Register Number														

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamil Nadu
Academic Year: 2024-25 (EVEN)

Set -

Test: FT4 | Date: 29-04-2025
Course Code & Title: 21CSS303T-Data Science | Duration: Two periods
Year& Sem: III Year /VI Sem | Max.Marks:50

Course Articulation Matrix:

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO3	-	-	-	-	1	-	-	-	-	-	-	-
CO4	-	-	-	-	1	-	-	-	-	-	-	-
CO5	-	-	-	-	1	-	-	-	-	-	-	-

Note: CO3 – To identify data manipulation and cleaning techniques using pandas
CO4 – To constructs the Graphs and plots to represent the data using python packages
CO5 – To apply the principles of the data science techniques to predict and forecast the outcome of real-world problem

Part – A (10 x 1 = 10 Marks)

Instructions:
1) Answer **ALL** questions.
2) The duration for answering Part A is **15 minutes** (this sheet will be collected after 15 minutes).
3**) Encircle the correct answer**.

S.No	Question	Marks	BL	CO	PO	PI Code
1	**What is a recommended technique for handling datasets that do not fit into memory?** A. Load the entire data into a list **B. Use streaming or chunking techniques** C. Increase screen resolution D. Use nested loops	1	1	3	5	
2	**What parameter allows merge() to join datasets using an index instead of a column?** A. on_index=True B. use_index=True **C. left_index=True/right_index=True** D. by_index=True	1	1	3	5	
3	**What is the default method of dropna() in pandas?** **A. Drops rows with missing values** B. Replaces missing values with 0 C. Drops columns with duplicates D. Sorts data	1	1	3	5	
4	**What is binning in data preprocessing?** A. Filling missing values **B. Converting continuous variables into categorical bins** C. Merging two datasets D. Sorting data by time	1	2	3	5	
5	**Which of the following techniques can be used to detect outliers or noise in a dataset?** A. Pivoting B. One-hot encoding **C. Z-score or IQR methods** D. Data splitting	1	2	3	5	

6	**Which command is used to create subplots in Matplotlib?** **A. plt.subplots()** B. plt.sub() C. plt.mplot() D. plt.subplotview()	1	1	4	5	
7	**What is Seaborn primarily used for?** A. Connecting APIs B. Creating responsive websites **C. Creating statistical graphics on top of Matplotlib** D. Managing databases	1	1	4	5	
8	**In Seaborn, which function is used to plot pairwise relationships in a dataset?** A. sns.relations() B. sns.matrixplot() **C. sns.pairplot()** D. sns.gridplot()	1	1	4	5	
9	**What function is used to create a scatter plot in Matplotlib?** A. plt.point() **B. plt.scatter()** C. plt.dot() D. plt.circles()	1	2	5	5	
10	**What is the purpose of a histogram?** A. To show relationship between two variables **B. To display data distribution and frequency** C. To visualize classification performance D. To plot trends over time	1	2	5	5	

Register Number														

Set -

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamil Nadu
Academic Year: 2024-25 (EVEN SEM)

Test: FT4 Date:29-04-2025
Course Code & Title: 21CSS303T-Data Science Duration: Two periods
Year& Sem: III Year /VI Sem Max.Marks:50

Part – B (4 x 5 = 20 Marks)
Instructions: Answer **ANY FOUR** Questions

Q. No	Question	Marks	BL	CO	PO	PI Code
11	Explain the difference between reshaping, pivoting, and concatenating datasets using pandas. Ans: ● **Reshaping**: Changing the structure of data (e.g., melt() converts wide to long format). ● **Pivoting**: Converting long data into a wide format (e.g., pivot() makes a column's values into new columns). ● **Concatenating**: Combining multiple datasets along rows or columns (e.g., concat()).	5	2	3	5	
12	Apply binning and standardization to a numerical dataset. Why are these processes important in data preparation? Ans: **Binning** and **standardization** are important data preprocessing techniques to improve the performance of machine learning models. 1. **Binning**: Converts continuous variables into discrete categories to reduce noise and make patterns clearer. o **Example**: import pandas as pd data = pd.Series([1, 5, 7, 9, 10, 14, 20]) bins = [0, 5, 10, 20] labels = ['Low', 'Medium', 'High'] binned_data = pd.cut(data, bins=bins, labels=labels) 2. **Standardization**: Scales data to have a mean of 0 and standard deviation of 1, which helps models converge faster. o **Example**: from sklearn.preprocessing import StandardScaler scaler = StandardScaler() standardized_data = scaler.fit_transform(data.values.reshape(-1, 1)) **Why important?** ● **Binning**: Simplifies complex data, making it easier for models to detect patterns. ● **Standardization**: Ensures that all features are on the same scale, preventing some features from dominating others in models.	5	3	3	5	
13	Compare and contrast the methods of handling missing data. When would you use each? Ans: **Removing Missing Data:** ● **Method**: Drop rows or columns with missing values (dropna()).	5	2	3	5	

