Marathwada Mitra Mandal’s

College of Engineering, Pune

Karvenagar, Pune-411 052

Accredited with 'A++' Grade by NAAC

Department of Artificial Intelligence & Data Science

Lab Manual

417525 – Computer Laboratory I

'येथे बहुतांचे हित ।'

Marathwada Mitra Mandal's

COLLEGE OF ENGINEERING

Karvenagar, Pune

Vision

To aspire for the Welfare of Society
through excellence in Science and Technology.

Mission

Our Mission is to

- **M**ould young talent for higher endeavours.
- **M**eet the challenges of globalization.
- **C**ommit for social progress with values and ethics.
- **O**rient faculty and students for research and development.
- **E**mphasize excellence in all disciplines.

Marathwada Mitra Mandal's

College of Engineering

Department of Artificial Intelligence and Data Science

VISION

"Excellence in the field of Artificial Intelligence and Data science"

MISSION

- **To encourage the students for learning various AI & DS based tools & techniques**
- **To groom students technologically superior**
- **To educate students in a way to meet the needs of society**

A. Course Outcome

Course Outcome	Statement
	At the end of the course, a student will be able to (write/install/solve/apply)
417525.1	Implement regression, classification and clustering models
417525.2	Integrate multiple machine learning algorithms in the form of ensemble learning
417525.3	Apply reinforcement learning and its algorithms for real world applications
417525.4	Analyze the characteristics, requirements of data and select an appropriate data model
417525.5	Apply data analysis and visualization techniques in the field of exploratory data science
417525.6	Evaluate time series data

B. CO-PO mapping

Course Outcome	Program outcomes											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
417525.1	3	3	3	2	3	-	-	-	2	2	1	1
417525.2	3	3	3	2	3	-	-	-	2	2	1	1
417525.3	3	3	3	2	3	-	-	-	2	2	1	1
417525.4	3	2	2	3	3	-	-	-	2	1	1	1
417525.5	3	2	2	3	3	-	-	-	2	1	1	1
417525.6	3	2	2	3	3	-	-	-	2	2	1	1
Average	3	3	3	3	3	0	0	0	2	2	1	1

CO-PSO mapping

Course Outcome	Program Specific Outcomes		
	1	**2**	**3**
417525.1	-	-	-
417525.2	-	-	-
417525.3	-	-	-
417525.4	-	-	-
417525.5	-	-	-
417525.6	2	-	-
Average	2	-	-

I N D E X

CERTIFICATE

Certified that Mr./Ms.____ of Class BE AI & DS Roll No.__________ has completed the Tutorial work in the subject **Computer Laboratory I (417525)** during the academic year 2024-25 Sem I.

Signature of the Faculty

Signature of Head of the Department

Date:

****

Part II
Assignment 7

****

Title: Data Loading, Storage and File Formats

Problem Statement: Analyzing Sales Data from Multiple File Formats

Dataset: Sales data in multiple file formats (e.g., CSV, Excel, JSON)

Description: The goal is to load and analyze sales data from different file formats, including CSV, Excel, and JSON, and perform data cleaning, transformation, and analysis on the dataset.

Tasks to Perform:

Obtain sales data files in various formats, such as CSV, Excel, and JSON.

1. Load the sales data from each file format into the appropriate data structures or dataframes.
2. Explore the structure and content of the loaded data, identifying any inconsistencies, missing values, or data quality issues.
3. Perform data cleaning operations, such as handling missing values, removing duplicates, or correcting inconsistencies.
4. Convert the data into a unified format, such as a common dataframe or data structure, to enable seamless analysis.
5. Perform data transformation tasks, such as merging multiple datasets, splitting columns, or deriving new variables.
6. Analyze the sales data by performing descriptive statistics, aggregating data by specific variables, or calculating metrics such as total sales, average order value, or product category distribution.
7. Create visualizations, such as bar plots, pie charts, or box plots, to represent the sales data and gain insights into sales trends, customer behavior, or product performance.

Objective of the Assignment:

Students should be able to load multiple file formats and perform data cleaning, transformation, and analysis on the dataset.

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Loading, Preprocessing, Data Formatting, Data Normalization and Data Cleaning.

Theory:

Reading data and making it accessible (often called data loading) is a necessary first step for using most of the tools. The term parsing is also sometimes used to describe loading text data and interpreting it as tables and different data types. To focus on data input and output using pandas, there are numerous tools in other libraries to help with reading and writing data in various formats. Pandas features a number of functions for reading tabular data as a DataFrame object. pandas.read_csv is one of the most frequently used to load csv files.

Some of data loading functions in pandas summarizes as;

Function	Description
read_csv	Load delimited data from a file, URL, or file-like object; use comma as default delimiter
read_fwf	Read data in fixed-width column format (i.e., no delimiters)
read_clipboard	Variation of read_csv that reads data from the clipboard; useful for converting tables from web pages
read_excel	Read tabular data from an Excel XLS or XLSX file
read_hdf	Read HDF5 files written by pandas
read_html	Read all tables found in the given HTML document
read_json	Read data from a JSON (JavaScript Object Notation) string representation, file, URL, or file-like object
read_feather	Read the Feather binary file format
read_orc	Read the Apache ORC binary file format
read_parquet	Read the Apache Parquet binary file format
read_pickle	Read an object stored by pandas using the Python pickle format
read_sas	Read a SAS dataset stored in one of the SAS system's custom storage formats
read_spss	Read a data file created by SPSS
read_sql	Read the results of a SQL query (using SQLAlchemy)
read_sql_table	Read a whole SQL table (using SQLAlchemy); equivalent to using a query that selects everything in that table using read_sql
read_stata	Read a dataset from Stata file format
read_xml	Read a table of data from an XML file

Indexing:

Can treat one or more columns as the returned DataFrame, and whether to get column names from the file, arguments you provide, or not at all.

