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

SET –
Answer Key

Part – A (10 x 1 = 10 Marks)

S.No	Question	Marks
1	a) Data Collection → Data Cleaning → Data Transformation → Data Analysis	1
2	b) Replaces all NaN values with 0	1
3	d) merge()	1
4	a) To combine datasets horizontally or vertically	1
5	b) Remove non-numeric values or replace them with NaN	1
6	b) To add text annotations to specific points on the plot	1
7	c) Pair plot	1
8	a) plt.style.use('seaborn-darkgrid')	1
9	b) sns.histplot()	1
10	d) Plots a scatter plot matrix grouped by species	1

Q. No	**Part – B** (4 x 5 = 20 Marks) Instructions: Answer **ANY FOUR**	Marks
11	**Discuss the general programming tips to deal with large data sets.** ● Don't reinvent the wheel. Use tools and libraries developed by others ● Get the most out of your hardware. Your machine is never used to it full potential; with simple adaptions you can make it work harder. ● Reduce the computing need. Slim down your memory and processing needs as much as possible.	5

12	**When merging two DataFrames in pandas that have columns with the same name, how can you ensure the column names are distinguishable?**
	Use the suffixes parameter in the merge() function to add distinguishing suffixes to overlapping column names.
	`import pandas as pd`
	`df1 = pd.DataFrame({'ID': [1, 2], 'Value': [10, 20]})`
	`df2 = pd.DataFrame({'ID': [1, 2], 'Value': [30, 40]})`
	`merged_df = pd.merge(df1, df2, on='ID', suffixes=('_left', '_right'))`
	`print(merged_df)`
		5

13	**Given the dataset data ={'Ages': [3, 18, 22, 10, 25, 29, 34, 14, 40, 45, 50, 55, 60, 12, 65, 70, 75, 80, 85]}, categorize the continuous Ages values into the groups of children, young, middle, and elder. Define appropriate age ranges for each category and implement the conversion.** import pandas as pd data = {'Ages': [3, 18, 22, 10, 25, 29, 34, 14, 40, 45, 50, 55, 60, 12, 65, 70, 75, 80, 85]} df = pd.DataFrame(data) bins = [0, 12, 24, 59, 100] labels = ['Child', 'Young', 'Middle', 'Elder'] df['Category'] = pd.cut(df['Ages'], bins=bins, labels=labels) print(df)	5
14	**Compare a box plot and a histogram, highlighting their use cases and strengths.** **Box Plot:** ● Displays the distribution of data and highlights outliers. ● Ideal for comparing multiple datasets. **Histogram:** ● Shows the frequency distribution of data values. ● Useful for understanding the shape of the data (e.g., skewness).	5
15	**How can you control the line properties (e.g., color, style, and width) of a chart in Matplotlib. Write the python code and explain.** import matplotlib.pyplot as plt x = [1, 2, 3, 4, 5] y = [10, 20, 15, 25, 30] plt.plot(x, y, color='red', linestyle='--', linewidth=2) plt.title("Line Properties Example") plt.xlabel("X-axis") plt.ylabel("Y-axis") plt.show() ● color: Sets the line color. ● linestyle: Controls the style of the line (e.g., dashed, solid). ● linewidth: Adjusts the thickness of the line.	5

Q. No	**Part – C (2 x 10 = 20 Marks)** Instructions: Answer ALL questions.	Marks
16 a	● **Missing Data**: Fill missing sales values with the median (robust to outliers). Drop rows if there are very few missing values. Example Code:` df['Sales'] = df['Sales'].fillna(df['Sales'].median()) ● **Irregular Formats**: Convert all dates into a uniform format (YYYY-MM-DD) using pd.to_datetime. Example Code: df['Date'] = pd.to_datetime(df['Date'], errors='coerce') ● **Duplicate Records**: Remove rows where Product, Region, and Date are duplicated, keeping the first occurrence: Example Code: df = df.drop_duplicates(subset=['Product', 'Region', 'Date'], keep='first') ● **Irrelevant Data**: Drop unnecessary or irrelevant columns like Transaction ID Example Code:	**10**

	df = df.drop(columns=['Transaction ID']) ● **Outliers**: Identify outliers in `Sales` using the interquartile range (IQR) Example Code: Q1 = df['Sales'].quantile(0.25) Q3 = df['Sales'].quantile(0.75) IQR = Q3 - Q1 lower_bound = Q1 - 1.5 * IQR upper_bound = Q3 + 1.5 * IQR df = df[(df['Sales'] >= lower_bound) & (df['Sales'] <= upper_bound)] ● **Categorical Inconsistencies**: Standardize inconsistent product names using a mapping dictionary: Example Code: product_mapping = {'Appl': 'Apple', 'Bananaa': 'Banana'} df['Product'] = df['Product'].replace(product_mapping) ● **Merging**: Load the profit margins dataset and merge with the sales data on Product and Region Example Code: profit_data = pd.read_csv('profit_margins.csv') df = pd.merge(df, profit_data, on=['Product', 'Region'], how='inner') . ● **Final Quality Checks** ● **Ensure all columns have consistent data types:** df['Sales'] = df['Sales'].astype(float) df['Date'] = pd.to_datetime(df['Date']) ● **Verify no missing or inconsistent values remain:** print(df.isnull().sum())	
16 b	**Output of pivot_df = df.pivot(index='Date', columns='Product', values='Sales')** The pivot function reshapes the DataFrame by specifying: ● index: Rows of the resulting DataFrame (Date here). ● columns: Columns of the resulting DataFrame (Product here). ● values: Data to fill the cells (Sales here). **Output:** Product Apple Banana Date 2023-01-01 100 150 2023-01-02 200 50 **Discussion:** ● The rows are indexed by Date. ● The columns are determined by unique values in Product. ● The values in the cells are taken from the Sales column. **2. Output of stacked_df = df.stack()** The stack function compresses columns into a hierarchical index at the row level. **Output:** 0 Date 2023-01-01	**10**