	• **Use**: When missing data is small and won't significantly affect the analysis or when data loss is acceptable. **Imputation:** • **Method**: Fill missing values with a constant (e.g., 0), mean, median, mode, or predicted values. • **Use**: When missing data is significant and removing it would lead to loss of important information. **Forward/Backward Fill:** • **Method**: Fill missing values with the previous (or next) available data (ffill(), bfill()). • **Use**: When data is time-series or ordered, and filling missing values with neighboring data is logical. **Predictive Imputation (e.g., using ML):** • **Method**: Use machine learning algorithms to predict missing values based on other features. • **Use**: When missing data is substantial and imputation needs to be more sophisticated.					

14	Demonstrate how to generate a 3D surface plot using Matplotlib. Mention the required imports and customization options.
	Ans:
	import numpy as np
	import matplotlib.pyplot as plt
	from mpl_toolkits.mplot3d import Axes3D
	
	# Create data
	X = np.linspace(-5, 5, 100)
	Y = np.linspace(-5, 5, 100)
	X, Y = np.meshgrid(X, Y)
	Z = np.sin(np.sqrt(X**2 + Y**2))
	
	# Create a figure and 3D axis
	fig = plt.figure()
	ax = fig.add_subplot(111, projection='3d')
	
	# Plot the surface
	ax.plot_surface(X, Y, Z, cmap='viridis')
	
	# Customize labels
	ax.set_xlabel('X axis')
	ax.set_ylabel('Y axis')
	ax.set_zlabel('Z axis')
	
	# Show plot
	plt.show()
	Customization Options:
	cmap: Color map for the surface (e.g., 'viridis', 'plasma').
	
	ax.set_xlabel(), ax.set_ylabel(), ax.set_zlabel(): Customize axis labels.
	
	ax.plot_surface(): You can add more options like edgecolor, alpha for transparency, etc.
		5	3	4	5	

15	Use Seaborn to create a pairplot and customize its style using sns.set_style() on iris dataset. What insights can a pairplot provide? Ans: import seaborn as sns import matplotlib.pyplot as plt # Load the Iris dataset	5	3	5	5	

	iris = sns.load_dataset('iris')
	
	# Set the style for the plot
	sns.set_style('whitegrid')
	
	# Create a pairplot
	sns.pairplot(iris, hue='species')
	
	# Show the plot
	plt.show()
	Customization:
	sns.set_style('whitegrid'): Sets the plot background to white with a grid, which enhances readability.
	
	hue='species': Colors the points according to the different species of the Iris flower, which helps in visualizing the relationship between features across categories.
	
	Insights Provided by a Pairplot:
	Relationships between Variables: Shows scatter plots between each pair of features (e.g., Sepal Length vs. Sepal Width), allowing you to identify correlations.
	
	Distributions: The diagonal plots (histograms or KDEs) show the distribution of each feature.
	
	Cluster Patterns: Helps detect if species clusters are separable based on the features (e.g., the species may be visually separable in certain feature combinations).					

Part – C (2 x 10 = 20 Marks) Instructions: Answer ALL questions.						
Q. No	Question	Marks	BL	CO	PO	PI Code
16 a	Describe and compare various techniques used to clean and prepare raw datasets for analysis. Include examples of handling missing data, standardization, string cleaning, and binning. Give python code examples of each. Ans: **1. Handling Missing Data** ● **Method**: Removing or imputing missing values. ● **Example**: o **Remove rows with missing data**: import pandas as pd df = pd.DataFrame({'A': [1, 2, None, 4], 'B': [5, None, 7, 8]}) df_cleaned = df.dropna() # Remove rows with any missing values o **Impute missing data**: df_imputed = df.fillna(df.mean()) # Replace missing with column mean **2. Standardization (Scaling)** ● **Method**: Scale features to have a mean of 0 and a standard deviation of 1. ● **Example**: from sklearn.preprocessing import StandardScaler	10	2	3	5	

