CCEE Mock- I | ADS & DBT

Total points 25/40

The respondent's email (**amolgavit158121@gmail.com**) was recorded on submission of this form.

0 of 0 points

Name *

Amol Gavit

PRN (12 Digits) *

250240320013

Centre * Dropdown

Kharghar

Questions 25 of 40 points

✓ Choose the equivalent prefix form of the following expression * 1/1

(a+(b-c))*((d-e)/(f+g-h))

- (●) A. *+a-bc/-de-+fgh ✓
- ( ) B. *+a-bc-/de-+fgh
- ( ) C. *+a-bc/-ed-+fgh
- ( ) D. *+ab-c/-de-+fgh

Explanation:
To convert the given infix expression (a+(b-c))*((d-e)/(f+g-h)) into prefix notation, follow these steps:

Identify the operators and operands
Operators: +, -, *, /
Operands: a, b, c, d, e, f, g, h

Apply operator precedence and associativity
Parentheses dictate the order of operations.

Multiplication (*) and division (/) have higher precedence than addition (+) and subtraction (-).

Convert to prefix notation using Preorder Traversal

Start from the root operator (*).

Convert subexpressions recursively.

Final Prefix Expression:

* + a - b c / - d e - + f g h
* is the root operator.

+ a - b c represents the left subtree.

/ - d e - + f g h represents the right subtree.

Why Other Options Are Incorrect:
Option B (*+a-bc-/de-+fgh) → Incorrect order of / and - d e.

Option C (*+a-bc/-ed-+fgh) → Incorrect order of - e d.

Option D (*+ab-c/-de-+fgh) → Incorrect placement of + ab.

Notation	Example	Operator Placement	Evaluation Complexity
Infix	A + B	Between operands	Requires precedence rules
Prefix	+ A B	Before operands	Easier for computers to parse
Postfix	A B +	After operands	Ideal for stack-based evaluation

Explanation:
Data Manipulation Language (DML) is used in application programs to retrieve, insert, update, and delete data from a database management system (DBMS).

It allows users to interact with the database by performing queries and modifications on stored data.

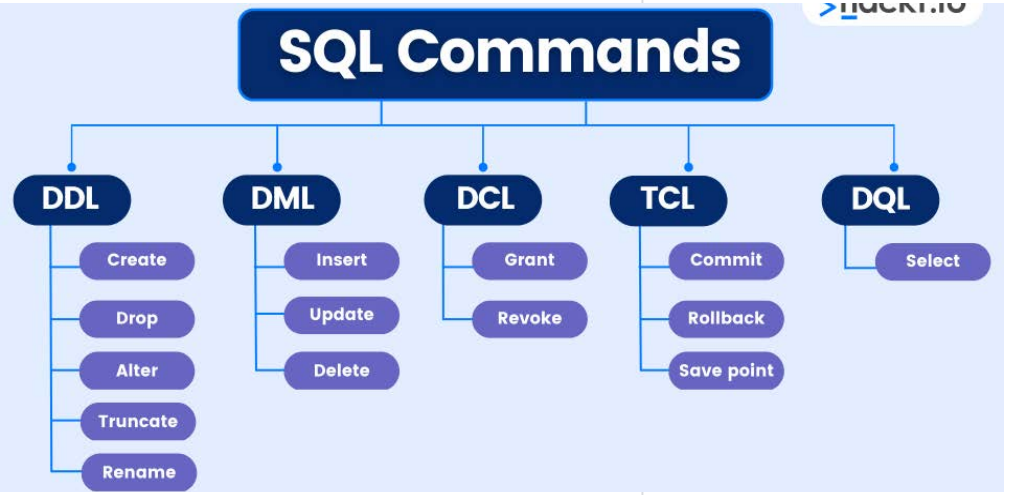
Common DML commands include:

SELECT → Retrieves data from the database.
INSERT → Adds new records to a table.
UPDATE → Modifies existing records.
DELETE → Removes records from a table.

Data Definition Language (DDL)
Data Query Language (DQL)
Data Manipulation Language (DML)
Transaction Control Language (TCL)
Data Control Language (DCL)

✓ Which of the following is used in the application programs to request data from the database management system? *1/1

- (●) 1. Data Manipulation language
- ( ) 2. Data Definition Language
- ( ) 3. Data Control Language
- ( ) 4. All of the above



✓ Which one of the following given statements possibly contains the error? * 1/1

- ( ) 1. select * from emp where empid = 10003;
- ( ) 2. select empid from emp where empid = 10006;
- ( ) 3. select empid from emp;
- (●) 4. select empid where empid = 1009 and Lastname = 'GELLER'; ✓

✓ If an array A contains the items 10, 4, 7, 23, 67, 12 and 5 in that order, what will be the resultant array A after third pass of insertion sort? *1/1

- A. 67,12,10,5,4,7,23
- B. 4,7,10,23,67,12,5
- C. 4,5,7,67,10,12,23
- D. 10,7,4,67,23,12,5

Explanation:
Insertion Sort works by building a sorted array one item at a time. It repeatedly picks an element and places it in its correct position relative to the already sorted portion of the array.

Step-by-Step Execution:
Given array: [10, 4, 7, 23, 67, 12, 5]

1 First Pass:
Compare 10 and 4, swap them.
Result: [4, 10, 7, 23, 67, 12, 5]

2 Second Pass:
Compare 10 and 7, swap them.
Result: [4, 7, 10, 23, 67, 12, 5]

3 Third Pass:
Compare 10 and 23, no swap needed.
Result: [4, 7, 10, 23, 67, 12, 5]

✓ All aggregate functions except _____ ignore null values in their input collection. *1/1

- a) Count(attribute)
- b) Count(*)
- c) Avg
- d) Sum

Explanation:
In SQL, aggregate functions process multiple rows and return a single value.

Most aggregate functions ignore NULL values in their calculations, meaning they only consider non-null entries.

COUNT() is the exception*—it counts all rows, including those with NULL values.

Behavior of Aggregate Functions with NULL Values:

Function	Effect of NULL Values
COUNT(attribute)	Ignores NULL values, counts only non-null entries.
COUNT()*	Includes NULL values, counting all rows.
AVG()	Ignores NULL values when calculating the average.
SUM()	Ignores NULL values when summing values.

✗ What does the following condition do? *
check(name in('Ryan', 'Cristiano', 'Leo'))

0/1

- (●) a) The condition checks whether the name attribute includes the three mentioned names ✗
- ( ) b) The condition allows the name attribute to possess only the three mentioned names
- ( ) c) The condition checks whether the given names are sub-strings in at least one of the values
- ( ) d) None of the mentioned

Correct answer

- (●) b) The condition allows the name attribute to possess only the three mentioned names

Explanation:
The CHECK constraint in SQL is used to enforce rules on column values.

The condition CHECK(name IN ('Ryan', 'Cristiano', 'Leo')) ensures that the name attribute can only take one of the specified values (Ryan, Cristiano, or Leo).

If an attempt is made to insert a different name, the database rejects the operation, maintaining data integrity.

Why Other Options Are Incorrect:
Option a (Checks whether the name attribute includes the three mentioned names) → Incorrect, because it restricts values rather than checking inclusion.

Option c (Checks whether the given names are substrings in at least one of the values) → Incorrect, because IN does not perform substring matching.

✗ A Boolean data type that can take values true, false, and_____ * 0/1

- a) 1
- b) 0
- c) Null
- d) Unknown

Explanation:
In SQL and certain database systems, the Boolean data type can take values TRUE, FALSE, and UNKNOWN.

The UNKNOWN value is represented by NULL, meaning that when a Boolean expression evaluates to NULL, it is considered unknown rather than explicitly true or false.

This is part of three-valued logic (3VL) used in SQL, where expressions can be TRUE, FALSE, or UNKNOWN

Correct answer

- d) Unknown

✓ Prim's algorithm and Kruskal's algorithm are used for: *

- a) Finding the shortest path between two nodes in a graph
- b) Finding the maximum flow in a network
- c) Finding the minimum spanning tree in a graph
- d) Sorting elements in an array

Explanation:
Prim's Algorithm and Kruskal's Algorithm are both used to find the Minimum Spanning Tree (MST) of a weighted, connected, and undirected graph.

An MST is a subset of edges that connect all vertices with the minimum possible total edge weight, ensuring no cycles.

Why Other Options Are Incorrect:
Option a (Finding the shortest path between two nodes in a graph) → Incorrect, because Dijkstra's Algorithm or Bellman-Ford Algorithm are used for shortest paths.

Option b (Finding the maximum flow in a network) → Incorrect, because Ford-Fulkerson Algorithm is used for maximum flow problems.

Option d (Sorting elements in an array) → Incorrect, because sorting is done using algorithms like Merge Sort, Quick Sort, or Heap Sort.

Comparison of Prim's and Kruskal's Algorithms:

Algorithm	Approach	Best for
Prim's Algorithm	Starts from a single vertex and grows the MST by adding the smallest edge that connects to an unvisited vertex.	Dense graphs
Kruskal's Algorithm	Sorts all edges by weight and adds them one by one, ensuring no cycles.	Sparse graphs

✓ What is the result of the following query? *1/1

```
SELECT studname
FROM college
WHERE marks > SOME (SELECT marks FROM student WHERE section =
'c');
```

- a) The query gives all the studnames for which marks are greater than all the students in section c
- (●) b) The query gives all the studnames for which the marks are greater than at least on student in section c ✓
- c) The query gives all the studnames for which the marks are less than all the students in section c
- d) The query is syntactically incorrect

Explanation:
The SOME operator in SQL is used to compare values against at least one value in a subquery.

The condition marks > SOME (SELECT marks FROM student WHERE section = 'c') means that the query selects students whose marks are greater than at least one student in section 'c'.

This differs from ALL, which would require the marks to be greater than every student in section 'c'

Why Other Options Are Incorrect:
Option a (Greater than all students in section c) → Incorrect, because SOME checks at least one, not all.

Option c (Less than all students in section c) → Incorrect, because SOME is used for greater than comparisons.

Option d (Syntactically incorrect) → Incorrect, because the query is valid SQL syntax.

Stack Operations:
push(54) → Stack: [54]

push(52) → Stack: [54, 52]

pop() → Removes 52, Stack: [54]

push(55) → Stack: [54, 55]

push(62) → Stack: [54, 55, 62]

s = pop() → Removes 62, so s = 62

Queue Operations:
enqueue(21) → Queue: [21]

enqueue(24) → Queue: [21, 24]

dequeue() → Removes 21, Queue: [24]

enqueue(28) → Queue: [24, 28]

enqueue(32) → Queue: [24, 28, 32]

q = dequeue() → Removes 24, so q = 24

Final Calculation:
S + Q = 62 + 24 = 86

✓ Consider the following sequence of operations on an empty stack. *1/1
push(54); push(52); pop(); push(55); push(62); s=pop();

Consider the following sequence of operations on an empty queue.
enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32);
q=dequeue();

The value of s+q is _____.

- A. 94
- B. 83
- C. 79
- D. 86 (selected)

1. Queue (FIFO - First In, First Out)
Definition: A queue follows the FIFO principle, meaning the first element added is the first one to be removed.

Operations:
Enqueue → Adds an element to the rear.
Dequeue → Removes the front element.
Front → Returns the front element without removing it.

Example Usage:
java
```
Queue<Integer> queue = new LinkedList<>();
queue.add(10);
queue.add(20);
int front = queue.remove();  // Removes 10
```

Use Cases:
Task scheduling (CPU scheduling).
Breadth-First Search (BFS) in graphs.
Print spooling in operating systems.

2. Stack (LIFO - Last In, First Out)
Definition: A stack follows the LIFO principle, meaning the last element added is the first one to be removed.

Operations:
Push → Adds an element to the top.
Pop → Removes the top element.
Peek → Returns the top element without removing it.

Example Usage:
java
```
Stack<Integer> stack = new Stack<>();
stack.push(10);
stack.push(20);
int top = stack.pop();  // Removes 20
```

Use Cases:
Function call management (Call Stack).
Undo/Redo operations in applications.
Expression evaluation (Postfix, Prefix).

1. Queue Operations (FIFO - First In, First Out)

Enqueue: Adds an element to the rear of the queue.
Dequeue: Removes the front element from the queue.
Front: Returns the front element without removing it.
isEmpty: Checks if the queue is empty.

2. Stack Operations (LIFO - Last In, First Out)

Push: Adds an element to the top of the stack.
Pop: Removes the topmost element from the stack.
Peek (Top): Returns the top element without removing it.
isEmpty: Checks if the stack is empty.

✗ What does p indicate in the following data type? *
time(p)

- a) The amount of delay that needs to be added to the time
- b) The number of fractional digits for the seconds
- c) The maximum number of allowed hours
- d) None of the mentioned

Correct answer

- b) The number of fractional digits for the seconds

Explanation:
In SQL, the time(p) data type allows specifying fractional seconds precision using the parameter p.

The value of p determines how many decimal places are retained in the seconds field.

The allowed range for p is typically 0 to 6, meaning up to microseconds precision can be stored

Example Usage:
sql
CREATE TABLE event_log (
event_time TIME(3) -- Stores time with 3 fractional digits (milliseconds)
);
If p = 3, the stored time format would be HH:MM:SS.sss (milliseconds).

If p = 6, the format would be HH:MM:SS.ssssss (microseconds).

Why Other Options Are Incorrect:
Option a (Amount of delay added to time) → Incorrect, because p controls precision, not delay.

Option c (Maximum number of allowed hours) → Incorrect, because time is limited to 24-hour format, and p does not affect hours.

✓ The preorder traversal of a binary search tree is 15,10,12,11,20,18,16,19.
Which one of the following is the postorder traversal of the tree?

- A. 10,11,12,15,16,18,19,20
- B. 11,12,10,16,19,18,20,15
- C. 20,19,18,16,15,12,11,10
- D. 19,16,18,20,11,12,10,15

Explanation:
To determine the postorder traversal from the given preorder traversal, follow these steps:

Step 1: Construct the Binary Search Tree (BST)
Given preorder traversal: 15, 10, 12, 11, 20, 18, 16, 19

Root: 15

Left Subtree: 10 → 12 → 11

Right Subtree: 20 → 18 → 16 → 19

Step 2: Apply Postorder Traversal (Left → Right → Root)
Left subtree traversal: 11, 12, 10

Right subtree traversal: 16, 19, 18, 20

Root node: 15

Thus, the postorder traversal is: 11, 12, 10, 16, 19, 18, 20, 15

Traversal Type	Order
Preorder	Root → Left → Right
Inorder	Left → Root → Right
Postorder	Left → Right → Root

✓ Which of the following should be used to find all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester . *1/1

a) SELECT DISTINCT course id
FROM SECTION
WHERE semester = 'Fall' AND YEAR= 2009 AND
course id NOT IN (SELECT course id
FROM SECTION
WHERE semester = 'Spring' AND YEAR= 2010);

b)SELECT DISTINCT course_id
FROM instructor
WHERE name NOT IN ('Fall', 'Spring');

c)(SELECT course id
FROM SECTION
WHERE semester = 'Spring' AND YEAR= 2010)

d)SELECT COUNT (DISTINCT ID)
FROM takes
WHERE (course id, sec id, semester, YEAR) IN (SELECT course id, sec id, semester, YEAR
FROM teaches
WHERE teaches.ID= 10101);

Explanation:
The query retrieves all courses taught in Fall 2009 but not in Spring 2010 using the NOT IN clause.

The subquery selects all courses from Spring 2010, and the outer query filters out those courses from the Fall 2009 list.

- (●) A ✓
- ( ) D

Explanation:
The in-order successor of a node in a Binary Search Tree (BST) is the smallest node in its right subtree.

The given preorder traversal of the BST is: 15, 10, 12, 11, 20, 18, 16, 19.

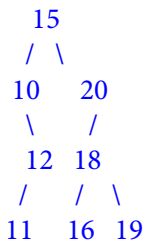
To determine the in-order successor of 15, we follow in-order traversal (Left → Root → Right).

- C
- B

✕ What is the in-order successor of 15 in the given binary search tree? * 0/1

- A. 18
- B. 6
- C. 17
- D. 20

Step-by-Step Process:
1) Construct the BST from the given preorder traversal:



2) Perform In-Order Traversal (Left → Root → Right):
Result: 10 → 11 → 12 → 15 → 16 → 17 → 18 → 19 → 20 3 Find the successor of 15:

The next node after 15 in in-order traversal is 17.

Correct answer

- C. 17

1. Preorder Traversal (Root → Left → Right)
Order: Visit the root first, then traverse the left subtree, followed by the right subtree.

Preorder Output: 15 → 10 → 12 → 11 → 20 → 18 → 16 → 19

Use Case: Used for copying a tree structure or evaluating expressions in prefix notation.

2. Inorder Traversal (Left → Root → Right)
Order: Traverse the left subtree first, then visit the root, followed by the right subtree.

Inorder Output: 10 → 11 → 12 → 15 → 16 → 18 → 19 → 20

Use Case: Retrieves sorted data in Binary Search Trees (BSTs).

3. Postorder Traversal (Left → Right → Root)
Order: Traverse the left subtree first, then the right subtree, and visit the root last.

Postorder Output: 11 → 12 → 10 → 16 → 19 → 18 → 20 → 15

Use Case: Used for deleting nodes in a tree or evaluating expressions in postfix notation.

Rotations to Maintain Balance
AVL trees use four types of rotations to maintain balance:

LL Rotation (Single Right Rotation)

RR Rotation (Single Left Rotation)

LR Rotation (Left-Right Rotation)

RL Rotation (Right-Left Rotation)

✕ What is the balance factor of a node in an AVL tree? * 0/1

- ○ a) The difference between the heights of its left and right subtrees
- ◉ b) The sum of the heights of its left and right subtrees ✕
- ○ c) The ratio between the heights of its left and right subtrees
- ○ d) The total number of nodes in its left and right subtrees

Correct answer

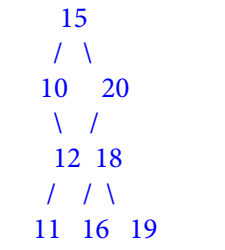
- ◉ a) The difference between the heights of its left and right subtrees

Explanation:
The balance factor of a node in an AVL tree is calculated as:

Balance Factor = Height of Left Subtree − Height of Right Subtree
The AVL tree is a self-balancing binary search tree, meaning it maintains a balance factor of -1, 0, or 1 for every node to ensure efficient operations.

If the balance factor exceeds 1 or falls below -1, the tree becomes unbalanced, requiring rotation operations to restore balance.

Example Calculation:
Consider the following AVL tree:



Balance Factor of Node 15:

Left Subtree Height = 2 (10 → 12 → 11)

Right Subtree Height = 2 (20 → 18 → 16, 19)

Balance Factor = 2 - 2 = 0

✓ The given Query can be replaced with _____: * 1/1

```
SELECT name
FROM instructor1
WHERE salary <= 100000 AND salary >= 90000;
```

```
a.SELECT name
FROM instructor1
WHERE salary BETWEEN 100000 AND 90000
```

```
b.SELECT name
FROM instructor|
WHERE salary BETWEEN 90000 AND 100000;
```

```
c.SELECT name
FROM instructor1
WHERE salary BETWEEN 90000 AND 100000;
```

```
d.SELECT name
FROM instructor!
WHERE salary <= 90000 AND salary>=100000;
```

Explanation:
The BETWEEN operator in SQL is used to filter values within a specified range, including both boundary values.

- ○ A
- ○ B
- ◉ C ✓

D

Explanation:
Breadth-First Search (BFS) is a graph traversal algorithm that explores nodes level by level using a queue.

In an unweighted graph, BFS ensures the shortest path between any two nodes because it visits all neighbors before moving to the next level.

BFS has a time complexity of O(V + E), where V is the number of vertices and E is the number of edges

✗ Which of the following statements about BFS (Breadth-First Search) is true? *0/1

a) BFS guarantees the shortest path between any two nodes in an unweighted graph.

b) BFS has a worst-case time complexity of O(nlog(n)).

c) BFS is a recursive algorithm that explores the deepest levels of a graph first.

d) BFS requires the use of a priority queue data structure for efficient traversal. ✗

Correct answer

a) BFS guarantees the shortest path between any two nodes in an unweighted graph.

Why Other Options Are Incorrect:
Option b (O(nlog(n)) complexity) → Incorrect, because BFS runs in O(V + E), not O(nlog(n)).

Option c (Recursive algorithm exploring deepest levels first) → Incorrect, because Depth-First Search (DFS) is recursive and explores deeper levels first, not BFS.

Option d (Requires a priority queue) → Incorrect, because BFS uses a queue, not a priority queue. Dijkstra's Algorithm uses a priority queue for shortest paths

✓ State true or false: We cannot write a where clause under an update command *1/1

- a) True
- b) False
- c) Truth and dare game lag rha h
- d) No game only padhai

Explanation:
The UPDATE command in SQL is used to modify existing records in a table.

A WHERE clause can be used in an UPDATE statement to specify which rows should be updated.

Without a WHERE clause, all rows in the table will be updated, which may lead to unintended changes.

✓ 'AS' clause is used in SQL for ? * 1/1

- (A) Selection operation.
- (B) Rename operation. ✓
- (C) Join operation.
- (D) Projection operation.

Explanation:
The AS clause in SQL is used to rename columns or tables temporarily within a query.

It helps improve query readability and allows assigning meaningful aliases to columns or tables.

The alias exists only for the duration of the query and does not affect the actual database schema.

Explanation:
Dijkstra's Algorithm is a single-source shortest path algorithm that efficiently finds the shortest path from a given starting point (Intersection A) to a destination (Intersection B) in a weighted graph.

It works by maintaining a priority queue to explore the shortest known distance to each node, ensuring the optimal path is found.

Since the transportation network consists of roads with distances, Dijkstra's algorithm is ideal for finding the shortest route.

✗ Scenario: *0/1

You are given a transportation network map for a city, consisting of various intersections and roads connecting them. Each road has a certain length associated with it, representing the distance between the intersections it connects. You are tasked with finding the shortest path for a delivery truck to travel from Intersection A to Intersection B.

Based on the given scenario, which algorithm would you use to efficiently find the shortest path for the delivery truck?

- a) Dijkstra's algorithm
- b) Kruskal's algorithm ✗
- c) Prim's algorithm
- d) Floyd-Warshall algorithm

Correct answer

- a) Dijkstra's algorithm

Why Other Options Are Incorrect:
Option b (Kruskal's Algorithm) → Incorrect, because Kruskal's algorithm is used for Minimum Spanning Tree (MST) problems, not shortest paths.

Option c (Prim's Algorithm) → Incorrect, because Prim's algorithm also finds MST, not shortest paths.

Option d (Floyd-Warshall Algorithm) → Incorrect, because Floyd-Warshall finds shortest paths between all pairs of nodes, making it inefficient for single-source shortest path problems.

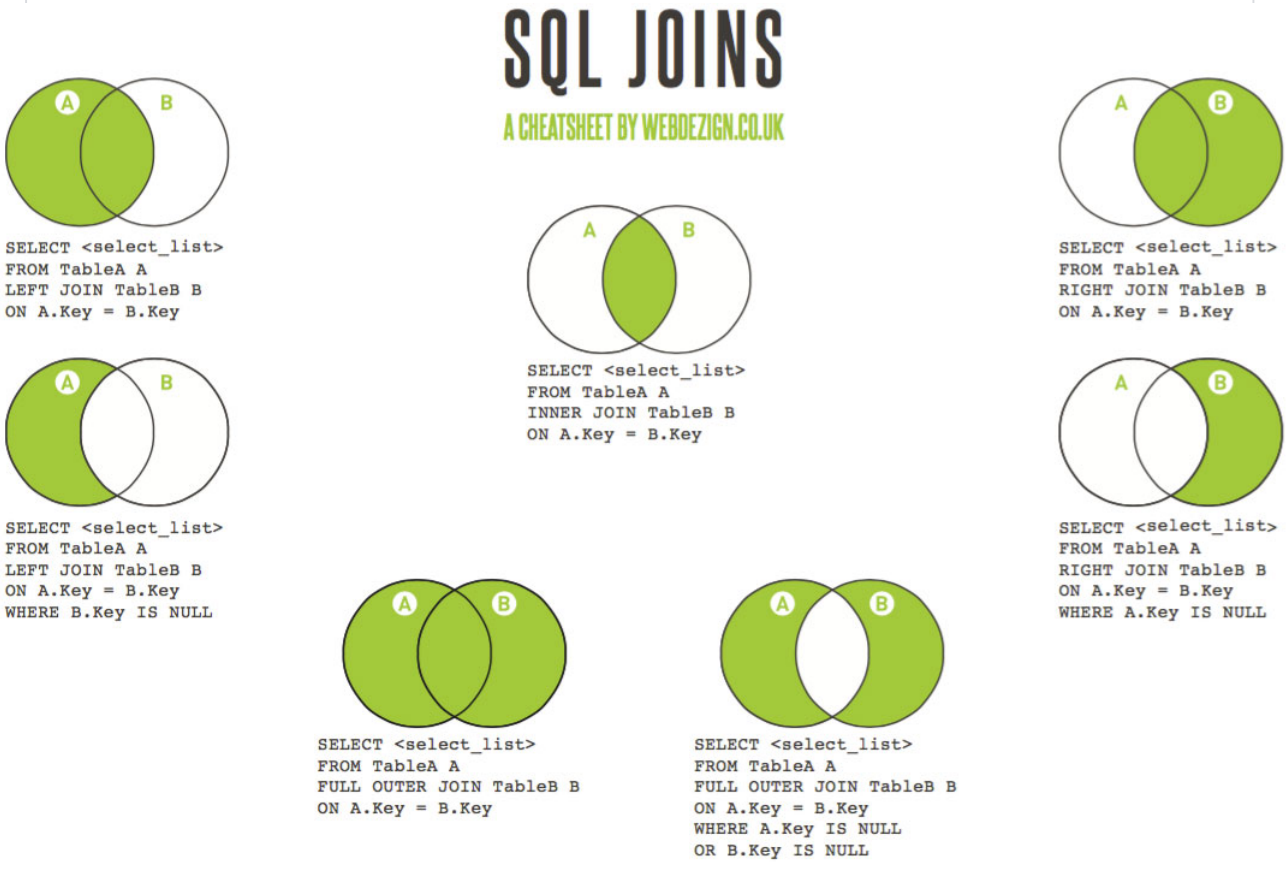
Comparison Table

Algorithm	Purpose	Approach	Best for	Time Complexity
Dijkstra	Shortest path from one source	Greedy, priority queue	Road networks, routing	$O(n + e \log n)$
Kruskal	Minimum Spanning Tree (MST)	Greedy, edge sorting	Sparse graphs	$O(e \log e)$
Prim	Minimum Spanning Tree (MST)	Greedy, priority queue	Dense graphs	$O(n^2)$ or $O(e + n \log n)$
Floyd-Warshall	Shortest paths between all pairs	Dynamic programming	Dense graphs	$O(n^3)$

✓ The given Query can also be replaced with_____: * 1/1

```
SELECT name, course_id
FROM instructor, teaches
WHERE instructor_ID= teaches_ID;
```

- ○ 1. Select name,course_id from teaches,instructor where instructor_id=course_id;
- ◉ 2. Select name, course_id from instructor natural join teaches; ✓
- ○ 3. Select name, course_id from instructor;
- ○ 4. Select course_id from instructor join teaches;



✕ create view studentdet *
select ID, address, name
from student;

What is the result of the above query?

- (●) a) It creates a view named studentdet with 3 attributes
- ( ) b) It creates a view named studentdet with 1 attribute
- ( ) c) It creates a view named ID with 2 attributes
- ( ) d) It is syntactically wrong and does not give a result

Correct answer

- (●) d) It is syntactically wrong and does not give a result

Explanation:
The given query attempts to create a view named studentdet using the SELECT statement.

However, the correct syntax for creating a view in SQL is:

sql
CREATE VIEW studentdet AS
SELECT ID, address, name
FROM student;
Since the query misses the CREATE VIEW statement, it is syntactically incorrect and will not execute properly.

Types of Views:
Type Description
Simple View Based on a single table, without aggregations.
Complex View Based on multiple tables, may include joins and aggregations.
Updatable View Allows modifications to underlying tables.
Read-Only View Prevents updates to underlying tables.

What is a View in SQL?
A view in SQL is a virtual table that is based on the result of an SQL query. It does not store data itself but dynamically presents data from one or more tables.

Key Features of Views:
Acts like a table but does not store data physically.

Simplifies complex queries by encapsulating them into a single virtual table.

Enhances security by restricting access to specific columns or rows.

Always up-to-date, as it retrieves fresh data whenever queried.

Explanation:
The WITH clause in SQL is used to define a Common Table Expression (CTE), which creates a temporary relation that exists only within the execution scope of a single query.

In the given query:

sql
WITH max_budget (VALUE) AS (SELECT MAX(budget) FROM department)
SELECT budget
FROM department, max_budget
WHERE department.budget = max_budget.value;
max_budget is a temporary relation that stores the maximum budget from the department table.

It is used only within this query and is automatically dropped once the query execution completes.

✕ WITH max_budget (VALUE) AS *0/1
(SELECT MAX(budget)
FROM department)
SELECT budget
FROM department, max_budget
WHERE department.budget = MAX budget.value;
In the query given above which one of the following is a temporary relation?

- a) Budget ✕
- b) Department
- c) Value
- d) Max_budget

Why Other Options Are Incorrect:
Option a (Budget) → Incorrect, because budget is an attribute, not a temporary relation.

Option b (Department) → Incorrect, because department is a permanent table in the database.

Option c (Value) → Incorrect, because VALUE is just an alias for the column inside the temporary relation max_budget.

Correct answer

- d) Max_budget

Explanation:
The unique construct in SQL returns true if the argument in the subquery contains no duplicate values.

It ensures that the subquery results are distinct, meaning each value appears only once.

This is useful when checking for uniqueness in a dataset or validating constraints.

✕ The _____ construct returns true if the argument in the sub-query is void of duplicates *0/1

- a) not null
- b) not unique
- c) unique
- d) null ✕

Why Other Options Are Incorrect:
Option a (not null) → Incorrect, because NOT NULL ensures values are not empty, but does not check for duplicates.

Option b (not unique) → Incorrect, because it does not exist as a valid SQL construct.

Option d (null) → Incorrect, because NULL represents missing values, not uniqueness.

Correct answer

- c) unique

Explanation:
Depth-First Search (DFS) is a graph traversal algorithm that explores nodes deeply before backtracking.

It uses a stack to keep track of visited vertices, ensuring that the most recently visited node is processed first.

The stack helps in backtracking when a dead-end is reached, allowing DFS to explore other paths.

✗ In DFS (Depth-First Search), which data structure is commonly used to keep track of visited vertices? *0/1

- a) Queue
- b) Stack
- c) Priority queue
- d) Hash table ✗

Why Other Options Are Incorrect:
Option a (Queue) → Incorrect, because Breadth-First Search (BFS) uses a queue, not DFS.

Option c (Priority queue) → Incorrect, because priority queues are used in Dijkstra's Algorithm, not DFS.

Option d (Hash table) → Incorrect, because hash tables store visited nodes but do not control traversal order.

Correct answer

- b) Stack

✓ Divide and Conquer Algorithm is characterized by: * 1/1

- a) Breaking a problem into smaller subproblems, solving them recursively, and combining their solutions ✓
- b) Taking the best immediate choice at each step to reach the optimal solution
- c) Iteratively optimizing the solution until reaching the desired result
- d) Using dynamic programming techniques to solve complex problems efficiently

Explanation:
In a relational database, an attribute refers to a column in a table.

Each column represents a specific property or characteristic of an entity.

For example, in a STUDENT table, attributes could be Student_ID, Name, and Age.

✓ In the relational table, which of the following can also be represented by the term "attribute"? *1/1

- 1. Entity
- 2. Row
- (●) 3. Column ✓
- 4. Both B &C

Why Other Options Are Incorrect:
Option 1 (Entity) → Incorrect, because an entity represents a real-world object (e.g., a student or employee), not an attribute.

Option 2 (Row) → Incorrect, because a row (or tuple) represents a single record, not an attribute.

Option 4 (Both B & C) → Incorrect, because only columns represent attributes, not rows.

1. Table
Relation (Used in relational databases)

Dataset (In analytical databases)

Entity Set (In ER models)

2. Row
Tuple (Formal term in relational databases)

Record (Common in database management systems)

Instance (Used in object-oriented databases)

3. Column
Attribute (Formal term in relational databases)

Field (Common in database management systems)

Property (Used in object-oriented databases)

✗ Which of the following statements about AVL trees is true? * 0/1

- a) AVL trees guarantee a worst-case time complexity of O(1) for search, insertion, and deletion operations.
- b) AVL trees maintain perfect balance at all times, resulting in a tree height of log(n). ✗
- c) AVL trees use rotation operations to maintain balance after insertions and deletions.
- d) AVL trees are efficient for maintaining sorted data, but not for searching or insertion.

Correct answer

- c) AVL trees use rotation operations to maintain balance after insertions and deletions.

Explanation:
AVL trees are self-balancing binary search trees (BSTs) that maintain balance using rotation operations after insertions and deletions.

The balance factor of each node is maintained within -1, 0, or 1, ensuring efficient operations.

If the balance factor exceeds ±1, the tree performs rotations to restore balance.

Why Other Options Are Incorrect:
Option a (O(1) time complexity) → Incorrect, because AVL trees have a worst-case time complexity of O(log n) for search, insertion, and deletion, not O(1).

Option b (Perfect balance at all times) → Incorrect, because AVL trees maintain near-perfect balance, but not absolute balance. The height is O(log n), not exactly log(n).

Option d (Not efficient for searching or insertion) → Incorrect, because AVL trees are highly efficient for searching, insertion, and deletion due to their balanced structure

✓ How many relations can a delete command operate on? *

- a) 0
- b) 1
- c) 2
- d) Infinitely many

Explanation:
The DELETE command in SQL is used to remove rows from a single table (relation) at a time.

It operates on one relation specified in the DELETE FROM clause.

If a WHERE condition is provided, only specific rows are deleted; otherwise, all rows in the table are removed.

Command Name	Effect	Rollback Support	Performance
DELETE	Removes specific rows	Yes (Can be rolled back)	Slow (Processes rows individually)
TRUNCATE	Removes all rows but keeps table structure	No (Auto-commit)	Faster than DELETE (Does not process rows one-by-one)
DROP	Removes the entire table & structure	No (Auto-commit)	Fastest (Completely removes table instantly)

Explanation:
Join (Inner Join) → Retrieves only matching rows from both tables based on the specified condition. If a row in one table does not have a corresponding match in the other table, it is excluded from the result.

Outer Join → Retrieves all rows from one or both tables, including unmatched rows. If a row does not have a match, it is included in the result with NULL values for missing columns.

✓ What is the difference between a join and an outer join operation? * 1/1

- a) There is no difference
- b) Join preserves a few tuples that are otherwise lost in the outer join
- c) Outer join preserves a few tuples that are otherwise lost in the join ✓
- d) An outer join can be used only on outer queries whereas a join operation can be used in Subqueries

Left Outer Join Returns all rows from the left table and matching rows from the right table. Unmatched rows from the left table appear with NULL values for right table columns.

Right Outer Join Returns all rows from the right table and matching rows from the left table. Unmatched rows from the right table appear with NULL values for left table columns.

Full Outer Join Returns all rows from both tables, filling in NULL values where matches are missing.

Explanation:
A max-heap is a complete binary tree where the value of each parent node is greater than or equal to the values of its children. The root node contains the largest value, and each subtree follows the same heap property.

Heap Structure (Tree Form):

```
        23
      /    \
    17      14
   /  \     /  \
  7    13  10   1
 / \   /
5   6 12
```

✗ Which one of the following sequences when stored in an array at locations A[1], . . . , A[10] forms a max-heap? *0/1

- A. 23, 17, 10, 6, 13, 14, 1, 5, 7, 12
- B. 23, 17, 14, 7, 13, 10, 1, 5, 6, 12
- C. 23, 17, 14, 6, 13, 10, 1, 5, 7, 15
- D. 23, 14, 17, 1, 10, 13, 16, 12, 7, 5 ✗

Correct answer

- B. 23, 17, 14, 7, 13, 10, 1, 5, 6, 12

Steps to Verify a Max-Heap:
1 Check the root node → The first element should be the largest (23). 2 Verify parent-child relationships → Each parent node should be greater than or equal to its children. 3 Ensure completeness → The tree must be filled level by level, from left to right.

Why Option B is Correct:
Root (23) is the largest

Children of 23: 17, 14 → Both are smaller

Children of 17: 7, 13 → Both are smaller

Children of 14: 10, 1 → Both are smaller

Children of 7: 5, 6 → Both are smaller

Children of 13: 12 → Smaller

Explanation:
A left outer join retrieves all rows from the left table and matching rows from the right table.

If a row in the left table has no matching row in the right table, the missing values from the right table are replaced with NULL.

This ensures that all rows from the left table are preserved, even if there is no corresponding match in the right table.

✓ If a left outer join is performed and the tuple on the left hand side does not match with the tuple on the right hand side, what happens to the values that are preserved on the left hand side? *1/1

- (●) a) They are given null values ✓
- ( ) b) They are given a random value
- ( ) c) The user is asked to enter data
- ( ) d) The query is declared invalid by the compiler

Why Other Options Are Incorrect:
Option b (Random value) → Incorrect, because SQL does not assign random values to missing data.

Option c (User is asked to enter data) → Incorrect, because SQL automatically assigns NULL, without user intervention.

Option d (Query is declared invalid) → Incorrect, because left outer joins are valid operations in SQL.

Explanation:
The LATERAL keyword in SQL allows a subquery in the FROM clause to reference columns from preceding tables in the same query.

This enables correlated subqueries, where the subquery can use values from the outer query.

Without LATERAL, subqueries in the FROM clause cannot access columns from preceding tables.

✗ The _____ keyword is used to access attributes of preceding tables or subqueries in the from clause. *0/1

- ( ) a) In
- ( ) b) Lateral
- (●) c) Having ✗
- ( ) d) With

Why Other Options Are Incorrect:
Option a (IN) → Incorrect, because IN is used for membership checks, not accessing preceding tables.

Option c (HAVING) → Incorrect, because HAVING filters aggregated results, not subqueries.

Option d (WITH) → Incorrect, because WITH defines Common Table Expressions (CTEs), not correlated subqueries.

Correct answer

- (●) b) Lateral

Explanation:
In SQL, when inserting a date value into a database, it must be enclosed in single quotes ('YYYY-MM-DD') to be recognized as a valid date format.

The format YYYY-MM-DD ensures consistency across different database systems and avoids ambiguity in date interpretation.

This format is widely used in MySQL, PostgreSQL, and SQL Server for storing and retrieving date values.

✗ What is the format of entering date into a database while inserting data into it? *0/1

- a) YYYY-MM-DD
- b) “YYYY-MM-DD” ✗
- c) ‘YYYY-MM-DD’
- d) “DD-MM-YYYY”

Why Other Options Are Incorrect:
Option a (YYYY-MM-DD) → Incorrect, because dates must be enclosed in single quotes when inserted.

Option b ("YYYY-MM-DD") → Incorrect, because double quotes are not used for date literals in SQL.

Option d ("DD-MM-YYYY") → Incorrect, because SQL databases primarily use YYYY-MM-DD format, not DD-MM-YYYY.

Correct answer

- c) ‘YYYY-MM-DD’

Explanation:
Nested subqueries can be used for comparing two different sets in SQL.

Set comparison is an important function of nested subqueries, allowing operations like IN, EXISTS, ALL, and SOME to compare values across multiple sets.

For example, a nested subquery can check whether a value exists in another dataset or compare aggregated results.

✓ State true or false : Nested Subqueries cannot be used for comparing two different sets *1/1

- a) True
- b) False ✓
- C) Compare krna paap h
- D) Gali kinaare saanp h

Why Option "True" is Incorrect:
Nested subqueries are designed for set comparisons, making the statement false.

SQL provides various operators (IN, EXISTS, ALL, SOME) to facilitate comparisons between different sets.

