|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register Number** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



**SRM Institute of Science and Technology**

Set - A

**College of Engineering and Technology**

**School of Computing**

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamil Nadu

# Academic Year: 2024-25 (EVEN)

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 | ----------- method is used to replace, predict, or create the missing values  A.Permutation  B.Deletion  C.Imputation  D.Updation | 1 | 1 | 3 | 5 | 1.4.1 |
| 2 | **melted\_df = pivot\_df.reset\_index().melt(id\_vars='Date', var\_name='City', value\_name='Sales')**  **print(melted\_df)**  What is the purpose of the reset\_index() function in the given code?  A. To rename the index B. To drop the index entirely C. To convert the index into a column D. To sort the DataFrame | 1 | 1 | 3 | 5 | 1.4.1 |
| 3 | What change should be made to the following code to perform column-wise concatenation? **concat\_df = pd.concat([df1, df2], -----------)**  A.concat\_df = pd.concat([df1, df2], axis=2)  B.concat\_df = pd.concat([df1, df2], axis=1)  C.concat\_df = pd.concat([df1, df2], axis=0)  D.concat\_df = pd.concat([df1, df2], axis=’TRUE’) | 1 | 1 | 3 | 5 | 1.4.1 |
| 4 | Which of the following libraries is not primarily involved in handling large volumes of data?  A. Cython B. Numexpr C. Numba  D.Seaborn | 1 | 2 | 3 | 5 | **1.4.1** |
| 5 | Which of the following statements is true regarding data structures?  A) Data structures have the same storage requirements for all types. B) Data structures influence the performance of CRUD operations (create, read, update, and delete). C) Data structures only affect the storage and not the performance of operations. D) Data structures do not affect the performance of CRUD operations. | 1 | 2 | 3 | 5 | 1.4.1 |
| 6 | Which of the following is the correct syntax for creating a subplot with 2 rows and 3 columns in the first position?  A) plt.subplot(2, 3, 0) B) plt.subplot(3, 2, 1) C) plt.subplot(2, 3, 1) D) plt.subplot(1, 2, 3) | 1 | 1 | 4 | 5 | 1.4.1 |
| 7 | Which parameter is used to create 100 evenly spaced values between 0 and 10?   1. np.linspace(0, 10, 100) B) np.linspace(0, 100, 10) C) np.linspace(0, 10, 100)   D)np.linspace(10, 100, 0) | 1 | 1 | 4 | 5 | 1.4.1 |
| 8 | What does the following matplotlib code do?  plt.annotate('Peak Point',  xy=(6, 15),  xytext=(4, 17),  fontsize=12, color='blue')  A) Adds the text *Peak Point* at point (6, 15) in blue font. B) Displays *Peak Point* directly at point (6, 15) without any arrow. C) Adds a blue annotation with the text *Peak Point*, pointing from (4, 17) to (6, 15). D) Draws a blue line between (4, 17) and (6, 15) and places *Peak Point* on it. | 1 | 1 | 4 | 5 | 1.4.1 |
| 9 | Which of the following best describes the purpose of GridSpec in data visualization?  A) Group data by a categorical variable and create subplots for each category. B) Visualize the relationship between two variables along with their distributions. C) Create custom grid layouts for organizing multiple subplots. D) Plot the relationships between all numeric column pairs in a DataFrame. | 1 | 2 | 5 | 5 | **1.4.1** |
| 10 | **What does the height of a bar in a histogram represent?**  A) The total number of data points B) The frequency of values within the interval C) The range of values in the data D) The cumulative frequency of the data | 1 | 2 | 5 | 5 | 1.4.1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register Number** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



