# CUET-UG Information Practices Sample Paper - 5

**Duration: 1 Hour** **Maximum Marks: 250**

## Instructions

- This paper contains a total of 50 Multiple Choice Questions.
- Each correct answer carries **+5 marks**.
- Each incorrect answer carries **-1 mark**.
- No negative marking for unattempted questions.

**Q1.** Given a Series $S$ as follows: $S = pd.Series([10, 20, 30, 40], index = ['a','b','c','d'])$. What will be the output of $print(S[S > 25] * 2)$?

(A) c 60, d 80

(B) c 30, d 40

(C) a 20, b 40

(D) 60, 80

**Q2.** Consider the following Python code:

```
import pandas as pd
df = pd.DataFrame({'A' : [1,2],'B' : [3,4]})
df['C'] = df['A'] + df['B']
df.iloc[0,2] = 10
```

What is the value of $df.at[0,'C']$?

(A) 4

(B) 10

(C) 7

(D) NaN

**Q3.** Which of the following attributes of a Pandas DataFrame returns a tuple representing the dimensionality (rows, columns)?

(A) size

(B) ndim

(C) shape

(D) axes

**Q4.** What is the correct syntax to delete the column 'Salary' from a DataFrame $df$ permanently?

(A) $df.drop('Salary', axis = 0)$

(B) $df.drop('Salary', axis = 1, inplace = True)$

(C) $df.remove('Salary')$

(D) $del\ df['Salary']$

**Q5.** Given a DataFrame $df$ with 5 rows and 3 columns. What will $df.count()$ return?

(A) A single integer 15

(B) A Series with 5 entries

(C) A Series with 3 entries

(D) A tuple (5, 3)

**Q6.** A Series $obj$ has the following data: $[10, NaN, 30, 40]$. What is the result of $obj.count()$?

(A) 4

(B) 3

(C) 2

(D) Error

**Q7.** Identify the correct statement regarding $df.loc$ and $df.iloc$:

(A) $loc$ uses integer positions, $iloc$ uses labels.

(B) Both are used to filter rows only.

(C) *loc* includes the last element in a slice, *iloc* excludes it.

(D) *loc* is faster than *iloc* for numerical indexing.

**Q8.** To display the last 3 rows of a DataFrame *df* with 100 rows, which command is most efficient?

(A) $df.head(-3)$

(B) $df.tail(3)$

(C) $df.iloc[0:3]$

(D) $df.tail(-3)$

**Q9.** What will be the output of the following code?

```
import pandas as pd
s = pd.Series([1, 2, 3], index = [2, 1, 0])
print(s.loc[1])
```

(A) 1

(B) 2

(C) 3

(D) Error

**Q10.** Which method is used to export a DataFrame to a file that uses ';' as a delimiter?

(A) $df.to\_csv('file.csv', sep = ';')$

(B) $df.save\_csv('file.csv', delimiter = ';')$

(C) $df.export('file.csv', sep = ';')$

(D) $df.to\_csv('file.csv', tab = ';')$

**Q11.** If a DataFrame *df* is created using a 2D Dictionary, the outer keys of the dictionary become the _____ of the DataFrame.

(A) Row Index

(B) Column Names

(C) Data Values

(D) Series Names

**Q12.** Given $df = pd.DataFrame([[1,2],[3,4]], columns = ['X','Y'])$. What does $df.rename(columns = \{'X' : 'Alpha'\})$ return?

(A) $df$ with column 'X' changed to 'Alpha' permanently.

(B) A new DataFrame with column 'X' changed to 'Alpha'.

(C) An error because 'inplace' is not True.

(D) Nothing, it modifies the original object.

**Q13.** Which of the following is used to check if a DataFrame is empty?

(A) $df.isempty()$

(B) $df.empty$

(C) $df.isnull()$

(D) $len(df) == 0$

**Q14.** To change the value of the 2nd row, 1st column in a DataFrame $df$ to 50 using integer-based indexing:

(A) $df.iloc[2,1] = 50$

(B) $df.iloc[1,0] = 50$

(C) $df.iat[2,1] = 50$

(D) $df.loc[1,0] = 50$

**Q15.** What does the $T$ attribute of a DataFrame do?

(A) Truncates the data

(B) Transposes rows and columns

(C) Calculates the Total

(D) Tests for missing values

**Q16.** Which parameter in $read\_csv()$ is used to skip the first $n$ rows of the file?

(A) $skiprows = n$

(B) $header = n$

(C) $drop = n$

(D) $ignore = n$

**Q17.** A Series $S$ is created as: $S = pd.Series(5, index = [0, 1, 2, 3])$. What is the value of $S[2]$?

(A) 2

(B) 5

(C) NaN

(D) 0

**Q18.** Which SQL clause is used to filter records after an aggregate function has been applied?

(A) WHERE

(B) ORDER BY

(C) HAVING

(D) GROUP BY

**Q19.** Consider a table 'Sales' with columns (ID, Amount, Region). To find the total amount per region where the total is greater than 5000:

(A) $SELECT\ Region, SUM(Amount)\ FROM\ Sales\ WHERE\ SUM(Amount) > 5000\ GROUP\ BY\ Region$;

(B) $SELECT\ Region, SUM(Amount)\ FROM\ Sales\ GROUP\ BY\ Region\ HAVING\ SUM(Amount) > 5000$;

(C) $SELECT\ Region, SUM(Amount)\ FROM\ Sales\ GROUP\ BY\ Region\ WHERE\ Amount > 5000$;

(D) $SELECT\ SUM(Amount)\ FROM\ Sales\ GROUP\ BY\ Region$;

**Q20.** How many records will be returned in a Cartesian Product (Cross Join) of Table A (10 rows) and Table B (5 rows)?

(A) 15

(B) 10

(C) 50

(D) 5

**Q21.** The result of an Equi-Join between two tables with $n$ and $m$ rows and a common column will have at most _____ rows.

(A) $n + m$

(B) $n \times m$

(C) $max(n, m)$

(D) $min(n, m)$

**Q22.** Which command is used to arrange the result set in descending order of 'Salary' and then ascending order of 'Name'?

(A) $ORDER\ BY\ Salary\ DESC, Name\ ASC$

(B) $ORDER\ BY\ Salary, Name\ DESC$

(C) $SORT\ BY\ Salary\ DESC, Name$

(D) $ORDER\ BY\ DESC\ Salary, ASC\ Name$

**Q23.** The GROUP BY clause is always placed _____ the WHERE clause and _____ the HAVING clause.

(A) Before, After

(B) After, Before

(C) Before, Before

(D) After, After

**Q24.** Which aggregate function ignores NULL values?

(A) $COUNT(*)$

(B) $SUM()$

(C) Both (A) and (B)

(D) None of the above

**Q25.** In a join query, if the joining condition is missing, the result is a:

(A) Natural Join

(B) Outer Join

(C) Cartesian Product

(D) Equi Join

**Q26.** To find the number of different departments in an 'Employee' table, use:

(A) $SELECT\ COUNT(Dept)\ FROM\ Employee$;

(B) $SELECT\ COUNT(DISTINCT\ Dept)\ FROM\ Employee$;

(C) $SELECT\ DISTINCT\ COUNT(Dept)\ FROM\ Employee$;

(D) $SELECT\ SUM(Dept)\ FROM\ Employee$;

**Q27.** What is the output of $SELECT\ ROUND(15.768, 2)$; in MySQL?

(A) 15.76

(B) 15.77

(C) 16

(D) 15.8

**Q28.** Which function returns the position of the first occurrence of a substring in a string?

(A) $LOCATE()$

(B) $INSTR()$

(C) $POSITION()$

(D) All of the above

**Q29.** What will be the result of $SELECT\ MOD(11, 3)$;?

(A) 3

(B) 2

(C) 3.66

(D) 1

**Q30.** Which function is used to extract the day of the week from a date?

(A) $DAYNAME()$

(B) $WEEKDAY()$

(C) Both (A) and (B)

(D) $DATEWEEK()$

**Q31.** The output of $SELECT\ SUBSTR('Informatics', 3, 4)$; is:

(A) 'form'

(B) 'info'

(C) 'orma'

(D) 'matice'

**Q32.** What does $SELECT\ LENGTH('Database\ ')$; (note the space at the end) return?

(A) 8

(B) 9

(C) 10

(D) Error

**Q33.** Which function returns the current system date and time?

(A) $DATE()$

(B) $CURDATE()$

(C) $NOW()$

(D) $SYSDATE$

**Q34.** The result of $SELECT\ LCASE(MID('PYTHON', 2, 3))$; is:

(A) 'yth'

(B) 'pyt'

(C) 'ytho'

(D) 'ytho'

**Q35.** Which function is used to specify the label for the x-axis in Matplotlib?

(A) $plt.xname()$

(B) $plt.xlabel()$

(C) $plt.labelx()$

(D) $plt.set\_x()$

**Q36.** To display a horizontal bar chart, which function is used?

(A) $plt.bar()$

(B) $plt.hbar()$

(C) $plt.barh()$

(D) $plt.plot(kind = 'barh')$

**Q37.** Which parameter in $plt.pie()$ is used to "pop out" a slice from the center?

(A) $separate$

(B) $explode$

(C) $shadow$

(D) $pop$

**Q38.** In Matplotlib, $plt.legend()$ is used to:

(A) Show the title of the graph

(B) Identify different data series in the plot

(C) Change the color of the bars

(D) Save the figure

**Q39.** Which type of plot is best suited for showing frequency distributions?

(A) Line chart

(B) Pie chart

(C) Histogram

(D) Scatter plot

**Q40.** A _____ is a set of one or more columns that can uniquely identify a record in a table, but is not chosen as the Primary Key.