Example Usage:
sql
SELECT student_name
FROM students
WHERE marks > SOME (SELECT marks FROM student WHERE section = 'C');
This query retrieves students whose marks are greater than at least one student in section 'C'.

The nested subquery enables comparison between two different sets (students and student in section 'C').

Explanation:
Aggregate functions (e.g., SUM(), AVG(), COUNT()) can be used in the HAVING clause to filter grouped results.

They cannot be used in the WHERE clause because WHERE filters individual rows before aggregation, while HAVING filters after aggregation.

✓ Aggregate functions can be used in the select list or the_____clause of a select statement or subquery. They cannot be used in a _____ clause. *1/1

- a) Where, having
- b) Having, where ✓
- c) Group by, having
- d) Group by, where

Example Usage:

```
sql
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

Here, HAVING filters departments where the average salary is greater than 50000.

Using AVG(salary) > 50000 in WHERE would cause an error because aggregation happens after filtering, not before.

Why Other Options Are Incorrect:
Option a (Where, having) → Incorrect, because aggregate functions cannot be used in WHERE.

Option c (Group by, having) → Incorrect, because GROUP BY organizes data but does not filter aggregated results.

Option d (Group by, where) → Incorrect, because WHERE filters individual rows, not aggregated values.

Explanation:
A Greedy Algorithm makes decisions step by step, always choosing the best immediate option without considering future consequences.

It assumes that local optimal choices will lead to a global optimal solution.

Greedy algorithms are commonly used in optimization problems, such as Dijkstra's Algorithm, Kruskal's Algorithm, Huffman Coding, and Prim's Algorithm

✓ Greedy Algorithm is characterized by: * 1/1

- a) Breaking a problem into smaller subproblems, solving them recursively, and combining their solutions
- b) Taking the best immediate choice at each step to reach the optimal solution ✓
- c) Iteratively optimizing the solution until reaching the desired result
- d) Using dynamic programming techniques to solve complex problems efficiently

WHERE CLAUSE	HAVING CLAUSE
It is used to filter the records from the table based on the specified condition.	It is used to filter records from the groups based on the specified condition.
It can be used without the GROUP BY Clause.	It cannot be used without GROUP BY Clause
WHERE Clause implements in row operations.	HAVING Clause implements in column operation.
It cannot contain an aggregate function.	It can contain an aggregate function
It can be used with SELECT, UPDATE, DELETE statements.	It can only be used with SELECT statement.
WHERE Clause is used before GROUP BY Clause	HAVING Clause is used after GROUP BY Clause

Why Other Options Are Incorrect:
Option a (Breaking a problem into smaller subproblems) → Incorrect, because this describes Divide and Conquer, not Greedy Algorithms.

Option c (Iteratively optimizing the solution) → Incorrect, because Greedy Algorithms do not iterate to refine solutions—they make decisions once and move forward.

Option d (Using dynamic programming techniques) → Incorrect, because Dynamic Programming stores subproblem results for reuse, whereas Greedy Algorithms do not

Explanation:
A cursor in SQL is a database object that allows traversal over the rows of a query result set.

It enables row-by-row processing, which is useful when operations cannot be performed on the entire dataset at once.

Cursors are commonly used in stored procedures to iterate over query results.

✓ Cursor is a pointer to the result set of a ___. * 1/1

- C. Dataset
- A. Query ✓
- D. Index
- B. Table

Example Usage:
sql
DECLARE cursor_name CURSOR FOR
SELECT name, salary FROM employees;

OPEN cursor_name;
FETCH NEXT FROM cursor_name INTO @name, @salary;
This cursor retrieves name and salary from the employees table and processes them row by row.

Why Other Options Are Incorrect:
Option C (Dataset) → Incorrect, because a dataset refers to a collection of data, but a cursor specifically operates on a query result set.

Option D (Index) → Incorrect, because an index is used for fast data retrieval, but does not involve row-by-row traversal.

Option B (Table) → Incorrect, because a cursor does not point to an entire table, only to the result set of a query