Type inference and data conversion:

Includes the user-defined value conversions and custom list of missing value markers**Date and time parsing:**

Includes a combining capability, including combining date and time information spread over multiple columns into a single column in the result

Iterating:
Support for iterating over chunks of very large files.
Unclean data issues:
Includes skipping rows or a footer, comments, or other minor things like numeric data with thousands separated by commas

Because of how messy data in the real world can be, some of the data loading functions (especially pandas.read_csv) have accumulated a long list of optional arguments over time.

Some of these functions perform *type inference*, because the column data types are not part of the data format. That means you don't necessarily have to specify which columns are numeric, integer, Boolean, or string. Other data formats, like HDF5, ORC, and Parquet, have the data type information embedded in the format.

Let's start to load sales data from different file format such as csv, exl,json etc

3. Perform data cleaning operations, such as handling missing values, removing duplicates, or correcting inconsistencies.

Checking of Missing Values in Dataset:

- **isnull()** is the function that is used to check missing values or null values in pandas python.
- **isna()** function is also used to get the count of missing values of column and row wise count of missing values

4. Convert the data into a unified format, such as a common dataframe or data structure, to enable seamless analysis.

Working with datetime in Pandas can sometimes be tricky if your datetime column is a string with messy, mixed formats. You will need to do some pre-processing and convert the mixed formats to the standard datetime type in order to perform any subsequent data analysis.

Let's take a look at the following example. In this dataframe, we have a column named 'orderdate' which appears to be a datetime column but is actually a string column. Moreover, the dates are all arranged in different formats.

For example, for ID 0, the date string is arranged in the DD.MM.YYYY format in which the day is written first. For ID 1, the date string is displayed in the DD-MM-YY format in which the day is also written first. For ID 2, the date string is arranged in the format MM/DD/YYYY with the month being written first which is common in the U.S.

So to clean this messy date string column that has mixed formats, easily and efficiently?

Let's first try Pandas' to_datetime() method which can parse any valid date strings to datetime easily. If we simply do the following without any additional arguments, we get the results shown below:

```
df['joining_date'] = pd.to_datetime(df['joining_date'])
```
We can see that with the pd.to_datetime() method, we are able to parse the date strings that have

mixed formats to datetime with a standard format (default is YYYY-MM-DD).

5. Perform data transformation tasks, such as merging multiple datasets, splitting columns, or deriving new variables.

Sort and filter

Often the ordering of data matters, particularly when manually inspecting datasets. In addition, it can be useful to filter the dataset and only use particular rows or particular columns of data. To do this, we will use the following functions from pandas: sort_values, query, and filter.

New variables

Another common example of creating a new variable is when a dataset contains birthdates. A more useful feature might be "age" rather than using the raw birthdate. The code below first creates a function to calculate age and then creates a new column to contain this data.

Splitting and combining

Often when working with text columns, data needs to be split and/or combined in various ways. The code below demonstrates how the name column can be split into first and last names. The next code block shows how to recombine the first and last names using the 'last, first' convention.

Conclusion:
We have successfully loaded and analyzed sales data from different file formats, including CSV, Excel, and JSON, and performed data cleaning, transformation, and analysis on the dataset.

FAQs:

1. **What are the best practices for storing sales data?**
2. **How should data be organized in a database for efficient querying?**
3. **What common errors should I watch out for when loading data?**
4. **How can I ensure data quality during the import process?**
5. **What are the pros and cons of CSV files?**

****

Part II

Assignment 8

****

Title: Interacting with Web APIs

Problem Statement: Analyzing Weather Data from OpenWeatherMap API

Dataset: Weather data retrieved from OpenWeatherMap API

Description: The goal is to interact with the OpenWeatherMap API to retrieve weather data for a specific location and perform data modeling and visualization to analyze weather patterns over time.

Tasks to Perform:

1. Register and obtain API key from OpenWeatherMap.
2. Interact with the OpenWeatherMap API using the API key to retrieve weather data for a specific location.
3. Extract relevant weather attributes such as temperature, humidity, wind speed, and precipitation from the API response.
4. Clean and preprocess the retrieved data, handling missing values or inconsistent formats.
5. Perform data modeling to analyze weather patterns, such as calculating average temperature, maximum/minimum values, or trends over time.
6. Visualize the weather data using appropriate plots, such as line charts, bar plots, or scatter plots, to represent temperature changes, precipitation levels, or wind speed variations.
7. Apply data aggregation techniques to summarize weather statistics by specific time periods (e.g., daily, monthly, seasonal).
8. Incorporate geographical information, if available, to create maps or geospatial visualizations representing weather patterns across different locations.
9. Explore and visualize relationships between weather attributes, such as temperature and humidity, using correlation plots or heatmaps.

What is Web API and why do we use it ?

API stands for Application Programming Interface. API is actually some kind of interface which has a set of functions. These set of functions will allow programmers to acquire some specific features or the data of an application.

ASP.NET Web API: ASP.NET stands for Active Server Pages.NET. It is mostly used for creating web pages and web technologies. It is considered a very important tool for developers to build dynamic web pages using languages like C# and Visual Basic.

How to use Web API?

Web API receives requests from different types of client devices like mobile, laptop, etc, and then sends those requests to the web server to process those requests and returns the desired output to the client.
Web API is a System to System interaction, in which the data or information from one system can be accessed by another system, after the completion of execution the resultant data or we can say as output is shown to the viewer.