(A) Candidate Key

(B) Foreign Key

(C) Alternate Key

(D) Super Key

**Q41.** In Relational Algebra, the Selection ($\sigma$) operation is used to:

(A) Pick specific columns

(B) Pick specific rows based on a condition

(C) Combine two tables

(D) Remove duplicate rows

**Q42.** Which of the following is a Referential Integrity constraint?

(A) Primary Key

(B) Unique Key

(C) Foreign Key

(D) Default

**Q43.** The _____ operation in Relational Algebra returns columns specified in the list and removes duplicates.

(A) Selection

(B) Projection

(C) Union

(D) Intersection

**Q44.** Which network topology uses a central hub/switch to connect all nodes?

(A) Mesh

(B) Bus

(C) Star

(D) Ring

**Q45.** Identify the hardware address that is permanently assigned to a Network Interface Card (NIC):

(A) IP Address

(B) MAC Address

(C) Port Address

(D) URL

**Q46.** Which device is used to connect two different types of networks that use different protocols?

(A) Switch

(B) Repeater

(C) Gateway

(D) Bridge

**Q47.** Which of the following refers to the trail of data you leave behind while using the internet?

(A) Digital Footprint

(B) Cyber Trail

(C) Data Shadow

(D) Web History

**Q48.** The legal right granted to an inventor or creator to protect their invention/work for a limited period is called:

(A) Plagiarism

(B) Intellectual Property Right (IPR)

(C) Privacy Policy

(D) Open Source

**Q49.** Improper disposal of old mobile phones and computers leads to:

(A) Cybercrime

(B) E-waste

(C) Phishing

(D) Identity Theft

**Q50.** Stealing someone's personal information to commit fraud is known as:

(A) Hacking

(B) Identity Theft

(C) Spamming

(D) Trolling

## Detailed Solutions

## Q1.

### Solution

**Concept: Vectorization and Boolean Indexing in Pandas**

Pandas is built on top of NumPy, which allows for "Vectorization"—the ability to perform operations on an entire array at once rather than looping through individual elements. Boolean Indexing is a specific type of vectorization where a comparison operator (like >, <, ==) is applied to a Series to create a "mask" of True and False values. When this mask is passed into the Series using square brackets, Pandas filters the data, keeping only the rows where the mask is True. This is a fundamental technique for data cleaning and subsetting in Informatics Practices.

**Solution:**

**Step 1:** Evaluate the Boolean condition $S > 25$. The Series $S$ has values: $a : 10, b : 20, c : 30, d : 40$.

Comparing each to 25: 10 > 25 (False), 20 > 25 (False), 30 > 25 (True), 40 > 25 (True).

**Step 2:** Apply the filter to the Series. The True values correspond to indices 'c' and 'd'. The resulting subset is:

c 30

d 40

**Step 3:** Perform the arithmetic operation $(*2)$. The scalar 2 is multiplied by each element in the filtered subset:

$30 \times 2 = 60$

$40 \times 2 = 80$

**Step 4:** Final alignment of output. The Series preserves its original index labels 'c' and 'd' for the modified values.

**Final Answer:** The output is:

c 60

d 80

**Answer: (A)**

## Q2.

### Solution

**Concept: DataFrame Mutability and Accessor Logic**

A Pandas DataFrame is a mutable object, meaning its values can be changed after creation. There are two primary ways to access data: label-based and position-based. `.iloc` is strictly integer-position based (0 to length-1), while `.at` or `.loc` are label-based. When you perform an operation like adding two columns to create a new one, the new column becomes an integral part of the DataFrame's structure, and any subsequent modification to a cell via one accessor will be reflected when accessing that same cell via another accessor.

**Solution:**

**Step 1:** Column Creation. $df['C'] = df['A'] + df['B']$ is executed. Row 0: 1 + 3 = 4. Row 1: 2 + 4 = 6. The DataFrame now has a column 'C' with values [4, 6].

**Step 2:** Identify the target cell for modification. $df.iloc[0, 2] = 10$ refers to the 1st row (index 0) and the 3rd column (index 2). In our DataFrame, the 3rd column is 'C'.

**Step 3:** Apply the update. The value at (row 0, column 'C') is changed from 4 to 10.

**Step 4:** Retrieval. $df.at[0,'C']$ looks for the row labeled 0 and the column labeled 'C'. Since this is the exact cell that was updated in Step 3, it retrieves the new value.

**Final Answer:** The value returned is 10.

**Answer: (B)**

## Q3.

### Solution

**Concept: Structural Attributes of DataFrames**

Attributes in Pandas provide metadata about the object's structure. Unlike methods, attributes do not perform computations and are accessed without parentheses. The `.shape` attribute is specifically designed to provide the "dimensionality" or "layout" of the DataFrame. It is inherited from NumPy's array structure and is essential for developers to verify the size of their datasets after operations like filtering, merging, or dropping rows.

**Solution:**

**Step 1:** Understand the `.shape` definition. The shape attribute returns a Python tuple. For a 2D DataFrame, this tuple always has exactly two elements.

**Step 2:** Identify the order of elements. The first element of the tuple represents the number of rows, and the second element represents the number of columns $(Rows, Columns)$.

**Step 3:** Compare with other attributes. - `.size` returns a single integer (Rows × Cols).
- `.ndim` returns the number of dimensions (always 2 for DataFrames).

**Step 4:** Conclusion. Since the question asks for the attribute representing dimensionality in terms of (rows, columns), `.shape` is the only correct fit.

**Final Answer:** The `shape` attribute returns a tuple representing (rows, columns).

**Answer: (C)**

## Q4.

### Solution

**Concept: Permanent Data Modification via Inplace Parameter**

In Pandas, most methods like `.drop()`, `.fillna()`, or `.rename()` return a new copy of the object by default, leaving the original DataFrame untouched. To modify the data "in place" (permanently changing the original variable), the user must set the parameter `inplace=True`. Furthermore, when dropping data, the `axis` parameter must be specified: `axis=0` for rows and `axis=1` for columns.

**Solution:**

**Step 1:** Select the correct method. The `.drop()` method is used to remove labels from an axis.

**Step 2:** Define the axis for column deletion. To delete a column named 'Salary', we must specify `axis=1`. If we used `axis=0`, Pandas would look for a row index named 'Salary' and likely throw an error.

**Step 3:** Ensure the change is permanent. To satisfy the "permanently" requirement in the question, the argument `inplace=True` must be passed.

**Step 4:** Evaluate syntax. $df.drop('Salary', axis = 1, inplace = True)$ correctly combines the label, the direction, and the persistence flag.

**Final Answer:** The correct syntax is $df.drop('Salary', axis = 1, inplace = True)$.

**Answer: (B)**

## Q5.

### Solution

**Concept: Column-wise Non-Null Counting**

The `.count()` method is an aggregation function used to find the number of non-missing (non-NaN) values in a dataset. In a DataFrame, aggregation can happen along two axes. By default, Pandas aggregates along the index (`axis=0`), meaning it produces a result for every column. It is important to distinguish this from `len(df)`, which only gives the number of rows, or `df.size`, which gives the total cell count.

**Solution:**

**Step 1:** Determine the default axis. For `df.count()`, the default is `axis=0`. This counts values row-by-row for each column.

**Step 2:** Identify the number of results. If a DataFrame has 3 columns, the function will calculate 3 separate totals.

**Step 3:** Determine the return type. Because there are multiple labeled results (one for each column name), Pandas packages these into a **Series** object.