	scaler = StandardScaler() df_scaled = scaler.fit_transform(df[['A', 'B']]) **3. String Cleaning** ● **Method**: Remove or replace unwanted characters, whitespace, or patterns from string columns. ● **Example**: df['Name'] = df['Name'].str.strip().str.replace(r'\d+', '') # Remove digits and whitespace **4. Binning (Discretization)** ● **Method**: Convert continuous variables into categorical bins. ● **Example**: df['Age'] = pd.cut(df['Age'], bins=[0, 18, 35, 50, 100], labels=['Child', 'Young', 'Adult', 'Senior'])					
	(OR)					

16 b	Write and explain a complete data transformation workflow using a sample dataset that includes missing values, text inconsistencies, numeric scaling, and outliers. Give examples using python code.
	1. Ans: **Load the Dataset**:
	
	import pandas as pd
	import numpy as np
	
	# Sample data with missing values, text inconsistencies, and outliers
	data = {
	'Age': [25, np.nan, 22, 35, 110, 29, 200],
	'Salary': [50000, 60000, np.nan, 45000, 120000, 70000, 400000],
	'Name': ['John Doe', ' Jane smith ', 'alice johnson', 'BOB', 'alice', ' john', ' jane'],
	'City': ['New York', 'Los Angeles', 'New York', np.nan, 'San Francisco', 'New York', 'Miami']
	}
	df = pd.DataFrame(data)
	1. Handle Missing Values:
	● **Impute** missing values with appropriate methods (mean for numeric, mode for categorical).
	
	# Impute missing numeric values
	df['Age'] = df['Age'].fillna(df['Age'].mean())
	df['Salary'] = df['Salary'].fillna(df['Salary'].median())
	
	# Impute missing categorical values
	df['City'] = df['City'].fillna(df['City'].mode()[0])
	2. Text Cleaning:
	● **Standardize** text data by removing extra spaces, converting to lowercase, etc.
	
	# Clean and standardize text data
	df['Name'] = df['Name'].str.strip().str.title() # Capitalize names and remove leading/trailing spaces
	df['City'] = df['City'].str.strip().str.title() # Ensure consistent city names
	3. Handle Outliers:
	● **Identify and remove outliers** using the IQR method.
		10	3	3	5	

	# Identifying outliers in 'Age' and 'Salary' using IQR
	Q1_age = df['Age'].quantile(0.25)
	Q3_age = df['Age'].quantile(0.75)
	IQR_age = Q3_age - Q1_age
	lower_bound_age = Q1_age - 1.5 * IQR_age
	upper_bound_age = Q3_age + 1.5 * IQR_age
	
	# Remove outliers
	df = df[(df['Age'] >= lower_bound_age) & (df['Age'] <= upper_bound_age)]
	4. Numeric Scaling (Standardization):
	• **Standardize** numeric columns like 'Age' and 'Salary' to have zero mean and unit variance.
	
	from sklearn.preprocessing import StandardScaler
	
	scaler = StandardScaler()
	df[['Age', 'Salary']] = scaler.fit_transform(df[['Age', 'Salary']])
	Final Dataframe:
	
	print(df)					

17 a	Explain how Matplotlib helps in customizing plots. Describe how to control axes, add labels, legends, annotations, and apply plot styles with examples. Explain the differences and use-cases of different plot types: Line plot, Bar chart, Histogram, Box plot, Scatter plot, and Pair plot. Ans: **Matplotlib provides powerful customization options for creating and enhancing plots. You can control various elements like axes, labels, legends, annotations, and styles. Here's how to customize these features:** **1. Controlling Axes:** • **You can control the axis limits, ticks, and labels using set_xlim(), set_ylim(), and set_xticks()/set_yticks().** **import matplotlib.pyplot as plt** **x = [1, 2, 3, 4]** **y = [1, 4, 9, 16]** **plt.plot(x, y)** **plt.xlim(0, 5) # Set x-axis limit** **plt.ylim(0, 20) # Set y-axis limit** **plt.show()** **2. Adding Labels and Title:** • **xlabel(), ylabel(), and title() are used to add labels and titles.** **plt.plot(x, y)** **plt.xlabel('X-axis Label')** **plt.ylabel('Y-axis Label')** **plt.title('Plot Title')** **plt.show()** **3. Legends:**	10	2	4	5	