**SRM Institute of Science and Technology**

Set -

**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 | Write a Python program to do the following:   1. Replace all missing (NaN) values in the **Name** column with the string 'Unknown'. 2. Replace all missing (NaN) values in the **Age** column with the **mean** of the available age values. 3. Add a new column named **City** and fill it with any default or custom city names for each student. 4. Print the final cleaned DataFrame.  |  |  | | --- | --- | | **Name** | **Age** | | Bob | 24 | | NaN | 25 | | Sweety | Nan | | Rita | 26 |   import pandas as pd  import numpy as np  # Create the initial DataFrame  data = {  'Name': ['Bob', np.nan, 'Sweety', 'Rita'],  'Age': [24, 25, np.nan, 26]  }  df = pd.DataFrame(data)  # Step 1: Replace NaN in 'Name' with 'Unknown'  df['Name'].fillna('Unknown', inplace=True)  # Step 2: Replace NaN in 'Age' with the mean age  mean\_age = df['Age'].mean()  df['Age'].fillna(mean\_age, inplace=True)  # Step 3: Add a new column 'City' with default value (e.g., 'Delhi')  df['City'] = ['Delhi', 'Mumbai', 'Pune', 'Chennai'] # You can customize this  # Display the cleaned DataFrame  print(df) | 5 | 2 | 3 | 5 | 1.4.1 |
| 12 | What is the purpose of the pandas.merge() function in Python?(2.5) Explain its use with an example by merging two DataFrames on a common column using python (2.5) ?  **Pandas merge function**  **pandas.merge connects rows in DataFrames based on one or more keys. This will be familiar to users of SQL or other relational databases, as it implements database *join* operations**  **Sample Example**  # Create the first DataFrame: student details  df\_students = pd.DataFrame({  'Student\_ID': [1, 2, 3],  'Name': ['Alice', 'Bob', 'Charlie']  })  # Create the second DataFrame: student scores  df\_scores = pd.DataFrame({  'Student\_ID': [1, 2, 3],  'Marks': [85, 90, 88]  })  # Merge the two DataFrames on the 'Student\_ID' column  merged\_df = pd.merge(df\_students, df\_scores, on='Student\_ID')  # Print the merged DataFrame  print**(merged\_df)** | 5 | 3 | 3 | 5 | 1.4.1 |
| 13 | What is **reshaping** in pandas, and what are the main methods used for reshaping a DataFrame?  **Reshaping in pandas refers to changing the structure or layout of a DataFrame**  **Pivoting Data (pivot() and pivot\_table())**   * **Pivoting rearranges data by turning unique values into columns.** * **Melting Data (melt())** * **The opposite of pivoting – it converts wide data into long format** * **Stacking (stack())** * **Converts columns into a hierarchical index (multi-index rows).** | 5 | 2 | 3 | 5 | 1.4.1 |
| 14 | Write a Matplotlib program to rotate tick labels in x and y axis ?  import matplotlib.pyplot as plt  # Sample data  x = [1, 2, 3, 4, 5]  y = [10, 20, 25, 30, 40]  # Create the plot  plt.plot(x, y)  # Set labels  plt.xlabel('X Axis')  plt.ylabel('Y Axis')  plt.title('Rotating Tick Labels Example')  # Rotate x-axis tick labels by 45 degrees  plt.xticks(rotation=45)  # Rotate y-axis tick labels by 90 degrees  plt.yticks(rotation=90)  # Show the plot  plt.tight\_layout() # Adjust layout to prevent label cutoff  plt.show() | 5 | 3 | 4 | 5 | 1.4.1 |
| 15 | What are the various types of annotations in Matplotlib? Give the syntax of annotation .   * **Text annotations** are used to add explanatory or descriptive text to specific points, regions, or features within a plot. * **Marker annotations** involve placing markers or symbols on specific points of interest within a plot to highlight or provide additional information about those points. * **Callouts** refer to a specific type of annotation that uses visual elements like arrows, lines, or text to draw attention to a particular area or feature within a plot. | 5 | 3 | 5 | 5 | 1.4.1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part – C (2 x 10 = 20 Marks)**  Instructions: Answer ALL questions. | | | | | | |
| Q.  No | Question | Marks | BL | CO | PO | PI  Code |
|  | (i)Explain outliers and their types. (5 marks)  **Outlier Noise or Outliers are the data points which deviate significantly from the norm.**  **Outliers can be single data points, or a subset of observations called a collective outlier.**  **The outlier data points can greatly impact the accuracy and reliability of statistical analyses and machine learning models.**  **Outliers can also be called abnormalities, discordant, deviants, or anomalies.**    Types of outlier  **Global outliers**   * **Global outliers are isolated data points that are far away from the main body of the data.** * **They are often easy to identify and remove.**   **Contextual outliers**   * **Contextual outliers are data points that are unusual in a specific context but may not be outliers in a different context.**   **They are often more difficult to identify and may require additional information or domain knowledge to determine their significance**  (ii)We create a panda DataFrame from a dictionary that holds the student data. student's ID, first name, last name, and grade(5Marks)  a. Combine First Name and Last Name into a new column called Full Name.  b. Display only the First Name and Grade columns.  c. Identify and display students who received a grade 'A'.  d. Create a new column Updated Grade, where every 'B' grade is replaced with 'A'.  import pandas as pd  # Create a dictionary of student data  data = {  'ID': [101, 102, 103, 104, 105],  'First Name': ['John', 'Jane', 'Jim', 'Jill', 'Jack'],  'Last Name': ['Doe', 'Smith', 'Beam', 'Hill', 'Black'],  'Grade': ['A', 'B', 'A', 'C', 'B']  }  # Create a DataFrame from the dictionary  students = pd.DataFrame(data)  # a. Combine First Name and Last Name into a new column called Full Name  students['Full Name'] = students['First Name'] + ' ' + students['Last Name']  print(" DataFrame with Full Name:\n", students, "\n")  # b. Display only the First Name and Grade columns  print("First Name and Grade columns:\n", students[['First Name', 'Grade']], "\n")  # c. Identify and display students who received a grade 'A'  grade\_A\_students = students[students['Grade'] == 'A']  print("Students with Grade 'A':\n", grade\_A\_students, "\n")  # d. Create a new column Updated Grade, where every 'B' grade is replaced with 'A'  students['Updated Grade'] = students['Grade'].replace('B', 'A')  print("DataFrame with Updated Grade:\n", students) | 10` | Q | 3 | 5 | 1.4.1 |
| **(OR)** | | | | | | |
| 16 b | (i)Explain about standardization with its type. (5 mark)   * **Standardization is a common preprocessing technique in data science that transforms numerical data to have a mean of 0 and a standard deviation of 1.** * **This is particularly useful when dealing with features that have different scales or units, as it ensures that all features contribute equally to the model.** * **Equalizes Feature Importance: Standardization prevents features with larger magnitudes from dominating the model, ensuring that all features are treated fairly.** * **Improves Model Performance: Many machine learning algorithms, especially those based on distance or gradient calculations, benefit from standardized data.** * **Compatibility with Certain Algorithms: Some algorithms, like K-Nearest Neighbors and Support Vector Machines, assume standardized data.** * **Z-score normalization is a data preprocessing technique that transforms numerical data to have a mean of 0 and a standard deviation of 1. This is particularly useful when dealing with features that have different scales or units, as it ensures that all features contribute equally to the model.** * The formula used is:   z = (x - mean) / standard\_deviation   * **where:**   + z is the normalized value.   + x is the original value.   + mean is the mean of the dataset.   + standard\_deviation is the standard deviation of the dataset. * **Min-max normalization is a data preprocessing technique that scales numerical data to a specific range, typically between 0 and 1. It's useful when you want to preserve the relative distances between data points while ensuring that all features have a similar scale.** * The formula used is:   x\_scaled = (x - min(x)) / (max(x) - min(x))  where:   * + x\_scaled is the normalized value.   + x is the original value.   + min(x) is the minimum value in the dataset.   + max(x) is the maximum value in the dataset.   (ii)You are provided with a dataset of student records in a Python program using the panda’s library. The dataset includes the following fields: ID, First Name, Last Name, and Grade. Write a Python program to perform the following tasks (5 marks)  a. Convert the First Name column to uppercase and store it in a new column Full Name Upper.  b. Create a new column name is Formatted Info which contain the information in the format : **"Full Name, Grade: <Grade>".**  c. Count and display the number of students who originally received a grade 'B'.  d. Calculate and display the length (number of characters) of each student's full name in a new column called Full Name Length  import pandas as pd  # Sample dataset  data = {  'ID': [1, 2, 3, 4],  'First Name': ['Alice', 'Bob', 'Charlie', 'Diana'],  'Last Name': ['Smith', 'Jones', 'Brown', 'Taylor'],  'Grade': ['A', 'B', 'B', 'C']  }  # Create DataFrame  df = pd.DataFrame(data)  # a. Convert First Name to uppercase and store in new column 'Full Name Upper'  df['Full Name Upper'] = df['First Name'].str.upper() + " " + df['Last Name'].str.upper()  # b. Create 'Formatted Info' column  df['Formatted Info'] = df['First Name'] + " " + df['Last Name'] + ", Grade: " + df['Grade']  # c. Count and display number of students with grade 'B'  count\_b = (df['Grade'] == 'B').sum()  print(f"Number of students who received grade 'B': {count\_b}")  # d. Calculate and store the length of full names  df['Full Name Length'] = (df['First Name'] + " " + df['Last Name']).str.len()  # Display the updated DataFrame  print("\nUpdated DataFrame:")  print(df) | 10 | 3 | 3 | 5 | 1.4.1 |