**Step 4:** Match with the scenario. A DataFrame with 5 rows and 3 columns will result in a Series where the index contains the 3 column names, and the values are the counts (which would be 5 if there are no nulls).

**Final Answer:** The result is a Series with 3 entries.

**Answer: (C)**

## Q6.

### Solution

**Concept: Handling Missing Data with the Count Method**

In Data Handling with Pandas, dealing with missing values ($NaN$ - Not a Number) is a critical skill. The `count()` method is specifically designed to evaluate the presence of valid data. It is important to distinguish between `count()` and `size`. While `size` is an attribute that returns the total number of elements including $NaN$, the `count()` method only tallies elements that are not null. This is essential for calculating statistics like averages where missing data must be excluded from the denominator.

**Solution:**

**Step 1:** Analyze the Series data. The Series $obj$ contains four elements: $[10, NaN, 30, 40]$.

**Step 2:** Identify the null values. The second element is $NaN$, which represents a missing value in Pandas (typically handled via the NumPy `np.nan` constant).

**Step 3:** Apply the `count()` logic. The method iterates through the Series and increments the counter only for valid, non-null entries: - 10: Valid (Count = 1) - $NaN$: Null (Count stays 1) - 30: Valid (Count = 2) - 40: Valid (Count = 3)

**Step 4:** Final count verification. Total elements are 4, but valid elements are 3.

**Final Answer:** The result of $obj.count()$ is 3.

**Answer: (B)**

## Q7.

### Solution

**Concept: Indexing and Slicing with .loc and .iloc**

Pandas provides two primary indexers for data selection: `.loc` and `.iloc`. The core difference lies in how they interpret the selection criteria. `.loc` is **label-based**, meaning you specify the names of the rows or columns. `.iloc` is **integer-position based**, meaning you specify the numerical index (starting from 0). A critical distinction frequently tested in the CUET-UG exam is how these two handle "slicing" ranges (e.g., $1:3$).

**Solution:**

**Step 1:** Define `.loc` slicing behavior. When using labels in a slice like `df.loc['A':'C']`, the endpoint ('C') is **included** in the results.

**Step 2:** Define `.iloc` slicing behavior. When using integer positions in a slice like `df.iloc[0:2]`, the endpoint is **excluded**, following the standard Python list slicing convention (stops at index $n-1$).

**Step 3:** Compare indexing types. Statement (A) in the question is a common distractor; `loc` does not use integer positions by definition (unless the labels themselves are integers).

**Step 4:** Synthesize the correct distinction. The most reliable difference between the two is that `loc` is inclusive of the stop boundary, whereas `iloc` is exclusive.

**Final Answer:** The correct statement is that `loc` includes the last element in a slice, while `iloc` excludes it.

**Answer: (C)**

## Q8.

### Solution

**Concept: Efficient Data Inspection with Head and Tail**

When working with large datasets (e.g., 100+ rows), displaying the entire DataFrame is inefficient and consumes excessive memory. Pandas provides the `.head(n)` and `.tail(n)` methods for quick inspection. `.head()` retrieves the first $n$ rows, while `.tail()` retrieves the last $n$ rows. If $n$ is not specified, it defaults to 5.

**Solution:**

**Step 1:** Identify the requirement. The goal is to display exactly the "last 3 rows."

**Step 2:** Select the appropriate method. The `.tail()` method is specifically designed for retrieving entries from the end of the DataFrame.

**Step 3:** Pass the correct argument. To get exactly 3 rows, we pass the integer 3 as the parameter: `df.tail(3)`.

**Step 4:** Evaluate alternatives. `df.head(-3)` would actually display all rows *except* the last 3, which is the opposite of the requirement. `df.iloc[0:3]` would display the first 3 rows (positions 0, 1, 2).

**Final Answer:** The most efficient command is $df.tail(3)$.

**Answer: (B)**

## Q9.

### Solution

**Concept: Label-based Indexing on Integer Labels**

One of the most confusing scenarios in Pandas occurs when the index labels of a Series are integers, but they are not in the default order (0, 1, 2...). In such cases, `.loc` still strictly looks for the **label** matching that number, not the position. This distinguishes the explicit label from the implicit positional index.

**Solution:**

**Step 1:** Analyze the Series construction. $s = pd.Series([1,2,3], index = [2,1,0])$. This means: - Label 2 maps to Value 1 - Label 1 maps to Value 2 - Label 0 maps to Value 3

**Step 2:** Interpret the command. The command is $s.loc[1]$. Because `.loc` is label-based, it searches for the value associated with the label '1'.

**Step 3:** Map label to value. From Step 1, we see that label 1 corresponds to the second value in the array, which is 2.

**Step 4:** Conclusion. If the command had been `s.iloc[1]`, it would also have returned 2 (the value at the second position). However, if the command was `s.loc[0]`, it would return 3.

**Final Answer:** The output is 2.

**Answer: (B)**

## Q10.

### Solution

**Concept: Data Export and Delimiter Customization**

Exporting data is a common task in Informatics Practices. The `to_csv()` method is the primary tool for saving a DataFrame to a text file. While "CSV" stands for Comma Separated Values, the method is flexible enough to use any character (like a semicolon, tab, or pipe) as a field separator through the `sep` parameter.

**Solution:**

**Step 1:** Identify the correct export method. The standard method is `df.to_csv()`.

**Step 2:** Understand the default behavior. By default, `to_csv()` uses a comma (,) as the delimiter.

**Step 3:** Use the separator parameter. To change the delimiter to a semicolon (;), we use the `sep` (short for separator) argument. The syntax is `sep=';'`.

**Step 4:** Rule out incorrect options. - `delimiter` is a valid alias in some functions, but `sep` is the standard in Pandas. - `save_csv` and `export` are not valid Pandas method names. - `tab` is not a valid parameter name for defining the separator.

**Final Answer:** The correct method is $df.to\_csv('file.csv', sep=';')$.

**Answer: (A)**

## Q11.

### Solution

**Concept: Creating DataFrames from Nested Dictionaries**

Pandas allows the creation of DataFrames from various Python structures, with dictionaries being the most common. When a "Dictionary of Dictionaries" (a nested dictionary) is used, Pandas interprets the data structurally to form a 2D grid. In this mapping, the keys of the outer dictionary and the keys of the inner dictionary serve specific roles as the axes of the DataFrame. Understanding this hierarchy is essential for correctly predicting the layout of data imported from JSON or nested Python objects.

**Solution:**

**Step 1:** Analyze the structure of a 2D dictionary. Consider an example: `data = {'Col1': {'r1': 1, 'r2': 2}, 'Col2': {'r1': 3, 'r2': 4}}`.

**Step 2:** Identify the outer keys. The outer keys are 'Col1' and 'Col2'.

**Step 3:** Map to DataFrame axes. In Pandas, the keys of the outer dictionary are automatically treated as the **Column Names**.

**Step 4:** Identify the inner keys. The inner keys ('r1', 'r2') are treated as the **Row Indices** (labels).

**Step 5:** Final confirmation. This behavior ensures that each inner dictionary represents a column of data. Therefore, the outer keys define the horizontal labels of the DataFrame.

**Final Answer:** The outer keys of the dictionary become the Column Names of the DataFrame.

**Answer: (B)**

## Q12.

### Solution

**Concept: The Rename Method and Object Persistence**

The `rename()` method in Pandas is used to alter axes labels (index or columns). Like most transformation methods in the library, it follows the principle of "immutability" by default. It does not modify the original DataFrame; instead, it generates a new DataFrame object with the changes applied. To make the change persist in the original object, one must use the `inplace=True` parameter or reassign the result to the original variable name.

**Solution:**

**Step 1:** Analyze the function call. The code provided is `df.rename(columns={'X':'Alpha'})`.

**Step 2:** Check for the persistence flag. The parameter `inplace=True` is missing from the function call.

**Step 3:** Determine the return value. Since `inplace` is False (default), the method creates a "copy" of the original DataFrame, applies the rename operation to that copy, and returns it.

**Step 4:** Evaluate the state of the original 'df'. The original DataFrame `df` remains unchanged with its original column name 'X'. The command effectively produces a "New DataFrame" result.

**Final Answer:** It returns a new DataFrame with the column 'X' changed to 'Alpha'.

**Answer: (B)**

## Q13.

### Solution

**Concept: The Empty Attribute in Pandas**

In data processing workflows, it is often necessary to check if a DataFrame contains any data before performing operations like aggregation or plotting. Pandas provides a built-in property called `.empty` for this purpose. This is an **attribute**, not a method, meaning it does not require parentheses. It returns a Boolean value (`True` if the DataFrame has no data, `False` otherwise).