- **Use legend() to add a legend to the plot. You can label your plots during plotting and then call legend().**

```
plt.plot(x, y, label='y = x^2')
plt.legend()
plt.show()
```

4. Annotations:

- **Use annotate() to add text or markers to specific points on the plot.**

```
plt.plot(x, y)
plt.annotate('Peak', xy=(2, 4), xytext=(3, 5), arrowprops=dict(facecolor='red', arrowstyle="->"))
plt.show()
```

5. Applying Plot Styles:

- **Use plt.style.use() to apply predefined styles such as ggplot, seaborn, etc.**

```
plt.style.use('ggplot')
plt.plot(x, y)
plt.show()
```

Different Plot Types and Their Use-Cases

1. **Line Plot:**
   - **Use-case: Ideal for showing trends over time or continuous data.**
   - **Example: Plotting stock prices or temperature changes.**

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```

2. **Bar Chart:**
   - **Use-case: Useful for comparing quantities across different categories (categorical data).**
   - **Example: Comparing sales across different products.**

```
plt.bar(['A', 'B', 'C'], [3, 7, 2])
plt.show()
```

3. **Histogram:**
   - **Use-case: Shows the distribution of data, often for continuous numerical data.**
   - **Example: Displaying the distribution of ages in a dataset.**

```
plt.hist([1, 2, 2, 3, 3, 3, 4], bins=4)
plt.show()
```

4. **Box Plot:**
   - **Use-case: Useful for visualizing the distribution of data, including outliers, median, and quartiles.**
   - **Example: Analyzing the spread of test scores.**

```
plt.boxplot([1, 2, 3, 4, 5, 6, 7])
plt.show()
```

5. **Scatter Plot:**
   - **Use-case: Displays relationships between two variables, useful for correlation analysis.**
   - **Example: Visualizing the relationship between height and weight.**

	plt.scatter([1, 2, 3, 4], [1, 4, 9, 16]) **plt.show()** **6. Pair Plot:** **o Use-case: Used for visualizing relationships between multiple variables in a dataset.** **o Example: Showing pairwise relationships in the Iris dataset.** **import seaborn as sns** **iris = sns.load_dataset('iris')** **sns.pairplot(iris, hue='species')** **plt.show()**					
	(OR)					

17 b	Apply advanced Seaborn visualizations to explore patterns in a real dataset. Include pair plots, heatmaps, and style settings. Write a Python program to visualize a 3D surface plot. Explain each component used in the plot.
	Ans:
	
	import seaborn as sns
	import matplotlib.pyplot as plt
	
	# Load dataset
	iris = sns.load_dataset('iris')
	
	# Set style
	sns.set_style("whitegrid")
	
	# Pair plot
	sns.pairplot(iris, hue='species')
	plt.show()
	
	# Heatmap (correlation matrix)
	corr = iris.drop('species', axis=1).corr()
	sns.heatmap(corr, annot=True, cmap='coolwarm')
	plt.title('Feature Correlation Heatmap')
	plt.show()
	
	Explanation:
	sns.set_style(): Sets plot background style.
	
	pairplot(): Shows pairwise relationships and class separation.
	
	heatmap(): Highlights correlations between numeric features.
	
	3D Surface Plot with Matplotlib
	import numpy as np
	import matplotlib.pyplot as plt
	from mpl_toolkits.mplot3d import Axes3D
	
	# Data for the surface
	X = np.linspace(-5, 5, 100)
	Y = np.linspace(-5, 5, 100)
	X, Y = np.meshgrid(X, Y)
	Z = np.sin(np.sqrt(X**2 + Y**2))
	
	# Create 3D plot
	fig = plt.figure()
		10	3	5	5	

	ax = fig.add_subplot(111, projection='3d')
	
	# Plot surface
	surf = ax.plot_surface(X, Y, Z, cmap='viridis')
	
	# Add labels
	ax.set_xlabel('X Axis')
	ax.set_ylabel('Y Axis')
	ax.set_zlabel('Z Axis')
	plt.title('3D Surface Plot')
	plt.show()
	
	Explanation:
	Axes3D: Enables 3D plotting.
	
	meshgrid: Generates grid for surface.
	
	plot_surface: Draws the 3D surface.
	
	cmap: Applies color styling to surface.					

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions: