Folium makes it simple to create a base map using the `folium.Map()` function. You can set the initial center and zoom level of the map.

```
import folium
# Create a map centered at a specific location
m = folium.Map(location=[latitude, longitude], zoom_start=12
```

2. Marker and Popup:

- You can add markers to the map to represent specific locations and attach popups with additional information.

```
# Add a marker with a popup
folium.Marker([latitude, longitude], popup='Marker Popup').add_to(m)
```

3. Circle and CircleMarker:

- Folium allows you to add circles and circle markers to represent areas or points of interest.

```
# Add a circle with a popup
folium.Circle([latitude, longitude], radius=100, popup='Circle Popup').add_to(m)
# Add a circle marker with a popup
folium.CircleMarker([latitude, longitude], radius=10, popup='Circle Marker Popup').add_to(m)
```

4. Choropleth Maps:

- Folium supports the creation of choropleth maps, where areas are shaded or colored based on a variable.

```
# Create a choropleth map
folium.Choropleth(
    geo_data=geojson_data,
    data=data,
    columns=['location', 'value'],
    key_on='feature.properties.name',
    fill_color='YlGnBu',
    fill_opacity=0.7,
    line_opacity=0.2,
).add_to(m)
```

5. Heatmaps:

- Folium enables the creation of heatmaps to visualize the intensity of data across geographic locations.

```
from folium.plugins import HeatMap
# Create a heatmap
HeatMap(data_points).add_to(m)
```

6. Layer Control:

- Folium provides a layer control feature, allowing users to toggle between different map layers.

```
# Add layer control
folium.LayerControl().add_to(m)
```

7. Save and Display:

- Once the map is created, you can save it as an HTML file or display it directly in a Jupyter notebook.

```
#Save the map as an HTML file
m.save('map.html')
# Display the map in a Jupyter notebook
m
```

Example:

Here's a simple example that creates a map with a marker:

```
import folium

# Create a map centered at San Francisco
m = folium.Map(location=[37.7749, -122.4194], zoom_start=12)

# Add a marker with a popup
folium.Marker([37.7749, -122.4194], popup='San Francisco').add_to(m)

# Display the map
m
```

This example creates a basic map centered on San Francisco with a marker indicating the city's location. Folium's simplicity and integration with Python make it a great choice for spatial visualizations and analysis.

10.b) An e-commerce company * wants to get into logistics “Deliver4U” . It wants to know the pattern for maximum pickup calls from different areas of the city throughout 3 the day. This will result in:
i) Build optimum number of stations where its pickup delivery personnel will be located.
ii) Ensure pickup personnel reaches the pickup location at the earliest possible time. For this the company uses its existing customer data in Delhi to find the highest density of probable pickup locations in the future.

Ans.10.b)

The e-commerce company "Deliver4U" can leverage visualizations to address its logistics objectives:

i) Build Optimum Number of Stations:

1. **Heatmap Analysis:**
   - **Use data visualization tools like Folium to create heatmaps that showcase the density of pickup calls in different areas of the city throughout the day.**
   - **Analyze these heatmaps to identify hotspots or areas with the highest concentration of pickup requests.**

2. **Cluster Analysis:**
   - **Employ clustering algorithms to group pickup locations based on their density. Visualize these clusters on a map to identify regions that require pickup stations.**

3. **Spatial Distribution Visualization:**
   - **Visualize the spatial distribution of pickup calls over time. This can be achieved using animated maps or time-series visualizations to highlight peak hours and areas.**

4. **Interactive Map:**
   - **Develop an interactive map that allows stakeholders to explore the pickup density in various areas and make informed decisions about station placement.**

ii) Ensure Pickup Personnel Reaches the Pickup Location Promptly:

1. **Real-Time Dashboard:**
   - **Create a real-time dashboard using data visualization tools to monitor the current location and status of pickup personnel.**
   - **Incorporate live updates and alerts to ensure timely response to changing conditions.**

2. **Route Optimization Visualization:**
   - **Implement visualizations for route optimization, displaying the most efficient routes for pickup personnel. Use color-coded lines or arrows to represent different routes and their congestion levels.**

3. **Time-Series Analysis:**
   - **Visualize time-series data of pickup times, identifying patterns and potential bottlenecks in the logistics process.**
   - **Use line charts or bar graphs to illustrate variations in pickup times throughout the day.**

4. **Performance Metrics Visualization:**
   - **Display key performance metrics related to pickup times and delivery efficiency. Visualize metrics such as average pickup time, on-time delivery rates, and any deviations from the schedule.**

5. **Predictive Analytics:**
   - **Implement predictive analytics models to forecast future pickup demands. Visualize these predictions on a map to identify areas with anticipated increases in demand.**

Overall:

- **Combine different visualization techniques to provide a comprehensive view of pickup patterns and logistics operations.**
- **Utilize interactive and dynamic visualizations to empower decision-makers with real-time insights.**
- **Continuously refine strategies based on visualized data and feedback to enhance the efficiency of pickup operations.**

By adopting a data-driven visualization approach, Deliver4U can make informed decisions about station placement, route optimization, and overall logistics management, ensuring timely and effective pickup operations in Delhi.