**Solution:**

**Step 1:** Distinguish between attributes and methods. `df.isempty()` is not a valid Pandas method. `len(df) == 0` is a valid Python check but specifically checks for the number of rows, not necessarily if the structure is "empty" in all contexts.

**Step 2:** Evaluate the `.empty` property. The `.empty` attribute is the standard Pandas way to verify if an object is devoid of elements. A DataFrame is considered empty if it has no rows OR no columns.

**Step 3:** Compare with `isnull()`. The `.isnull()` method checks for missing values ($NaN$) within an existing structure; it does not tell you if the structure itself is empty.

**Step 4:** Conclusion. The attribute `.empty` is the most concise and direct way to perform this check.

**Final Answer:** The `empty` attribute is used to check if a DataFrame is empty.

**Answer: (B)**

## Q14.

### Solution

**Concept: Integer-Based Scalar Access with .iloc**

Accessing and modifying specific cells in a DataFrame is a core task. For integer-based indexing, Pandas offers `.iloc` (for slices/groups) and `.iat` (optimized for single values). Since the question specifies using "integer-based indexing" and provides coordinates for a single cell, we must correctly map the 0-based index of the rows and columns.

**Solution:**

**Step 1:** Understand 0-based indexing. In programming, the first item is at index 0. Therefore: - 1st row = Index 0 - 2nd row = Index 1 - 1st column = Index 0 - 2nd column = Index 1

**Step 2:** Map the question's coordinates. The question asks for the "2nd row, 1st column". Mapping to indices: 2nd row $\rightarrow$ 1, 1st column $\rightarrow$ 0.

**Step 3:** Formulate the command. Using `.iloc`, the syntax is `df.iloc[row_index, column_index]`. Plugging in our values: `df.iloc[1, 0] = 50`.

**Step 4:** Eliminate incorrect options. Option A (2, 1) refers to the 3rd row and 2nd column. Option C (`iat`) uses the wrong coordinates. Option D (`loc`) is label-based, not integer-based.

**Final Answer:** The correct command is $df.iloc[1, 0] = 50$.

**Answer: (B)**

## Q15.

### Solution

**Concept: Transposing Data Structures**

The `T` attribute (short for Transpose) is a convenient way to flip a DataFrame over its main diagonal. This operation switches the row index with the column index. Transposing is frequently used in Informatics Practices when data is received in a format where features are in rows and observations are in columns, but the analysis requires the standard format (features as columns).

**Solution:**

**Step 1:** Define the Transpose operation. Transposing converts the rows of the original DataFrame into the columns of the new DataFrame, and vice-versa.

**Step 2:** Identify the attribute. In Pandas, this is achieved using the `.T` attribute. It is important to note that `.T` is an accessor, not a function (no parentheses).

**Step 3:** Understand the effect on metadata. If the original DataFrame has 5 rows and 3 columns, the transposed version (`df.T`) will have 3 rows and 5 columns. The index of the new DataFrame will be the column labels of the original.

**Step 4:** Final Answer selection. Among the choices, transposing is the only function associated with the `T` attribute.

**Final Answer:** The $T$ attribute transposes rows and columns.

**Answer: (B)**

## Q16.

### Solution

**Concept: Skipping Rows during CSV Import**

When importing data from external sources like CSV files, the file often contains metadata, descriptions, or unnecessary headers at the beginning that are not part of the actual dataset. The `read_csv()` function in Pandas provides the `skiprows` parameter to handle this. This parameter allows the user to define exactly how many lines at the start of the file should be ignored before the parser begins reading the data into a DataFrame.

**Solution:**

**Step 1:** Identify the requirement. We need to ignore the first $n$ rows of a CSV file while loading it.

**Step 2:** Analyze the `read_csv()` parameters. The `skiprows` parameter can take an integer (e.g., `skiprows=5` skips the first 5 lines) or a list of row indices (e.g., `skiprows=[0, 2]` skips the 1st and 3rd lines).

**Step 3:** Distinguish from `header`. The `header` parameter tells Pandas which row to use as column names; it does not "skip" the rows above it in the same structural way that `skiprows` does.

**Step 4:** Conclusion. The standard and most direct parameter for this task is `skiprows`.

**Final Answer:** The parameter `skiprows = n` is used to skip the first $n$ rows.

**Answer: (A)**

## Q17.

### Solution

**Concept: Scalar Value Series Creation**

A Pandas Series can be created from a single "scalar" value (like an integer or string). When a scalar is provided along with an index, Pandas performs "Broadcasting." This means it repeats the scalar value for every index label specified. This is a very efficient way to initialize a Series where every entry starts with the same default value.

**Solution:**

**Step 1:** Analyze the constructor. $S = pd.Series(5, index = [0, 1, 2, 3])$. Here, 5 is the data, and the index has four elements: 0, 1, 2, and 3.

**Step 2:** Apply Broadcasting. Since only one data value (5) is provided for four index labels, Pandas fills every position with 5. The Series effectively becomes: 0: 5, 1: 5, 2: 5, 3: 5.

**Step 3:** Retrieve the value. The command $S[2]$ asks for the value associated with index label 2.

**Step 4:** Final Answer. As determined in Step 2, every index in this Series points to the value 5.

**Final Answer:** The value of $S[2]$ is 5.

**Answer: (B)**

## Q18.

### Solution

**Concept: Filtering Grouped Data in SQL**

In SQL, there is a strict logical order of operations. The `WHERE` clause is used to filter individual rows *before* any grouping or aggregation takes place. However, once data is grouped using the `GROUP BY` clause, we often need to filter the results based on an aggregate value (like a sum, average, or count). Because the `WHERE` clause cannot see the results of aggregate functions, SQL provides the `HAVING` clause specifically for this purpose.

**Solution:**

**Step 1:** Understand the limitation of `WHERE`. You cannot write `WHERE SUM(Salary) > 1000` because `WHERE` filters the raw data before `SUM` is even calculated.

**Step 2:** Identify the role of `HAVING`. The `HAVING` clause acts as a filter on the "groups" created by `GROUP BY`. It is processed after the aggregation has occurred.

**Step 3:** Sequence of execution. 1. `FROM` (Load table) → 2. `WHERE` (Filter rows) → 3. `GROUP BY` (Group rows) → 4. `HAVING` (Filter groups).

**Step 4:** Final selection. Since the question asks to filter records *after* an aggregate function is applied, `HAVING` is the correct clause.

**Final Answer:** The `HAVING` clause is used to filter records after aggregation.

**Answer: (C)**

## Q19.

### Solution

**Concept: Aggregate Filtering Syntax in SQL**

This question tests the practical application of the `GROUP BY` and `HAVING` clauses. When generating a report that summarizes data (e.g., "Total per region"), we use the `SUM()` function along with a grouping column. If that report needs to exclude certain groups based on their calculated total, the logic must follow the standard SQL syntax hierarchy.

**Solution:**

**Step 1:** Select the columns. We need the Region and the total amount: `SELECT Region, SUM(Amount)`.

**Step 2:** Define the source and grouping. Data comes from 'Sales', and we want results per region: `FROM Sales GROUP BY Region`.

**Step 3:** Apply the aggregate condition. We need regions where the total is > 5000. Since this is a condition on an aggregate (`SUM`), we must use `HAVING`. The condition is `HAVING SUM(Amount) > 5000`.

**Step 4:** Combine the clauses. `SELECT Region, SUM(Amount) FROM Sales GROUP BY Region HAVING SUM(Amount) > 5000;`.

**Step 5:** Eliminate errors. Option A is wrong because it uses `SUM` in a `WHERE` clause. Option C is wrong because it puts `WHERE` after `GROUP BY`.

**Final Answer:** The correct query uses `GROUP BY` followed by `HAVING SUM(Amount) > 5000`.

**Answer: (B)**

## Q20.

### Solution

**Concept: Cartesian Product (Cross Join) Logic**

In Database Management Systems, a Cartesian Product (or Cross Join) occurs when every row of the first table is paired with every row of the second table. This usually happens in SQL when two tables are listed in the `FROM` clause without a joining condition (`WHERE` or `ON` clause). It represents the maximum possible size of a join and is used as a mathematical foundation for relational algebra.

**Solution:**

**Step 1:** Understand the multiplication rule. If Table A has $n$ rows and Table B has $m$ rows, the number of rows in their Cartesian Product is always $n \times m$.

**Step 2:** Apply the numbers from the question. Table A has 10 rows. Table B has 5 rows.

**Step 3:** Perform the calculation. $10 \times 5 = 50$.

**Step 4:** Visualize the result. For the first row of Table A, there are 5 matching rows from Table B. This repeats 10 times. Total rows = 5 + 5 + 5...(10 times) = 50.