|  | | | | | | |
| 17 a | Explain the features of seaborn(5)   * **Statistical Graphics:** Seaborn is specifically designed for creating statistical graphics, providing built-in functions for common visualizations like scatter plots, line plots, histograms, and more. This makes it easier to create visually appealing and informative plots for data analysis. * **Data Visualization Themes:** Seaborn offers pre-defined styles and themes that can quickly change the overall appearance of your plots. This helps create consistent and aesthetically pleasing visualizations without requiring extensive customization. * **Integration with Pandas and NumPy:** Seaborn seamlessly integrates with Pandas and NumPy, making it easy to work with dataframes and arrays directly. This simplifies the workflow and reduces the amount of code needed for data analysis and visualization. * **FacetGrid and Pair Plots:** Seaborn provides FacetGrid for grouping data and creating subplots based on categorical variables. This is useful for comparing distributions or relationships across different groups. Pair plots allow you to visualize the relationships between all pairs of numeric columns in a DataFrame, helping you identify correlations and patterns. * **Customization and Flexibility:** While Seaborn provides a high-level interface, it's built on top of Matplotlib, giving you access to its extensive customization options. This allows you to fine-tune your plots to meet your specific needs. * **Ease of Use:** Seaborn's API is designed to be user-friendly and intuitive, making it easier to learn and use compared to Matplotlib. Its documentation is also well-written and provides clear examples.   Illustrate the Python program uses Matplotlib to compare sales of two products over six months using a line chart. Each product is represented with a unique marker and labeled for clarity. A legend is added to distinguish between the products visually. Ticks are used for specify the range of values(5)  import matplotlib.pyplot as plt  # Sample data  months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']  product\_a\_sales = [20, 35, 30, 35, 27, 40]  product\_b\_sales = [25, 32, 34, 20, 25, 30]  # Create the plot  plt.plot(months, product\_a\_sales, marker='o', label='Product A') # Circle markers  plt.plot(months, product\_b\_sales, marker='s', label='Product B') # Square markers  # Title and labels  plt.title('Sales Comparison Over 6 Months')  plt.xlabel('Month')  plt.ylabel('Sales')  # Set y-axis ticks at every 5 units  plt.yticks(range(0, 51, 5))  # Add a legend  plt.legend()  # Show the plot  plt.grid(True)  plt.show() | 10 | 2 | 4 | 5 | 1.41 |
| **(OR)** | | | | | | |
| 17 b | Give your own Seaborn library example for a 3D line plot, 3D scatter plot, and 3D surface plot. Draw the output for each example.      **import seaborn as sns # Not directly used for surface plots**  **import matplotlib.pyplot as plt**  **from mpl\_toolkits.mplot3d import Axes3D**  **import numpy as np**  **# Sample data (ensure x and y are 2D for surface plot)**  **x = np.linspace(0, 5, 10) # Create equally spaced points from 0 to 5 with 10 elements**  **y = np.linspace(0, 5, 10)**  **X, Y = np.meshgrid(x, y) # Create a 2D grid from x and y for surface evaluation**  **def f(x, y):**  **return x\*\*2 + y\*\*2 # Replace with your desired function**  **# Calculate z values based on the function**  **z = f(X, Y)**  **# Create a 3D figure and axes**  **fig = plt.figure(figsize=(8, 6)) # Adjust figure size as needed**  **ax = fig.add\_subplot(111, projection='3d')**  **surf = ax.plot\_surface(X, Y, z, cmap='viridis', linewidth=0, antialiased=True) # Adjust colormap**  **ax.set\_xlabel('X-axis')**  **ax.set\_ylabel('Y-axis')**  **plt.title('3D Surface Plot’)**  **# Customize viewing angle (optional)**  **ax.view\_init(elev=20, azim=45) # Adjust elevation and azimuth angles**  **# Show the plot**  **plt.show()** | 10 | 3 | 5 | 5 | 1.4.1 |

**Course Outcome (CO) and Bloom’s level (BL) Coverage in Questions**

|  |  |
| --- | --- |
|  |  |