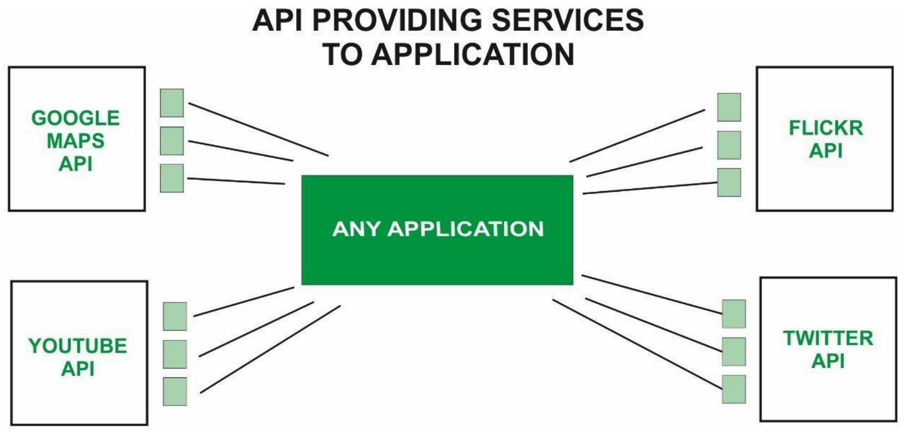
API provides data to its programmers which is made available to outside users. When programmers decide to make some of their data available to the public, they "expose endpoints" meaning they publish a portion of the language they have used to build their program. Other programmers can then extract the data from the application by building URLs or using HTTP clients to request data from those endpoints.

Server Side: A server-side web API is a programmatic interface. It consists of one or more publicly exposed endpoints. It defines a request-response message system. Mashup is a web application that is a server-side API that combines several server-side APIs. Webhook is a server-side API that takes input as a uniform resource identifier.

Client Side: Client Side web APIs target standardized JavaScript bindings. Google created their native client architecture designed to replace native plug-ins with secure native sandboxed extensions and applications.

Popular API Examples:

- **Google Maps API's**: Google Maps APIs allows developers to use Google Maps on Webpages using a JavaScript or Flash interface.
- **YouTube API's**: Google's API lets developers integrate YouTube and functionality into websites or applications. YouTube APIs includes the YouTube analytics API, YouTube Data API, YouTube live streaming API, YouTube Player APIs and others.
- **The Flickr APIs**: It is used by developers to access the Flick photo sharing community data.
- **Twitter APIs**: Twitter offers two APIs, the REST API allows developers to access core Twitter data and the search API provides methods for developers to



interact with twitter search and trends data.

1. To register and obtain an API key from OpenWeatherMap, follow these steps:

- Go to the OpenWeatherMap website and create an account.
- Once you have created an account, you will be redirected to your dashboard.

- On your dashboard, click the API keys tab.
- Click the Generate API key button.
- Enter a name for your API key and click the Generate button.
- Your API key will be displayed on the screen. Copy and paste your API key in a
- safe place.
- You will need your API key to access the OpenWeatherMap API and retrieve
- weather data.
- If you lose your API key, you can generate a new one from your dashboard.

2. Interact with the OpenWeatherMap API using the API key to retrieve weather data for a specific location.

- Construct the API request URL
- Send the API request
- api_response = requests.get(api_request_url)

Parse the API response. The API response will be in JSON format. You can use a JSON parsing library to parse the API response and extract the relevant weather data.

3. Extract relevant weather attributes such as temperature, humidity, wind speed, and precipitation from the API response.

Once you have extracted the relevant weather attributes, you can use them for your analysis or visualization needs. The OpenWeatherMap API provides weather data in different units of measurement. You can use the units parameter in the API request to specify the desired units of measurement.

4. Clean and preprocess the retrieved data, handling missing values or inconsistent formats.

To clean and preprocess the retrieved weather data, you can follow these steps:

Identify missing values. You can use a variety of methods to identify missing values, such as checking for empty fields or values that are outside of the expected range.
Handle missing values. Once you have identified missing values, you can handle them in a variety of ways, such as imputing the missing values with a default value or removing the rows with missing values.
Identify inconsistent formats. You can use a variety of methods to identify inconsistent formats, such as checking for dates in different formats or numbers with different decimal places.
Convert inconsistent formats to a consistent format. Once you have identified inconsistent formats, you can convert them to a consistent format. For example, you can convert all dates to the same format or all numbers to the same number of decimal places.
Remove outliers. You can use a variety of methods to identify outliers, such as using the interquartile range (IQR). Handle outliers. Once you have identified outliers, you

can handle them in a variety of ways, such as removing them from the data or trimming them to the nearest value within the IQR.

5. Perform data modeling to analyze weather patterns, such as calculating average temperature, maximum/minimum values, or trends over time.

To perform data modeling to analyze weather patterns, you can use a variety of statistical methods. Some common methods include:

Descriptive statistics: Descriptive statistics can be used to calculate summary statistics such as the mean, median, mode, standard deviation, and range. These summary statistics can be used to describe the overall characteristics of the weather data.
Time series analysis: Time series analysis can be used to identify trends and patterns in the weather data over time. For example, you could use time series analysis to identify the seasonal trend in temperature or the long-term trend in precipitation.

Regression analysis: Regression analysis can be used to identify relationships between different weather variables. For example, you could use regression analysis to identify the relationship between temperature and humidity, or the relationship between wind speed and precipitation.

Here are some specific examples of data modeling that you could perform to analyze weather patterns:

- Calculate the average temperature for each month of the year. This would give you an idea of the seasonal variation in temperature.
- Calculate the maximum and minimum temperature for each day of the week. This would give you an idea of the daily range in temperature.
- Calculate the linear trend of the temperature over time. This would tell you whether the temperature is increasing or decreasing over time.
- Create a scatter plot of temperature and humidity. This would allow you to visualize the relationship between these two variables.
- Use regression analysis to model the relationship between temperature and humidity. This would allow you to predict the humidity for a given temperature.
- You can also use data modeling to analyze weather patterns across different locations. For example, you could calculate the average temperature for different countries around the world, or the total precipitation for each season. You could also create maps or geospatial visualizations to represent weather patterns across different locations.
- The specific data modeling techniques that you use will depend on your specific research questions. However, the methods described above are a good starting point for analyzing weather patterns.

6. Visualize the weather data using appropriate plots, such as line charts, bar plots, or scatter plots, to represent temperature changes, precipitation levels, or wind speed variations.

Choose the appropriate plot type. The type of plot that you choose will depend on the specific weather patterns that you want to visualize. For example, if you want to visualize temperature changes over time, you could use a line chart. If you want to visualize precipitation levels by month, you could use a bar plot. If you want to visualize the relationship between temperature and humidity, you could use a scatter plot.

Interpret the visualization. Once you have created the visualization, you need to interpret it. What does the visualization tell you about the weather patterns?

Here are some specific examples of visualizations that you could create to represent temperature changes, precipitation levels, or wind speed variations:

Line chart of temperature over time: This visualization would show how the temperature has changed over time. You could use this visualization to identify trends in the temperature, such as whether the temperature is increasing or decreasing over time.
Bar plot of precipitation by month: This visualization would show the total precipitation for each month. . You could use this visualization to identify seasonal trends in precipitation.
Scatter plot of temperature and humidity: This visualization would show the relationship between temperature and humidity. You could use this visualization to identify whether there is a correlation between these two variables.

7. Apply data aggregation techniques to summarize weather statistics by specific time periods (e.g., daily, monthly, seasonal).

- Interpret the summary statistics. What do the summary statistics tell you about the weather patterns during the specified time period?
- Here are some specific examples of data aggregation techniques that you could use to summarize weather statistics by specific time periods:
- Calculate the maximum and minimum temperature for each day of the week. This would give you an idea of the daily range in temperature.
- Calculate the total precipitation for each season. This would give you an idea of the seasonal variation in precipitation.
- Calculate the average wind speed for each month of the year. This would give you an idea of the seasonal variation in wind speed.

8. Incorporate geographical information, if available, to create maps or geospatial visualizations representing weather patterns across different locations

- Obtain geographical information for the weather data. This geographical information could include latitude and longitude coordinates, or country or region names.
- Choose a mapping or geospatial visualization tool. There are a variety of mapping and geospatial visualization tools available, both free and paid. Some popular options include Google Maps, Carto, and QGIS.
- Load the weather data and geographical information into the mapping or geospatial visualization tool.
- Create the map or geospatial visualization. The specific steps involved in creating the map or geospatial visualization will vary depending on the tool that you are using. However, most tools will allow you to select the type of map or visualization that you want to create, and to specify the data that you want to display.
- Interpret the map or geospatial visualization. What does the map or geospatial visualization tell you about the weather patterns across different locations?
- Here are some specific examples of maps or geospatial visualizations that you could create to represent weather patterns across different locations:

Conclusion: We have studied Web API and Analyzing Weather Data from OpenWeatherMap using API.

FAQs:

1. **What is the OpenWeatherMap API?**
2. **What kind of weather data can I access through the OpenWeatherMap API?**
3. **What are the common endpoints available in the OpenWeatherMap API?**
4. **How is the weather data returned by the API structured?**
5. **What units are used for the weather data, and can they be converted?**
6. **What should I do if I receive an error response from the API?**

****

Part II
Assignment 9

****

Title: Data Cleaning and Preparation

Problem Statement: Analyzing Customer Churn in a Telecommunications Company

Dataset: "Telecom_Customer_Churn.csv"

Description: The dataset contains information about customers of a telecommunications company and whether they have churned (i.e., discontinued their services). The dataset includes various attributes of the customers, such as their demographics, usage patterns, and account information. The goal is to perform data cleaning and preparation to gain insights into the factors that contribute to customer churn.

Tasks to Perform:

1. Import the "Telecom_Customer_Churn.csv" dataset.
2. Explore the dataset to understand its structure and content.
3. Handle missing values in the dataset, deciding on an appropriate strategy.
4. Remove any duplicate records from the dataset.
5. Check for inconsistent data, such as inconsistent formatting or spelling variations, and standardize it.
6. Convert columns to the correct data types as needed.
7. Identify and handle outliers in the data.
8. Perform feature engineering, creating new features that may be relevant to predicting customer churn.
9. Normalize or scale the data if necessary.
10. Split the dataset into training and testing sets for further analysis.
11. Export the cleaned dataset for future analysis or modeling.

Theory:

Data cleaning and preparation is the process of transforming raw data into a format that is ready for analysis. This involves identifying and correcting errors, filling in missing values, and dealing with outliers. Data preparation can be a time-consuming process, but it is essential for ensuring that your data is accurate and ready for analysis.

Steps involved in data cleaning and preparation:

Step 1: Remove duplicate or irrelevant observations.

Duplicate observations can occur when data is collected from multiple sources or when data is entered manually. Irrelevant observations are data points that are not relevant to your analysis. For example, if you are analyzing data on customer purchases, you may want to remove observations for customers who have not made a purchase in the past year.

Step 2: Fix structural errors.

Structural errors are errors in the format of the data. For example, a date column may contain dates in different formats, or a customer name column may contain names in different formats. Structural errors can be corrected by using consistent formatting rules and by converting data to a consistent format.

Step 3: Filter unwanted outliers.

Outliers are data points that are significantly different from the rest of the data. Outliers can be caused by errors in data collection or entry, or they may be legitimate data points that are simply unusual. Identifying and filtering outliers before analysing your data is important, as they can skew your results.

Step 4: Handle missing data.

Missing data is a common problem in data sets. Missing data can occur when data is not collected properly, or when data is lost during data processing. There are a number of ways to handle missing data, such as deleting observations with missing values, imputing missing values, or creating a new variable to indicate whether a value is missing.

Step 5: Validate and QA.

Once you have cleaned and prepared your data, it is important to validate it to ensure that it is accurate and ready for analysis. This involves checking for any remaining errors, inconsistencies, or outliers. You may also want to perform a quality assurance (QA) check to ensure that your data meets your specific requirements.

1. **Import the "Telecom_Customer_Churn.csv" dataset.**

Since this is comma-delimited, we can then use pandas. read_csv is used to read it into a DataFrame:

2. **Explore the dataset to understand its structure and content.**

As you can see, there are 10 missing values in the Year_Built column. We can decide on an appropriate strategy for handling these missing values based on the following factors:

- **The percentage of missing values in the column:** If the percentage of missing values is high, it may be necessary to drop the column or impute the missing values with a constant value. In this case, the percentage of missing values in the Year_Built column is relatively low (10%), so we can impute the missing values.
- **The type of data in the column:** If the data in the column is continuous, we can impute the

missing values with the mean, median, or mode of the column. If the data in the column is categorical, we can impute the missing values with the most frequent value in the column. In this case, the data in the Year_Built column is continuous, so we can impute the missing values with the median value of the column.
To impute the missing values in the Year_Built column with the median value, we can use the following Python code:

3. Handle missing values in the dataset, deciding on an appropriate strategy.
Checking of Missing Values in Dataset:
● **isnull()** is the function that is used to check missing values or null values in pandas python.
● **isna()** function is also used to get the count of missing values of column and row wise count of missing values

4. Remove any duplicate records from the dataset.

5. Check for inconsistent data, such as inconsistent formatting or spelling variations, and standardize it.

To check for inconsistent data in the Telecom_Customer_Churn dataset, you can use the following code:

```
# Check for inconsistent formatting
for column in df.columns:
print("Column name:", column)
print("Unique values:", df[column].unique())

# Check for spelling variations
for column in df.columns:
print("Column name:", column)
print("Most common values:", df[column].value_counts().head(10))
```

This will print the unique values and most common values for each column in the dataset. You can then manually inspect the output to identify any inconsistencies.

Once you have identified any inconsistencies, you can standardize the data using the following code:

```
# Standardize the data
# For example, to standardize the "MonthlyCharge" column, you could use
the following code:
df["MonthlyCharge"] = df["MonthlyCharge"].apply(lambda x:
float(x.replace("$", "").replace(",", "")))
# Save the cleaned dataset
df.to_csv("Telecom_Customer_Churn_cleaned.csv",
index=False)
```

6. Convert columns to the correct data types as needed.

To convert columns to the correct data types in the Telecom_Customer_Churn dataset in Python, you can use the following code:

```
import pandas as
pd # Load the
dataset
df = pd.read_csv("Telecom_Customer_Churn.csv")

# Convert columns to the correct data types
df["customerID"] = df["customerID"].astype(str)
df["tenure"] = df["tenure"].astype(int)
df["MonthlyCharge"] = df["MonthlyCharge"].astype(float)
df["TotalCharge"] = df["TotalCharge"].astype(float)
df["Churn"] = df["Churn"].astype(int)

# Save the cleaned dataset
df.to_csv("Telecom_Customer_Churn_cleaned.csv",
index=False)
```

7. Identify and handle outliers in the data.

Once you have identified outliers, you can handle them in a number of ways. You can:

- Delete the outliers: This is the simplest approach, but it can lead to a loss of data.
- Impute the outliers: This involves replacing the outliers with more reasonable values.
- Transform the data: This involves transforming the data in a way that reduces the impact of outliers.

8. Perform feature engineering, creating new features that may be relevant to predicting customer churn.

Feature engineering is the process of creating new features from existing features in a dataset. This can be done to improve the performance of a machine learning model or to make the data easier to understand and analyze.

Examples of features for the Telecom_Customer_Churn dataset:

- Average monthly spends: This could be calculated by dividing the TotalCharge column by the Tenure column.

- Tenure since last service change: This could be calculated by subtracting the most recent service change date from the current date.

9. Normalize or scale the data if necessary.

Normalizing or scaling data is an important step in data preparation, especially for machine learning models. Normalization and scaling can make the data easier to analyze and improve the performance of machine learning models.

Normalization is the process of transforming data so that it has a mean of 0 and a standard deviation of 1. This can be done using the following formula:

Normalized value = (Value - Mean) / Standard deviation

Scaling is the process of transforming data so that it falls within a specific range. This can be done using the following formula:

Scaled value = (Value - Minimum value) / (Maximum value - Minimum value)

To normalize or scale the data in the Telecom_Customer_Churn dataset in Python, you can use the following code:

When to normalize and when to scale:

- Normalize the data if you are using a machine learning model that is sensitive to the scale of the data, such as logistic regression.
- Scale the data if you are using a machine learning model that is not sensitive to the scale of the data, such as support vector machines.
- If you are unsure whether to normalize or scale the data, you can try both approaches and see which one performs better on your dataset.

10. Split the dataset into training and testing sets for further analysis.

To split the Telecom_Customer_Churn dataset into training and testing sets in Python, You can then use the training set to train your machine-learning model and the testing set to evaluate the performance of your model.

11. Export the cleaned dataset for future analysis or modeling.

To export the cleaned Telecom_Customer_Churn dataset for future analysis or modeling in Python, you can use the following code:

```
import pandas as pd

# Load the cleaned dataset

df =

pd.read_csv("Telecom_Customer_Churn_cleaned.csv") #

Export the cleaned dataset

df.to_csv("Telecom_Customer_Churn_final.csv", index=False)
```

This code will export the cleaned dataset to a new CSV file called "Telecom_Customer_Churn_final.csv". You can then use this dataset for future analysis or modeling.

Conclusion:

We have implemented data cleaning and preparation to gain insights into the factors that contribute to customer churn.

FAQs:

1. **Why is analyzing customer churn important for telecommunications companies?**
2. **What are common reasons for customer churn in telecommunications?**
3. **How can we measure customer churn?**
4. **What analytical methods are commonly used to study churn?**
5. **How can we predict which customers are likely to churn?**
6. **What strategies can help reduce customer churn?**

****

Part II
Assignment
10

****

Title: Data Wrangling

Problem Statement: Data Wrangling on the Real Estate Market.

Dataset: "RealEstate_Prices.csv"

Description: The dataset contains information about housing prices in a specific real estate market. It includes various attributes such as property characteristics, location, sale prices, and other relevant features. The goal is to perform data wrangling to gain insights into the factors influencing housing prices and prepare the dataset for further analysis or modeling.

Tasks to Perform:

1. Import the "RealEstate_Prices.csv" dataset. Clean column names by removing spaces, special characters, or renaming them for clarity.
2. Handle missing values in the dataset, deciding on an appropriate strategy (e.g., imputation or removal).
3. Perform data merging if additional datasets with relevant information are available (e.g., neighborhood demographics or nearby amenities).
4. Filter and subset the data based on specific criteria, such as a particular time period, property type, or location.
5. Handle categorical variables by encoding them appropriately (e.g., one-hot encoding or label encoding) for further analysis.
6. Aggregate the data to calculate summary statistics or derived metrics such as average sale prices by neighborhood or property type.
7. Identify and handle outliers or extreme values in the data that may affect the analysis or modeling process.

Theory:

Data wrangling is the process of transforming raw data into a more usable form. It involves cleaning, organizing, and enriching data to make it ready for analysis. Data wrangling can be a complex and time-consuming process, but it is essential for ensuring that data is accurate, complete, and consistent before it is used for downstream purposes such as machine learning or reporting.

Benefits of data wrangling:

- **Improved data quality**: Data wrangling can help to identify and correct errors in the data, as well as format the data in a consistent manner. This can lead to improved data quality, which can in turn lead to more accurate and reliable insights.
- **Increased data usability**: Data wrangling can help to make data more usable by organizing it in a way that is easy to understand and analyze. This can lead to faster and more efficient data analysis.

- **Enhanced data insights**: By cleaning and organizing data, data wrangling can help to reveal hidden patterns and trends in the data. This can lead to more valuable and actionable insights.

Data wrangling tasks include:

- **Data cleaning**: This involves identifying and correcting errors in the data, such as duplicate values, missing values, and inconsistent formatting.
- **Data organization**: This involves structuring the data in a way that makes it easy to understand and analyze. This may involve creating new columns, merging or splitting datasets, and renaming columns.
- **Data enrichment**: This involves adding new information to the data to make it more complete and informative. This may involve using external data sources, such as government databases or industry reports.
- Explore the data: This involves getting to know the data and understanding its structure, format, and content.
- Identify and correct errors: This may involve checking for duplicate values, missing values, and inconsistent formatting.
- Transform the data: This may involve converting data types, merging datasets, and splitting datasets.
- Clean the data: This may involve removing outliers, imputing missing values, and handling special characters.
- Validate the data: This involves ensuring that the data is accurate and consistent with the desired outcome.

1. **Import the "RealEstate_Prices.csv" dataset. Clean column names by removing spaces, special characters, or renaming them for clarity in detail**

Since this is comma-delimited, we can then use pandas. read_csv is used to read it into a DataFrame:

2.Handle missing values in the dataset, deciding on an appropriate strategy (e.g., imputation or removal).

Checking of Missing Values in Dataset:
isnull() is the function that is used to check missing values or null values in pandas python.
isna() function is also used to get the count of missing values of column and row wise count of missing values.

As you can see, there are 10 missing values in the Year_Built column. We can decide on an appropriate strategy for handling these missing values based on the following factors:

- **The percentage of missing values in the column**: If the percentage of missing values is high, it may be necessary to drop the column or impute the missing values with a constant value. In this case, the percentage of missing values in the Year_Built column is relatively low (10%), so we can impute the missing values.

- **The type of data in the column**: If the data in the column is continuous, we can impute the missing values with the mean, median, or mode of the column. If the data in the column is categorical, we can impute the missing values with the most frequent value in the column. In this case, the data in the Year_Built column is continuous, so we can impute the missing values with the median value of the column.

Once we have imputed the missing values, we can check for any remaining missing values in the dataset.

3. Perform data merging if additional datasets with relevant information are available .(e.g., neighborhood demographics or nearby amenities).

This will merge the two DataFrames on the Neighborhood_Name column, creating a new DataFrame that contains all of the columns from both of the original DataFrames.

Benefits of data merging

Data merging can be used to:

- Combine data from multiple sources to create a more comprehensive dataset.
- Create new features from existing data.
- Identify relationships between different datasets

5. Handle categorical variables by encoding them appropriately (e.g., one-hot encoding or label encoding) for further analysis.

The encoding technique will depend on the specific dataset and the machine learning algorithm being used. However, a good general rule of thumb is to use one-hot encoding for categorical variables with a large number of categories and label encoding for categorical variables with a small number of categories.

6. Aggregate the data to calculate summary statistics or derived metrics such as average sale prices by neighborhood or property type.

As you can see, the average sale price has been calculated by neighborhood and property type.

7. Identify and handle outliers or extreme values in the data that may affect the analysis or modeling process.

How to handle outliers

Once you have identified the outliers in your data, you can choose to handle them in a number of ways, including:

- **Remove the outliers.** This is the simplest and most common way to handle outliers.

However, it is important to note that removing outliers can reduce the power of your analysis and may bias your results.

- **Cap the outliers.** This involves setting the values of the outliers to the lower or upper bound. This can be a good option if you want to reduce the impact of the outliers without removing them entirely.
- **Transform the data.** This involves using a mathematical transformation to reduce the impact of the outliers. For example, you could use a log transformation or a square root transformation.
- **Model the outliers.** This involves using a statistical model to account for the outliers. This is the most complex option, but it can be the most effective if you have a good understanding of the data and the underlying process that generated it.

As you can see, the data point with a value of 1000000 has been flagged as an outlier.

Conclusion:

We have implemented Data Wrangling on the Real Estate Market and gain insights into the factors influencing housing prices.

FAQs:

1. **What is data wrangling?**
2. **What types of data are commonly used in real estate analysis?**
3. **How do you handle missing data in real estate datasets?**
4. **What tools can be used for data wrangling in real estate?**
5. **How can data wrangling improve decision-making in real estate investment?**

****

Part II
Assignment 11

****

Title: Data Visualization using matplotlib

Problem Statement: Analyzing Air Quality Index (AQI) Trends in a City.

Dataset: “City_Air_Quality.csv”

Description: The dataset contains information about air quality measurements in a specific city over a period of time. It includes attributes such as date, time, pollutant levels (e.g., PM2.5, PM10, CO), and the Air Quality Index (AQI) values. The goal is to use the matplotlib library to create visualizations that effectively represent the AQI trends and patterns for different pollutants in the city.

Tasks to Perform:

1. Import the “City_Air_Quality.csv” dataset.
2. Explore the dataset to understand its structure and content.
3. Identify the relevant variables for visualizing AQI trends, such as date, pollutant levels, and AQI values.
4. Create line plots or time series plots to visualize the overall AQI trend over time.
5. Plot individual pollutant levels (e.g., PM2.5, PM10, CO) on separate line plots to visualize their trends over time.
6. Use bar plots or stacked bar plots to compare the AQI values across different dates or time periods.
7. Create box plots or violin plots to analyze the distribution of AQI values for different pollutant categories.
8. Use scatter plots or bubble charts to explore the relationship between AQI values and pollutant levels.
9. Customize the visualizations by adding labels, titles, legends, and appropriate color schemes.

Theory:

Data Visualization is the process of presenting data in the form of graphs or charts. It helps to understand large and complex amounts of data very easily. It allows the decision-makers to make decisions very efficiently and also allows them to identify new trends and patterns very easily. It is also used in high-level data analysis for Machine Learning and Exploratory Data Analysis (EDA).

Matplotlib is a low-level library of Python which is used for data visualization. It is easy to use and emulates MATLAB like graphs and visualization. This library is built on the top of NumPy arrays and consists of several plots like line chart, bar chart, histogram, etc. Installation,we will use the pip command to install this module.

To install Matplotlib type the below command in the terminal:

Import the "City_Air_Quality.csv" dataset.

In this experiment City_Air_Quality.csv is used. The dataset contains information about air quality measurements in a specific city over a period of time. It includes attributes such as date, time, pollutant levels (e.g.,PM2.5, PM10, CO), and the Air Quality Index (AQI) values. The goal is to use the matplotlib library to create visualizations that effectively represent the AQI trends and patterns for different pollutants in the city.

Since this is comma-delimited, we can then use pandas.read_csv to read it into a DataFrame:

Explore the dataset to understand its structure and content.

Data is to discover the different data types it contains. While you can put anything into a list, the columns of a DataFrame contain values of a specific data type.

You can display all columns and their data types with .info():

What data types are in your dataset, it's time to get an overview of the values each column contains. You can do this with .describe():

```
pipinstall matplotlib
```

Identify the relevant variables for visualizing AQI trends, such as date, pollutant levels, and AQI values.

Predicting Air Quality Index (AQI)

AQI: The air quality index is an index for reporting air quality on a daily basis. In other words, it is a measure of how air pollution affects one's health within a short time period. The AQI is calculated based on the average concentration of a particular pollutant measured over a standard time interval. Generally, the time interval is 24 hours for most pollutants, and 8 hours for carbon monoxide and ozone.

We can see how air pollution is by looking at the AQ

AQI Level	AQI Range
Good	0-50
Moderate	51-100
Unhealthy	101-150
Unhealthy for strong people	151-200
Hazardous	200+

Checking the dataset

We can see that there are quite a number of NaNs in the dataset. To proceed with the EDA, we must handle these NaNs by either removing them or filling them. I will be doing both.

Cleaning the Dataset

Removing NaNs Looking at the dataset head, we can conclude that the following columns:

1. stn_code
2. Agency
3. sampling_date
4. location_monitoring_agency

1. Dropping unnecessary columns: dataset in terms of information that can't already be extracted from other columns. Therefore, we drop these columns. We use `drop()` function to remove the columns 'stn_code', 'agency', 'sampling_date', and 'location_monitoring_station' from the DataFrame. These columns are dropped because they are not required for further analysis.
2. Dropping rows with missing dates: The date also has missing values, we will drop the rows containing these values as they're of little use. The `dropna()` function is used to remove rows where the 'date' column is missing (NaN). This ensures that only rows with valid dates are kept in the DataFrame.
3. Cleaning up state name changes: Cleaning values since the geographical nomenclature has changed over time, we change it here as well to correspond to more accurate insights. The code replaces the state name 'Uttaranchal' with 'Uttarakhand' using the `replace()` function. It also updates the state name to 'Jharkhand' for rows where the location is 'Jamshedpur'.
4. Changing types to uniform format: The type column has several names for the same type and therefore, it is better to clean it up and make it more uniform. It defines a dictionary `types` that maps different types of areas to a uniform format. It then replaces the values in the 'type' column using the `replace()` function.

Create line plots or time series plots to visualize the overall AQI trend over time.
Creating line plots or time series plots is a common and effective way to visualize the overall trend of the Air Quality Index (AQI) over time. AQI is a numerical scale that communicates how clean or polluted the air currently is and what associated health effects might be a concern for the population.
Plot individual pollutant levels (e.g., PM2.5, PM10, CO) on separate line plots to visualize their trends over time.
Visualizing individual pollutant levels over time using line plots can provide valuable insights into their trends and variations.Interpret the line plots to understand the trends and patterns for each pollutant. Look for spikes, dips, or any noticeable patterns that might indicate changes in pollutant levels over time.
You can enhance the plots by adding more styling, adjusting colors, or incorporating additional information, such as markers for significant events or annotations for specific data points.

Plotting Uttar Pradesh, we see that SO_2 levels have fallen in the state while NO_2 levels have risen. Information about RSPM and SPM can't be concluded since a lot of data is missing.

Use bar plots or stacked bar plots to compare the AQI values across different dates or time periods.
Bar plots provide a visual representation of data, making it easier to compare AQI values across different dates or time periods at a glance. The length of each bar directly corresponds to the magnitude of the AQI, allowing for a quick and intuitive comparison.

Conclusion:
We used matplotlib library to create visualizations that effectively represent the AQI trends and patterns for different pollutants in the city.

FAQs:

Q1] What specific pollutants are included in the dataset, and how are their levels measured?
Q2] Are there any missing or anomalous values in the dataset that need to be addressed before visualization?
Q3] How do the AQI values change over different times of the day or different days of the week?
Q4] How do the trends of different pollutants (e.g., PM2.5 vs. PM10) compare over time in the visualizations?
Q5] What insights can be drawn from the visualizations regarding air quality trends in the city? Are there specific times or conditions when pollutant levels are higher?

****

Part II
Assignment 12

****

Title: Data Aggregation

Problem Statement: Analyzing Sales Performance by Region in a Retail Company.

Dataset: ""Retail_Sales_Data.csv"

Description: The dataset contains information about sales transactions in a retail company. It includes attributes such as transaction date, product category, quantity sold, and sales amount. The goal is to perform data aggregation to analyze the sales performance by region and identify the top-performing regions.

Tasks to Perform:

1. Import the "Retail_Sales_Data.csv" dataset.
2. Explore the dataset to understand its structure and content.
3. Identify the relevant variables for aggregating sales data, such as region, sales amount, and product category.
4. Group the sales data by region and calculate the total sales amount for each region.
5. Create bar plots or pie charts to visualize the sales distribution by region.
6. Identify the top-performing regions based on the highest sales amount.
7. Group the sales data by region and product category to calculate the total sales amount for each combination.
8. Create stacked bar plots or grouped bar plots to compare the sales amounts across different regions and product categories.

Theory:

Data aggregation is the process of combining data from multiple sources or summarizing data from a single source to produce a more concise and meaningful representation of the data. It can be used to identify trends, patterns, and relationships in the data that would not be apparent if the data was analyzed individually.

Data aggregation can be performed on a variety of data types, including numerical data, categorical data, and text data. Some common aggregation operations include:

- **Sum**: Calculates the sum of all values in a column or group of columns.
- **Mean**: Calculates the average of all values in a column or group of columns.
- **Median**: Calculates the middle value in a sorted list of values.
- **Mode**: Calculates the most frequently occurring value in a column or group of columns.
- **Count**: Counts the number of non-null values in a column or group of columns.

Examples of how data aggregation:

- Identifying the top-performing sales regions: A retail company can aggregate sales data by region to identify the regions that are generating the most revenue.
- Tracking website traffic: A website owner can aggregate website traffic data by source to identify the most effective marketing channels.
- Data aggregation is a powerful tool that can be used to gain insights from data and make better decisions.

Benefits of data aggregation:

- **Improved efficiency**: Data aggregation can help businesses to improve their efficiency by automating tasks such as report generation and data analysis.

- **Increased accuracy**: Data aggregation can help businesses to increase the accuracy of their data by reducing the number of errors that occur when data is manually processed.

- **Enhanced decision-making**: Data aggregation can help businesses to make better decisions by providing them with a more complete and accurate view of their data.

1. **Import the "Retail_Sales_Data.csv" dataset.**

It is a csv file, we can use pandas. The read_csv is used to read it into a
DataFrame:

2. E**xplore the dataset to understand its structure and content.**
   - **data.head()**: Assuming that **data** is a pandas DataFrame, **head()** is a method that is used to display the first few rows of the DataFrame. It provides a quick way to inspect the structure and content of the dataset. by default, shows the first 5 rows of the DataFrame. You can specify the number of rows you want to display by passing a number inside the parentheses, like **data.head(10)** to show the first 10 rows.
   - **info()** method in Pandas is used to get a concise summary of a DataFrame, including information about its columns, data types, non-null values, and memory usage. When you call df.info(), it prints a summary report to the console.

3. Identify the relevant variables for aggregating sales data, such as region, sales
amount, and product category.
Aggregating sales data involves summarizing and grouping information to derive
meaningful insights. The choice of relevant variables depends on the specific goals of
analysis and the nature of the business. However, some common variables for aggregating
sales data include:
1. Shopping Mall:
Mall ID or Name: To identify and group sales data based on the shopping mall
where the transactions occurred.
2. Price:
Unit Price: The price of each unit of the product sold.
Total Sales: The total revenue generated from the sale of products.

3. Category:
Product Category: Categorization of products based on their type (e.g., electronics, clothing,

food).
Sales by Category: Aggregating sales data based on the product categories.

4. Group the sales data by region and calculate the total sales amount for each region.
Grouping sales data by region and shopping mall and calculating the total sales amount for each region serves several important purposes in business analysis and decision-making.

5. Create bar plots or pie charts to visualize the sales distribution by region.
Pie charts represent data in a circular graph, where each slice (or sector) of the pie corresponds to a particular category or group. In the context of sales distribution by region,
each slice represents a different region, and the size of each slice indicates the proportion of sales attributed to that region.

A stacked bar plot to compare sales amounts:

1. Plotting:
- Use a plotting library (such as Matplotlib in Python or ggplot2 in R) to create a stacked bar plot.
- The x-axis should represent the regions, and the y-axis should represent the total sales amount.
- Each bar is divided into segments, with each segment representing a different product category.
- The height of each segment corresponds to the sales amount for that specific product category in the respective region.

2. Color Coding:
- Assign different colors to each product category to make it visually clear which part of the bar corresponds to which category.
- The stacked nature of the bars helps in comparing the total sales amounts across regions while also understanding the contribution of each product category within a region.

3. Legend and Labels:
- Include a legend to explain the color-coding of the product categories.
- Label the axes and provide a title to make the plot more informative.

Conclusion:
We have implemented data aggregation, we were able to analyze the sales performance of the retail company by region and identify the top-performing regions. We also identified the top-selling product categories.

FAQs:

Q1] What specific attributes are included in the dataset, and how are they defined? For example, what formats are used for transaction dates, and how is the sales amount recorded?

Q2] Are there any missing values or anomalies in the sales data that need to be addressed before aggregation, such as negative sales amounts or invalid dates?

Q3] What aggregation metrics will be used to analyze sales performance by region, such as total sales amount, total quantity sold, or average transaction value?

Q4] How will you compare the sales performance of different regions? Will you use visualizations like bar charts or pie charts to represent the top-performing regions?

Q5] What insights can be derived from the aggregated data regarding sales trends by region? How might these insights influence business decisions, such as marketing strategies or inventory management?