**Final Answer:** The result will contain 50 records.

**Answer: (C)**

## Q21.

### Solution

**Concept: Maximum Rows in an Equi-Join**

An Equi-Join is a type of join that combines rows from two tables based on equivalent values in specified columns. Unlike a Cartesian Product (Cross Join), which always results in $n \times m$ rows, an Equi-Join filters these pairs. However, if the common column in both tables contains identical values in every row (for example, if every row in Table A and every row in Table B has the value '1' in the join column), the Equi-Join will behave exactly like a Cartesian Product.

**Solution:**

**Step 1:** Recall the Cartesian Product rule. The total possible combinations of rows from Table A ($n$) and Table B ($m$) is $n \times m$.

**Step 2:** Analyze the Equi-Join constraint. An Equi-Join is a subset of the Cartesian Product. It only keeps rows where the join condition (e.g., $A.id = B.id$) is met.

**Step 3:** Determine the "at most" (maximum) scenario. The maximum number of rows occurs when the join condition is true for every single possible combination. This happens if all join-key values are the same in both tables.

**Step 4:** Conclusion. Since the Equi-Join cannot exceed the total number of mathematical combinations of the two sets, the upper limit is the product of the number of rows in both tables.

**Final Answer:** The result will have at most $n \times m$ rows.

**Answer: (B)**

## Q22.

### Solution

**Concept: Multi-Column Sorting with ORDER BY**

The `ORDER BY` clause in SQL is used to sort the result set. It allows for multi-level sorting, where the data is first sorted by the first specified column. If there are "ties" (identical values) in the first column, SQL then uses the second specified column to sort those specific tied records. This process continues for as many columns as are listed. Each column can have its own sort direction: `ASC` (ascending, default) or `DESC` (descending).

**Solution:**

**Step 1:** Identify the primary sort requirement. The question asks to sort by 'Salary' in descending order. The keyword is `Salary DESC`.

**Step 2:** Identify the secondary sort requirement. For employees with the same salary, they should be sorted by 'Name' in ascending order. The keyword is `Name ASC` (or just `Name`, as `ASC` is the default).

**Step 3:** Structure the SQL clause. The syntax is `ORDER BY column1 [direction], column2 [direction]`. Applying our columns: `ORDER BY Salary DESC, Name ASC`.

**Step 4:** Rule out incorrect options. - Option B swaps the directions. - Option C uses "SORT BY" which is not standard SQL (it's `ORDER BY`). - Option D puts the keywords before the column names, which is syntactically invalid.

**Final Answer:** The correct command is $ORDER\ BY\ Salary\ DESC, Name\ ASC$.

**Answer: (A)**

## Q23.

### Solution

**Concept: Logical Query Processing Order**

SQL queries are not processed in the order they are written. They follow a specific logical sequence: `FROM` → `WHERE` → `GROUP BY` → `HAVING` → `SELECT` → `ORDER BY`. Understanding this sequence is vital because it determines which aliases can be used and where specific filters must be placed. The `GROUP BY` clause must work on the data filtered by `WHERE`, and the `HAVING` clause must work on the groups created by `GROUP BY`.

**Solution:**

**Step 1:** Locate `GROUP BY` in the sequence. As shown in the sequence above, `GROUP BY` comes after the `WHERE` clause.

**Step 2:** Analyze the relationship with `HAVING`. `HAVING` is used to filter groups. Therefore, the groups must exist before they can be filtered. This means `GROUP BY` must come before `HAVING`.

**Step 3:** Synthesize the positions. `GROUP BY` is placed **after** the `WHERE` clause and **before** the `HAVING` clause.

**Step 4:** Match with the MCQ options. Option B states: "After [WHERE], Before [HAVING]". This perfectly matches the logical flow.

**Final Answer:** The GROUP BY clause is placed after the WHERE clause and before the HAVING clause.

**Answer: (B)**

## Q24.

### Solution

**Concept: Aggregate Functions and NULL Values**

In SQL, `NULL` represents a missing or unknown value. Aggregate functions (`SUM`, `AVG`, `MIN`, `MAX`, `COUNT`) handle `NULL`s in a specific way: they generally ignore them. For example, `SUM()` only adds up non-null numbers. However, there is a major exception in the `COUNT()` function family that students must memorize for the CUET-UG exam.

**Solution:**

**Step 1:** Evaluate `SUM()`. If a column has values $[10, NULL, 20]$, `SUM()` returns 30. It effectively ignores the `NULL`.

**Step 2:** Evaluate `COUNT(column_name)`. This also ignores `NULL`s and only counts rows where the column has a value.

**Step 3:** Evaluate `COUNT(*)`. This is the exception. `COUNT(*)` counts **rows**, not column values. Therefore, it counts every row in the result set, regardless of whether some columns contain `NULL`.

**Step 4:** Select the functions that ignore NULLs. Both `SUM()` and `COUNT(column_name)` ignore `NULL`s. Since `SUM()` is specifically listed and the prompt implies general aggregate behavior, we look for the best fit. Note: `COUNT(*)` is the only one that *includes* them.

**Step 5:** Conclusion. While most aggregates ignore NULL, the question asks which one (singular or plural) specifically does so. Since `COUNT(*)` includes them, and `SUM()` ignores them, Option B is a correct statement, but Option C (Both) is invalid because `COUNT(*)` does NOT ignore them.

**Final Answer:** $SUM()$ is an aggregate function that ignores NULL values.

**Answer: (B)**

## Q25.

### Solution

**Concept: Cartesian Product resulting from Missing Joins**

A "Join" in SQL is the process of combining columns from one or more tables based on a logical relationship. If a developer lists two tables in a query (e.g., `SELECT * FROM Table1, Table2`) but fails to provide a joining condition (like `WHERE Table1.id = Table2.id` or `JOIN...ON`), the SQL engine has no instructions on how to match the rows.

**Solution:**

**Step 1:** Define the default behavior. When no logic is provided to link Table A to Table B, the database performs a "Cross Join."

**Step 2:** Identify the mathematical term. The mathematical name for a Cross Join is the **Cartesian Product**.

**Step 3:** Describe the outcome. The result set will contain every possible combination of rows from the two tables. This often leads to a massive, mostly useless result set if the tables are large.

**Step 4:** Compare with other joins. - Equi-Join requires an '=' condition. - Natural Join automatically joins on columns with the same name. - Outer Join requires specific keywords to include unmatched rows. Only the Cartesian Product happens by default when the condition is missing.

**Final Answer:** The result is a Cartesian Product.

**Answer: (C)**

## Q26.

### Solution

**Concept: Counting Unique Values with DISTINCT**

In SQL, the `COUNT()` function can be modified by the `DISTINCT` keyword. By default, `COUNT(column_name)` returns the total number of non-null entries in that column, including duplicates (e.g., if three employees are in 'Sales', 'Sales' is counted three times). To find the count of *different* categories or departments, we must instruct the engine to remove duplicates before counting. This is achieved by placing `DISTINCT` inside the parentheses of the count function.

**Solution:**

**Step 1:** Analyze the requirement. We need the number of "different" departments. This implies we need a unique count.

**Step 2:** Evaluate `COUNT(Dept)`. This would return the total number of department entries (e.g., if there are 50 employees, it returns 50), which is incorrect.

**Step 3:** Apply `DISTINCT`. By using `COUNT(DISTINCT Dept)`, SQL first identifies the unique department names (e.g., 'HR', 'IT', 'Sales') and then counts that specific list.

**Step 4:** Check syntax. The `DISTINCT` keyword must be inside the `COUNT()` function. Writing `DISTINCT COUNT()` would try to find distinct values of the final total, which is always just one number.

**Final Answer:** The correct query is $SELECT\ COUNT(DISTINCT\ Dept)\ FROM\ Employee;$.

**Answer: (B)**

## Q27.

### Solution

**Concept: Numerical Rounding in SQL**

The `ROUND(number, decimals)` function is used to round a numeric value to a specified number of decimal places. The logic follows standard mathematical rounding: if the digit at the $(n+1)^{th}$ position is 5 or greater, the $n^{th}$ digit is incremented by 1. If the second argument is positive, it rounds to the right of the decimal point; if zero, to the nearest integer; and if negative, to the left of the decimal point (tens, hundreds, etc.).

**Solution:**

**Step 1:** Identify the inputs. The number is 15.768 and the decimal precision is 2.

**Step 2:** Locate the rounding boundary. We need to keep two decimal places: .76. The next digit (the 3rd decimal) is 8.

**Step 3:** Apply rounding rules. Since $8 \geq 5$, we must round up the last kept digit. The 6 becomes 7.

**Step 4:** Final result. The number 15.768 rounded to two places is 15.77.

**Final Answer:** The output of $SELECT\ ROUND(15.768, 2)$; is 15.77.

**Answer: (B)**

## Q28.

### Solution

**Concept: Substring Searching Functions**

MySQL provides several functions to find the position of a substring within a larger string. This is useful for data parsing and validation. The position returned is 1-indexed (the first character is position 1, not 0). The most common functions for this purpose are `INSTR()`, `LOCATE()`, and `POSITION()`. While they have slight variations in syntax/argument order, they all serve the same fundamental purpose in Informatics Practices.

**Solution:**

**Step 1:** Evaluate `LOCATE(substr, str)`. This returns the position of the first occurrence of `substr` in `str`.

**Step 2:** Evaluate `INSTR(str, substr)`. This is similar to `LOCATE` but swaps the order of arguments. It also returns the position of the first occurrence.

**Step 3:** Evaluate `POSITION(substr IN str)`. This is the standard SQL syntax for the same operation.

**Step 4:** Conclusion. Since all three functions are valid MySQL commands to find a substring's location, "All of the above" is the correct choice.

**Final Answer:** All of the above functions can be used.

**Answer: (D)**

## Q29.

### Solution

**Concept: The Modulo Operator (MOD)**

The `MOD(n, m)` function (or the `%` operator) returns the remainder of a division operation. It is a mathematical function frequently used to determine if a number is even or odd (using `MOD(x, 2)`) or to perform cyclical logic. In SQL, `MOD(n, m)` is calculated as: $n - (m \times \text{floor}(n/m))$.

**Solution:**

**Step 1:** Set up the division. We are dividing 11 by 3.

**Step 2:** Find the quotient (integer part). $11 \div 3 = 3$ with some remainder ($3 \times 3 = 9$).

**Step 3:** Calculate the remainder. $11 - 9 = 2$.

**Step 4:** Result verification. Since 2 is less than the divisor (3), it is the final remainder. Therefore, $MOD(11, 3)$ is 2.

**Final Answer:** The result is 2.

**Answer: (B)**

## Q30.

### Solution

**Concept: Date Extraction Functions**

Working with temporal data requires the ability to extract specific components from a `DATE` or `DATETIME` value. MySQL provides specific functions to retrieve the day of the week in different formats. `DAYNAME()` returns the name of the day (e.g., 'Monday'), while `WEEKDAY()` or `DAYOFWEEK()` return numeric representations (e.g., 0-6 or 1-7). These are essential for grouping sales by day or generating weekly reports.

**Solution:**

**Step 1:** Evaluate `DAYNAME()`. This extracts the literal name of the day from a date string or object.

**Step 2:** Evaluate `WEEKDAY()`. This extracts the index of the day (where Monday=0, Tuesday=1, etc.).

**Step 3:** Compare with requirements. Both functions are used to "extract the day of the week," even if the output format (string vs. number) differs.

**Step 4:** Evaluate `DATEWEEK()`. This is not a standard MySQL function.

**Step 5:** Conclusion. Since both A and B are valid extraction methods for the day of the week, Option C is the correct answer.

**Final Answer:** Both (A) and (B) are used to extract day-of-week information.

**Answer: (C)**

## Q31.

### Solution

**Concept: String Extraction using SUBSTR / MID**

In SQL, extracting a portion of a string is handled by functions like `SUBSTR()`, `SUBSTRING()`, or `MID()`. These functions are synonymous in MySQL. They require three arguments: the source string, the starting position, and the number of characters to extract. A vital rule to remember for competitive exams is that SQL string indexing starts at 1, not 0. If the starting position is positive, it counts from the beginning; if negative, it counts from the end.

**Solution:**

**Step 1:** Analyze the function call. $SELECT\ SUBSTR('Informatics', 3, 4)$; Source string = 'Informatics'.

**Step 2:** Identify the starting position. The second argument is 3. Position 1: 'I', Position 2: 'n', Position 3: 'f'. Extraction begins at 'f'.

**Step 3:** Determine the length. The third argument is 4. We must extract 4 characters starting from 'f'. - 1st char: 'f' - 2nd char: 'o' - 3rd char: 'r' - 4th char: 'm'

**Step 4:** Combine the results. The resulting substring is 'form'.

**Final Answer:** The output is 'form'.

**Answer: (A)**

## Q32.

### Solution

**Concept: Length Calculation and Whitespace**

The `LENGTH()` function in MySQL returns the length of a string measured in bytes. For standard alphanumeric characters, this corresponds directly to the number of characters in the string. A common pitfall is ignoring trailing or leading spaces. In SQL, a space (' ') is a valid character and occupies a position in the string, thus contributing to the total length.

**Solution:**

**Step 1:** Analyze the input string. The string is `'Database '`.

**Step 2:** Count the visible characters. 'D', 'a', 't', 'a', 'b', 'a', 's', 'e' counts to 8 characters.

**Step 3:** Identify trailing whitespace. The string ends with a single space character after the 'e'.

**Step 4:** Calculate the final sum. 8 (letters) + 1 (space) = 9.

**Step 5:** Conclusion. The `LENGTH()` function does not automatically trim strings. Therefore, the space is included.

**Final Answer:** The function returns 9.

**Answer: (B)**

## Q33.

### Solution

**Concept: Current Date and Time Functions**

MySQL offers various functions to retrieve the system's current temporal data. Understanding the difference between a function that returns only the date and one that returns both date and time is crucial for database logging and timestamping. - `CURDATE()`: Returns the current date only ('YYYY-MM-DD'). - `NOW()`: Returns the current date and time ('YYYY-MM-DD HH:MM:SS'). - `SYSDATE()`: Similar to `NOW()` but returns the time at which the function executes.

**Solution:**

**Step 1:** Evaluate `DATE()`. This is usually used to extract the date part from a datetime expression, not to fetch the current system time.

**Step 2:** Evaluate `CURDATE()`. This returns only the date, excluding the time.

**Step 3:** Evaluate `NOW()`. This is the standard function used in MySQL to get both the current system date and the current system time.

**Step 4:** Evaluate `SYSDATE`. While valid, in the context of standard MCQ options for CUET, `NOW()` is the primary expected answer for combined "date and time." Note that `SYSDATE` usually requires parentheses `()` to be treated as a function in most contexts.

**Final Answer:** The function `NOW()` returns the current system date and time.

**Answer: (C)**

## Q34.

### Solution

**Concept: Nested String Functions**

In Informatics Practices, you will often encounter "nested" functions where the output of one function serves as the input for another. To solve these, you must apply the "Inside-Out" rule: solve the innermost function first, then work your way to the outer layers. This specific question combines `MID()` (an alias for `SUBSTR`) with `LCASE()` (an alias for `LOWER`).

**Solution:**

**Step 1:** Solve the innermost function: `MID('PYTHON', 2, 3)`. - Start at position 2: 'Y'. - Extract 3 characters: 'Y', 'T', 'H'. - Inner result = 'YTH'.

**Step 2:** Apply the outer function: `LCASE('YTH')`. - `LCASE` converts all characters to lowercase. - 'Y' → 'y', 'T' → 't', 'H' → 'h'.

**Step 3:** Final Result. The processed string is 'yth'.

**Final Answer:** The result is 'yth'.

**Answer: (A)**

## Q35.

### Solution

**Concept: Labeling Axes in Matplotlib**

Matplotlib is the standard library for data visualization in Python. To make a graph informative, the axes must be labeled. This is done using specific functions within the `pyplot` module (usually imported as `plt`). These functions allow for customization of font size, color, and label text.

**Solution:**

**Step 1:** Identify the library module. We are using `matplotlib.pyplot`.

**Step 2:** Locate the x-axis label function. The library uses intuitive naming: `xlabel()` for the horizontal axis and `ylabel()` for the vertical axis.

**Step 3:** Rule out incorrect options. - `xname()` and `labelx()` are not valid Matplotlib functions. - `set_x()` is used in the object-oriented API (on an Axes object), but the standard `pyplot` function is `xlabel()`.

**Step 4:** Syntax check. The usage is $plt.xlabel(\text{"}TextLabel\text{"})$.

**Final Answer:** The function used is `plt.xlabel()`.

**Answer: (B)**

## Q36.

### Solution

**Concept: Horizontal Bar Charts in Matplotlib**

A bar chart is used to compare different categories of data. While the standard vertical bar chart is most common, horizontal bar charts are often preferred when the category labels are long or when there are many categories to display. In Matplotlib, the `pyplot` module provides distinct functions for these two variations. The vertical chart uses `bar()`, whereas the horizontal version uses a specific variation of that function.

**Solution:**

**Step 1:** Recall the vertical bar function. The function $plt.bar(x, height)$ creates vertical bars.

**Step 2:** Identify the horizontal variant. Matplotlib uses the suffix 'h' to denote horizontal. Thus, the function is $plt.barh(y, width)$.

**Step 3:** Analyze the arguments. In a horizontal bar chart, the categorical labels are usually on the y-axis, and the numerical values (widths) are on the x-axis.

**Step 4:** Rule out incorrect options. - `hbar()` is not a standard function name in Matplotlib. - `plot(kind='barh')` is the syntax used in the Pandas plotting API, but the question specifically asks for the Matplotlib function.

**Final Answer:** The function used to display a horizontal bar chart is $plt.barh()$.

**Answer: (C)**

## Q37.

### Solution

**Concept: Pie Chart Customization and the Explode Property**

A pie chart represents data as slices of a circle, where each slice's size is proportional to the quantity it represents. To emphasize a particular category, Matplotlib allows the user to "explode" or pull a slice away from the center of the pie. This is a common visualization technique used to draw the viewer's attention to a specific data point, such as the highest-selling product or a critical demographic.

**Solution:**

**Step 1:** Identify the `plt.pie()` function parameters. The main parameters include `labels`, `colors`, `autopct`, and `explode`.

**Step 2:** Understand the `explode` parameter. The `explode` argument takes a list or array of values. Each value represents the fraction of the radius by which that specific slice should be offset from the center.

**Step 3:** Logical application. A value of 0 keeps the slice in place. A value like 0.1 pulls the slice out slightly.

**Step 4:** Evaluate the options. Terms like `separate`, `shadow`, and `pop` are either incorrect or serve different purposes (e.g., `shadow=True` adds a 3D effect).

**Final Answer:** The `explode` parameter is used to pop out a slice.

**Answer: (B)**

## Q38.

### Solution

**Concept: Data Identification using Legends**

When a single plot contains multiple data series (for example, two different line graphs or grouped bar charts), it becomes difficult for the viewer to distinguish which color or marker represents which dataset. The `legend()` function solves this by creating a small descriptive box within the plot area that maps the visual styles (colors/labels) to the data names.

**Solution:**

**Step 1:** Define the purpose of a legend. The legend acts as a key for the chart. Without it, a multi-series graph is often unreadable.

**Step 2:** Analyze how legends are created. First, the user must provide a `label` argument inside the plotting function (e.g., $plt.plot(x, y, label = "Sales")$).

**Step 3:** Invoke the legend. The command $plt.legend()$ then automatically searches for these labels and displays them in a designated area (top right, bottom left, etc.).

**Step 4:** Rule out distractors. The legend does not set the main title (that's `plt.title()`) and does not save the file (that's `plt.savefig()`).

**Final Answer:** The `plt.legend()` function is used to identify different data series in the plot.

**Answer: (B)**

## Q39.

### Solution

**Concept: Histograms vs. Other Charts**

Choosing the right chart type is a major part of the Informatics Practices syllabus. While bar charts compare discrete categories (e.g., Apple vs. Orange), histograms are used to show the distribution of a continuous numerical variable. They group data into ”bins” (intervals) and show the frequency of data points falling within each bin. This makes them the ideal tool for visualizing the spread, skewness, and central tendency of a dataset.

**Solution:**

**Step 1:** Define Frequency Distribution. This refers to how often specific ranges of values occur in a dataset (e.g., ”How many students scored between 80-90?”).

**Step 2:** Evaluate the options. - Line charts show trends over time. - Pie charts show parts of a whole. - Scatter plots show the relationship between two variables.

**Step 3:** Identify the Histogram’s role. The Histogram is the only chart specifically designed to divide numerical data into intervals and count frequencies.

**Step 4:** Conclusion. For showing frequency distributions, the Histogram is the standard statistical choice.

**Final Answer:** The Histogram is best suited for showing frequency distributions.

**Answer: (C)**

## Q40.

### Solution

**Concept: Database Keys and Alternate Keys**

In a relational database, keys are used to identify rows and establish relationships. - **Candidate Key:** Any column or set of columns that can uniquely identify a row. - **Primary Key:** The specific Candidate Key chosen by the database designer to be the main identifier. - **Alternate Key:** Any Candidate Key that was **not** selected as the Primary Key. Understanding the hierarchy from Candidate Keys to Primary/Alternate keys is a high-yield topic for CUET.

**Solution:**

**Step 1:** Understand the relationship between keys. All keys that have the potential to be a Primary Key are called Candidate Keys.

**Step 2:** Identify the Primary Key. From the pool of Candidate Keys, exactly one is chosen as the Primary Key.

**Step 3:** Define the ”leftover” keys. The remaining Candidate Keys that are unique but not ”Primary” are designated as Alternate Keys.

**Step 4:** Evaluate the options. A Foreign Key is used for relationships, and a Super Key is a broader set that may contain unnecessary columns. Therefore, the specific definition in the question refers to the Alternate Key.

**Final Answer:** An Alternate Key is a candidate key not chosen as the primary key.

**Answer: (C)**

## Q41.

### Solution

**Concept: Relational Algebra - The Selection Operation**

Relational Algebra is a formal language used to model the operations performed on database tables. One of its most fundamental operations is **Selection**, denoted by the Greek letter Sigma ($\sigma$). This operation acts as a horizontal filter. It scans a table and extracts specific rows that satisfy a given mathematical condition (predicate). It is the relational algebra equivalent of the `WHERE` clause in SQL.

**Solution:**

**Step 1:** Define the scope of Selection. Selection ($\sigma$) works on the **rows** (tuples) of a relation. It does not reduce the number of columns; it only reduces the number of rows based on a criteria.

**Step 2:** Contrast with Projection. While Selection filters rows, Projection ($\pi$) filters columns. Confusion between these two is a common error in competitive exams.

**Step 3:** Identify the operation in the question. The question asks about the Selection ($\sigma$) operation.

**Step 4:** Match with the correct definition. Option (B) correctly identifies that it picks specific rows based on a condition.

**Step 5:** Example logic. $\sigma_{salary>50000}(Employee)$ would return only the records of employees earning more than 50,000.

**Final Answer:** The Selection ($\sigma$) operation is used to pick specific rows based on a condition.

**Answer: (B)**

## Q42.

### Solution

**Concept: Referential Integrity and Foreign Keys**

Referential Integrity is a set of rules that ensures the consistency of data between related tables. It guarantees that a value in one table (the referencing table) must exist in another table (the referenced table). This prevents ”orphaned” records and maintains the logical link between data entities, such as ensuring that an `Order` cannot be placed for a `Customer_ID` that does not exist in the `Customers` table.

**Solution:**

**Step 1:** Identify the mechanism of referential integrity. The primary tool used to enforce this integrity is the **Foreign Key**.

**Step 2:** Define Foreign Key logic. A Foreign Key is a column in one table that points to the Primary Key of another table. The database engine enforces that any value entered in the Foreign Key column must already exist in the Primary Key column of the target table.

**Step 3:** Evaluate other constraints. - Primary Key and Unique Key ensure **Entity Integrity** (uniqueness within a table). - Default ensures data entry ease but does not link tables.

**Step 4:** Conclusion. Since the question specifically asks for a ”Referential Integrity” constraint, the Foreign Key is the only appropriate choice.

**Final Answer:** The Foreign Key is a Referential Integrity constraint.

**Answer: (C)**

## Q43.

### Solution

**Concept: Relational Algebra - The Projection Operation**

The **Projection** operation, denoted by the Greek letter Pi ($\pi$), is used to select a subset of columns from a table while discarding the rest. It is a vertical filter. One of the most important properties of the Projection operation in formal Relational Algebra is that it automatically **removes duplicate rows** from the result set. This is because a ”relation” in set theory cannot contain duplicate elements.

**Solution:**

**Step 1:** Define the scope of Projection. Projection ($\pi$) focuses on selecting specific **attributes** (columns) of a table.

**Step 2:** Identify the duplicate removal rule. In pure Relational Algebra, after columns are removed, the remaining data is treated as a set. If two rows become identical because the differentiating columns were projected out, only one unique row is kept.

**Step 3:** Map to SQL. Projection is equivalent to `SELECT DISTINCT column_names FROM table`.

**Step 4:** Match with the question description. The question describes an operation that returns specified columns and removes duplicates, which is the exact definition of Projection.