	Product Apple Region North Sales 100 1 Date 2023-01-01 Product Banana Region North Sales 150 2 Date 2023-01-02 Product Apple Region South Sales 200 3 Date 2023-01-02 Product Banana Region South Sales 50 dtype: object **Discussion:** ● Each row is identified by a combination of the original row index (e.g., 0, 1, etc.) and the column name (e.g., Date, Product, Region, Sales). ● The DataFrame is reshaped into a Series with a multi-level index. **3. Output of stacket_pivot = pivot_df.stack()** The stack function on pivot_df moves the columns (Product) back into the row index. **Output:** Date Product 2023-01-01 Apple 100 Banana 150 2023-01-02 Apple 200 Banana 50 dtype: int64 **Discussion:** ● The columns (Apple, Banana) are turned into a new hierarchical index level under Date. ● The resulting structure is a Series, with the multi-level index representing the combination of Date and Product.	
17 a	**Functionalities of the Seaborn Library** Seaborn is a Python library built on top of Matplotlib that simplifies creating informative and aesthetically pleasing statistical graphics. It provides high-level interfaces for creating attractive and complex visualizations. **Key Features:** 1. Theme Customization: Automatically styles plots for aesthetics. 2. Dataset-Oriented: Works efficiently with DataFrames and arrays. 3. Built-In Statistical Analysis: Includes options for regression, distribution fitting, and more. 4. Integration with Pandas: Seamless handling of DataFrame columns. 5. Wide Range of Plot Types: Includes pair plots, box plots, violin plots, heatmaps, and more. **Examples of Key Visualizations** **1. Pair Plot** A pair plot is useful for visualizing pairwise relationships in a dataset, especially numerical features. It provides scatterplots for relationships and histograms for univariate distributions.	10

	Code Example:
	import seaborn as sns
	import pandas as pd
	
	# Example Dataset
	data = sns.load_dataset('iris')
	
	# Pair Plot
	sns.pairplot(data, hue='species')
	Functionality:
	● Displays scatter plots between every pair of numerical columns.
	● Includes diagonal histograms to visualize the distribution of each feature.
	● Uses hue to color the data points based on a categorical column (species).
	
	2. Box Plot
	A box plot summarizes the distribution of a dataset through five-number summary statistics: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It also highlights potential outliers.
	Code Example:
	# Box Plot
	sns.boxplot(x='species', y='sepal_width', data=data)
	Functionality:
	● Displays distributions and compares groups (e.g., species) for a numerical column (sepal_width).
	● Identifies outliers as points outside the whiskers.
	● Can be enhanced with swarm plots to overlay individual data points.
	
	3. Histogram
	A histogram visualizes the distribution of a single numerical variable by grouping data into bins.
	Code Example:
	# Histogram
	sns.histplot(data['sepal_length'], kde=True, bins=20)
	Functionality:
	● Shows the frequency of data points within specified bins.
	● Optionally overlays a kernel density estimate (KDE) curve for a smoothed representation of the distribution.
	● Parameters like bins control the granularity of the visualization.	

17 b	**Creating and Interpreting 3D Surface Plots** **Example Program (5 marks for the correct usage of add.subplot, polt.surface and set_xlabel methods)** 3D surface plots are a type of visualization used to represent three-dimensional data where the z-axis corresponds to the dependent variable, and the x and y axes represent independent variables. These plots are useful for exploring relationships between variables and identifying patterns or trends. import numpy as np import matplotlib.pyplot as plt from mpl_toolkits.mplot3d import Axes3D # Generate data	10

```
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 the 3D plot
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
surface = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')

# Add labels and a color bar
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
plt.colorbar(surface, ax=ax, shrink=0.5, aspect=10)
plt.title('3D Surface Plot Example')
plt.show()
```

Scenarios Where 3D Surface Plots Are Beneficial (5 Mark)

1. **Data Exploration in Engineering:**
    - Analyzing stress, temperature, or pressure distribution over a 2D plane.
    - Example: Surface temperature of a material under specific conditions.
2. **Optimization Problems:**
    - Visualizing cost functions in machine learning or operations research.
    - Example: Understanding the shape of loss functions during model training.
3. **Geographic and Environmental Data:**
    - Representing terrain elevation or pollution levels.
    - Example: A surface plot of altitude over a geographical region.
4. **Physics and Mathematics:**
    - Illustrating functions or mathematical surfaces.
    - Example: Visualizing wave functions or potential fields.
5. **Economics and Finance:**
    - Exploring relationships between variables like interest rates, risk, and returns.
    - Example: 3D visualization of portfolio optimization.