Explanation:
A stack is the best data structure for checking balanced parentheses in an arithmetic expression.

It follows the Last In, First Out (LIFO) principle, which ensures that each opening parenthesis is matched with its corresponding closing parenthesis in the correct order.

The algorithm works by pushing opening parentheses onto the stack and popping them when a closing parenthesis is encountered.

If the stack is empty at the end, the parentheses are balanced; otherwise, they are unbalanced.

✓ The best data structure to check whether an arithmetic expression has balanced parenthesis is a: *1/1

- A. Queue
- B. Stack ✓
- C. Tree
- D. List

✓ The EXISTS keyword will be true if: * 1/1

- (●) a) Any row in the subquery meets the condition only ✓
- ( ) b) All rows in the subquery fail the condition only
- ( ) c) Both of these two conditions are met
- ( ) d) Neither of these two conditions is met

Explanation:
The EXISTS keyword in SQL is used to check whether a subquery returns any rows.

If the subquery returns at least one row, the EXISTS condition evaluates to TRUE.

If the subquery returns no rows, the EXISTS condition evaluates to FALSE.

1. Stored Procedure
Purpose: Used to perform operations like modifying data, executing complex logic, or handling transactions.

Returns: Does not return a value directly; instead, it can return multiple result sets using OUT parameters.

Usage: Called using the CALL statement.

Can Modify Data: Yes, can perform INSERT, UPDATE, and DELETE operations.

Can Be Used in SQL Queries: No, cannot be used inside SELECT statements.

Example:
sql
```
CREATE PROCEDURE GetEmployeeDetails(IN emp_id INT)
BEGIN
   SELECT * FROM employees WHERE id = emp_id;
END;
```

2. Function
Purpose: Used for calculations, data transformations, or returning a single value.

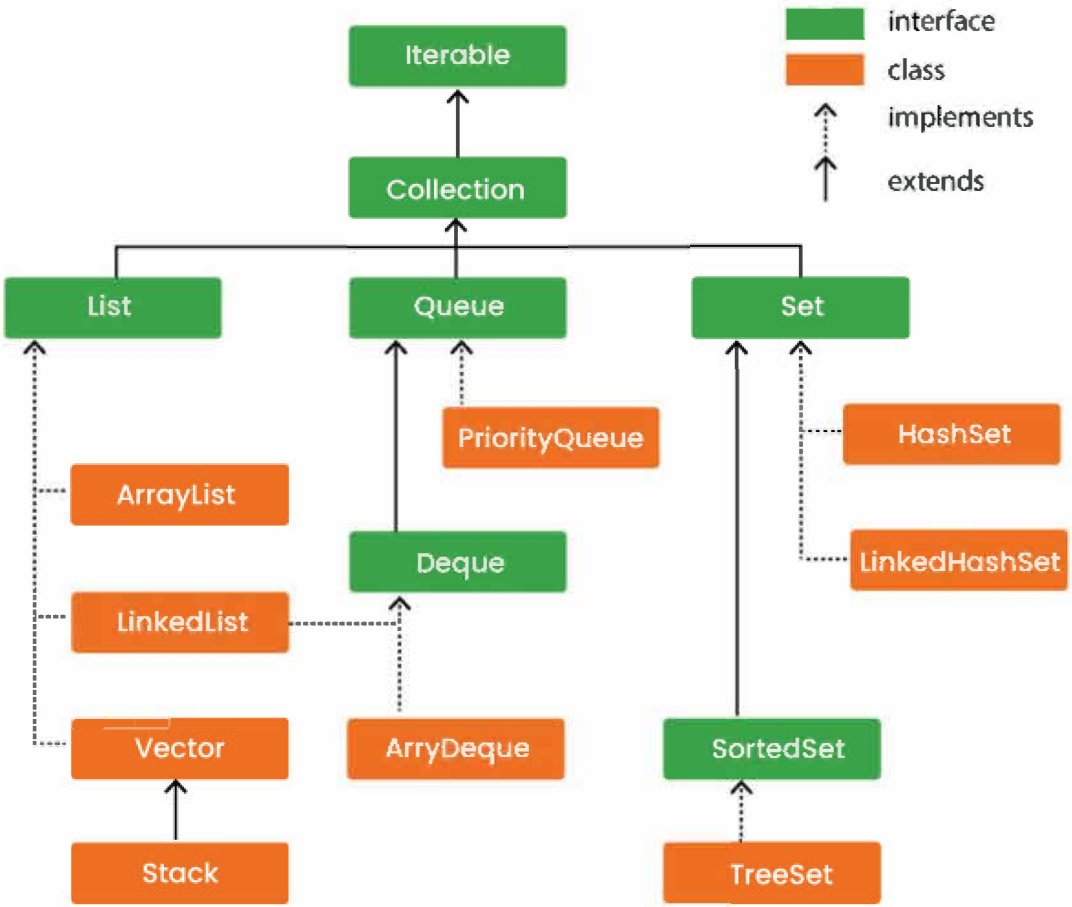
Returns: Must return a single value using the RETURN statement.

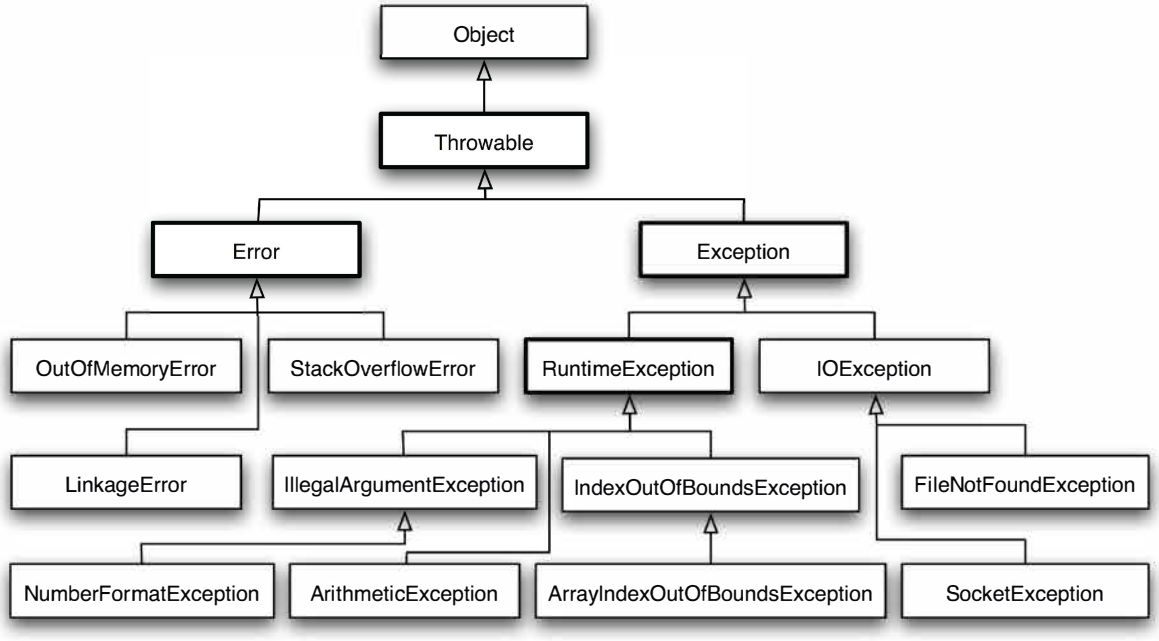
Usage: Called inside SQL expressions like SELECT, WHERE, or HAVING.

Can Modify Data: No, functions cannot perform INSERT, UPDATE, or DELETE.

Can Be Used in SQL Queries: Yes, can be used inside SELECT statements.

Example:
sql
```
CREATE FUNCTION GetBonus(salary DECIMAL) RETURNS DECIMAL
BEGIN
   RETURN salary * 0.10;
END;
```


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArryDeque
SortedSet
Stack
TreeSet


Object
Throwable
Error
Exception
OutOfMemoryError
StackOverflowError
RuntimeException
IOException
LinkageError
IllegalArgumentException
IndexOutOfBoundsException
FileNotFoundException
NumberFormatException
ArithmeticException
ArrayIndexOutOfBoundsException
SocketException

Indexing in MySQL

Indexing in MySQL is a technique used to speed up data retrieval operations by creating a data structure that allows the database to quickly locate rows without scanning the entire table.

1. What is an Index?

An index is similar to a book index, where you can quickly find a topic without reading every page.

It improves query performance by reducing the number of rows scanned.

MySQL automatically creates indexes for primary keys and unique constraints.

2. Types of Indexes in MySQL

Index Type	Description
Primary Index	Created automatically for the primary key, ensuring uniqueness.
Unique Index	Ensures all values in a column are unique, but allows multiple unique indexes.
Full-Text Index	Used for text-based searches, enabling fast retrieval of words or phrases.
Composite Index	Created on multiple columns, improving performance for multi-column queries.
Spatial Index	Used for geographic data types, optimizing location-based queries.

3. How MySQL Uses Indexes

Speeds up searches by reducing the number of rows scanned.
Optimizes joins by allowing efficient row lookups.
Improves sorting by avoiding full table scans.
Enhances filtering in WHERE clauses.

4. Creating an Index in MySQL

sql
```
CREATE INDEX idx_lastname ON Persons (LastName);
```
This creates an index named idx_lastname on the LastName column.

You can also create a composite index:

sql
```
CREATE INDEX idx_name ON Persons (LastName, FirstName);
```

5. Considerations When Using Indexes

Indexes improve read performance but slow down write operations (INSERT, UPDATE, DELETE).

Too many indexes can increase storage usage and decrease performance.

Use indexes strategically on frequently queried columns.