**Final Answer:** The Projection operation returns specific columns and removes duplicates.

**Answer: (B)**

## Q44.

### Solution

**Concept: Star Topology and Central Hubs**

In computer networking, ”Topology” refers to the physical or logical arrangement of nodes. The **Star Topology** is currently the most popular configuration for Local Area Networks (LANs). In this setup, every individual node (computer, printer, server) is connected to a single central device, usually a **Hub** or a **Switch**. Communication between any two nodes must pass through this central controller.

**Solution:**

**Step 1:** Analyze the Star Topology structure. Each node has a dedicated point-to-point connection to the central hub. This creates a shape resembling a star.

**Step 2:** Evaluate performance and failure. If a single node’s cable fails, only that node is disconnected. However, if the central Hub/Switch fails, the entire network goes down.

**Step 3:** Compare with other topologies. - **Bus:** Nodes connect to a single backbone cable. - **Ring:** Nodes connect in a closed loop. - **Mesh:** Every node is connected to every other node.

**Step 4:** Conclusion. The presence of a ”central hub/switch” is the defining characteristic of a Star Topology.

**Final Answer:** The Star topology uses a central hub/switch to connect all nodes.

**Answer: (C)**

## Q45.

### Solution

**Concept: Physical Addressing and the MAC Address**

Every device on a network requires identification. There are two primary types of addresses: Logical (IP Address) and Physical (MAC Address). The **MAC (Media Access Control) Address** is a unique identifier assigned to the **Network Interface Card (NIC)** by the manufacturer. It is often referred to as a ”burned-in” address because it is hard-coded into the hardware and generally does not change, unlike an IP address which changes based on the network connection.

**Solution:**

**Step 1:** Identify the hardware-level address. The address permanently assigned to the NIC hardware is the MAC address.

**Step 2:** Understand the format. MAC addresses are typically 48-bit numbers represented as 12 hexadecimal digits (e.g., `00:1A:2B:3C:4D:5E`).

**Step 3:** Differentiate from IP Address. An IP address is assigned by software (DHCP or manual config) and helps in routing across different networks. A MAC address is used for delivery within the same local segment.

**Step 4:** Evaluate other options. - **Port Address:** Used to identify specific applications (e.g., Port 80 for HTTP). - **URL:** A human-readable web address.

**Final Answer:** The MAC Address is permanently assigned to the NIC.

**Answer: (B)**

## Q46.

### Solution

**Concept: Networking Devices - The Gateway**

In networking, different devices serve different layers of communication. While a Switch or Bridge connects segments within the same type of network, a **Gateway** is a much more powerful device (often implemented as software on a router). It acts as a ”translator” or protocol converter. Gateways operate at the upper layers of the OSI model and are essential when two networks with entirely different architectures, protocols, or data formats need to communicate with each other.

**Solution:**

**Step 1:** Identify the requirement. The device must connect ”different types of networks” that use ”different protocols.”

**Step 2:** Evaluate the options. - **Switch:** Connects devices within a single LAN. - **Repeater:** Amplifies signals to cover longer distances; it doesn’t look at protocols. - **Bridge:** Connects two similar network segments.

**Step 3:** Define the Gateway’s role. A Gateway is specifically designed to handle protocol mismatch. For example, it can connect a local AppleTalk network to a TCP/IP-based internet.

**Step 4:** Conclusion. Because it handles the translation between differing protocols, the Gateway is the correct choice.

**Final Answer:** A Gateway is used to connect networks with different protocols.

**Answer: (C)**

## Q47.

### Solution

**Concept: Understanding Digital Footprints**

As we navigate the digital world, every action we take—from posting on social media and sending emails to simply visiting a website—leaves a trace. This cumulative record of an individual's online activities is known as a **Digital Footprint**. There are two types: **Active** (data you intentionally share, like a post) and **Passive** (data collected without your active involvement, like your IP address or browsing history). Once created, a digital footprint is often permanent and searchable.

**Solution:**

**Step 1:** Define the term. The trail of data left behind during internet usage is the "Digital Footprint."

**Step 2:** Analyze the components. This includes cookies, login records, social media interactions, and even search engine queries.

**Step 3:** Evaluate the options. - **Cyber Trail** and **Data Shadow** are descriptive terms but not the standardized terminology used in the IP curriculum. - **Web History** is specific to a single browser's local record, whereas a digital footprint is the global presence of your data.

**Step 4:** Significance. Understanding your digital footprint is vital for maintaining privacy and a positive online reputation.

**Final Answer:** The trail of data left behind is called a Digital Footprint.

**Answer: (A)**

## Q48.

### Solution

**Concept: Intellectual Property Rights (IPR)**

Intellectual Property refers to creations of the mind: inventions, literary and artistic works, designs, symbols, names, and images used in commerce. **IPR** are the legal rights given to the creator to protect their invention or creation for a certain period. Common forms of IPR include Patents (for inventions), Copyrights (for artistic works), and Trademarks (for brand identity). IPR encourages innovation by ensuring creators can benefit financially from their work without fear of immediate unauthorized copying.

**Solution:**

**Step 1:** Identify the context of the right. The right is granted to an ”inventor or creator” for their ”invention/work.”

**Step 2:** Define the term. This is the literal definition of Intellectual Property Rights.

**Step 3:** Rule out distractors. - **Plagiarism:** This is the act of stealing work (an ethical/legal violation), not a ”right.” - **Open Source:** This is a philosophy where the creator *allows* others to modify and share the work freely. - **Privacy Policy:** A document explaining how a company handles user data.

**Step 4:** Conclusion. IPR is the legal framework that protects original creations.

**Final Answer:** The legal right granted is known as Intellectual Property Right (IPR).

**Answer: (B)**

## Q49.

### Solution

**Concept: Environmental Impact of E-waste**

Electronic waste, or **E-waste**, refers to discarded electrical or electronic devices. Used electronics which are destined for refurbishment, reuse, resale, or salvage-recycling through material recovery are also considered e-waste. Improper disposal is a major environmental concern because electronics contain hazardous materials like lead, mercury, and cadmium, which can leach into the soil and groundwater if not handled by specialized recycling facilities.

**Solution:**

**Step 1:** Identify the objects mentioned. Mobile phones and computers are electronic devices.

**Step 2:** Analyze the problem. The question focuses on ”improper disposal” of these physical objects.

**Step 3:** Categorize the waste. Waste resulting from electronics is termed E-waste.

**Step 4:** Evaluate the options. - **Cybercrime** and **Phishing** are digital crimes, not physical waste issues. - **Identity Theft** is a result of data loss, but the physical act of throwing away a phone primarily contributes to environmental e-waste.

**Final Answer:** Improper disposal of electronics leads to E-waste.

**Answer: (B)**

## Q50.

### Solution

**Concept: Cybercrime - Identity Theft**

Identity Theft is a type of cybercrime where a criminal wrongfully obtains and uses another person's personal data (such as Aadhaar numbers, credit card details, or passwords) in some way that involves fraud or deception, typically for economic gain. In the digital age, this often happens through phishing, hacking, or social engineering. It is one of the most serious societal impacts of the internet, leading to financial loss and reputational damage for victims.

**Solution:**

**Step 1:** Define the criminal act. The act involves "stealing personal information" to "commit fraud."

**Step 2:** Identify the specific term. When you assume someone else's persona using their data, it is called **Identity Theft**.

**Step 3:** Contrast with other terms. - **Hacking:** The act of gaining unauthorized access (which may *lead* to identity theft, but is a different action). - **Spamming:** Sending unsolicited bulk messages. - **Trolling:** Posting inflammatory content to upset people online.

**Step 4:** Conclusion. Fraudulent use of personal information is the hallmark of Identity Theft.

**Final Answer:** Stealing personal information for fraud is known as Identity Theft.

**Answer: (B)**

# Answer Key

| Q | Ans | Q | Ans | Q | Ans | Q | Ans | Q | Ans |
|---|---|---|---|---|---|---|---|---|---|
| 1 | A | 2 | B | 3 | C | 4 | B | 5 | C |
| 6 | B | 7 | C | 8 | B | 9 | B | 10 | A |
| 11 | B | 12 | B | 13 | B | 14 | B | 15 | B |
| 16 | A | 17 | B | 18 | C | 19 | B | 20 | C |
| 21 | B | 22 | A | 23 | B | 24 | B | 25 | C |
| 26 | B | 27 | B | 28 | D | 29 | B | 30 | C |
| 31 | A | 32 | B | 33 | C | 34 | A | 35 | B |
| 36 | C | 37 | B | 38 | B | 39 | C | 40 | C |
| 41 | B | 42 | C | 43 | B | 44 | C | 45 | B |
| 46 | C | 47 | A | 48 | B | 49 | B | 50 | B |