

- B) ALTER TABLESPACE
 - C) CREATE DATAFILE
 - D) ADD DATAFILE
- Answer: B) ALTER TABLESPACE
-

Would you like to continue with **Structured and Unstructured Data** next (the next topic in your syllabus flow)?



Introduction to SQL

Topic 11: Introduction to SQL

◆ Concept Explanation

1. Definition:

SQL (Structured Query Language) is a standard language used to communicate with relational databases — it allows you to create, retrieve, update, and delete data.

2. Purpose:

SQL provides a uniform interface to manage data stored in relational databases like MySQL, Oracle, PostgreSQL, etc.

3. Key Categories of SQL Commands:

- **DDL (Data Definition Language):** Defines database structure — `CREATE` , `ALTER` , `DROP` .
- **DML (Data Manipulation Language):** Manipulates data — `SELECT` , `INSERT` , `UPDATE` , `DELETE` .
- **DCL (Data Control Language):** Manages user permissions — `GRANT` , `REVOKE` .
- **TCL (Transaction Control Language):** Manages transactions — `COMMIT` , `ROLLBACK` , `SAVEPOINT` .

4. Basic SQL Syntax Example:

sql

```
SELECT name, salary FROM employees WHERE department = 'Sales';
```

5. Features:

- Simple and easy to learn
 - Declarative (you tell *what* to do, not *how*)
 - Works across multiple database systems
-



5 Quality MCQs

Q1. What does SQL stand for?

- A) Simple Query Language
- B) Structured Query Language
- C) Sequential Query Language
- D) Standard Query Logic

Answer: B) Structured Query Language

Q2. Which of the following is a **DML** command?

- A) CREATE
- B) ALTER
- C) SELECT
- D) GRANT

Answer: C) SELECT

Q3. Which SQL category is responsible for defining the structure of a database?

- A) DDL
- B) DML
- C) DCL
- D) TCL

Answer: A) DDL

Q4. The command `ROLLBACK` belongs to which SQL category?

- A) DML
- B) DDL
- C) DCL
- D) TCL

Answer: D) TCL

Q5. Which of the following statements about SQL is true?

- A) SQL is a procedural language.
- B) SQL allows both data definition and data manipulation.
- C) SQL cannot be used to control user access.
- D) SQL is only used in Oracle databases.

Answer: B) SQL allows both data definition and data manipulation.

Would you like to continue with **DDL Commands** next?



[SQL Commands Overview](#)

Topic 12: SQL Commands Overview

◆ Concept Explanation

1. Definition:

SQL Commands are instructions used to interact with a relational database — they define structures, manipulate data, control access, and manage transactions.

2. Main Categories of SQL Commands:

A) **DDL (Data Definition Language)** – defines the database schema

- `CREATE` → Create tables or databases

- ALTER → Modify existing tables
- DROP → Delete tables or databases
- TRUNCATE → Remove all data from a table but keep the structure

B) DML (Data Manipulation Language) – handles data inside tables

- SELECT → Retrieve data
- INSERT → Add new records
- UPDATE → Modify existing records
- DELETE → Remove records

C) DCL (Data Control Language) – controls user access

- GRANT → Give user permissions
- REVOKE → Remove user permissions

D) TCL (Transaction Control Language) – manages database transactions

- COMMIT → Save changes permanently
- ROLLBACK → Undo recent changes
- SAVEPOINT → Set a point within a transaction to roll back to

3. Example SQL Commands:

sql Copy code

```
CREATE TABLE students (id INT, name VARCHAR(50), age INT);
INSERT INTO students VALUES (1, 'Riya', 22);
UPDATE students SET age = 23 WHERE id = 1;
DELETE FROM students WHERE id = 1;
COMMIT;
```

4. Key Point:

- DDL changes the structure, not the data.
- DML changes the data, not the structure.
- DCL & TCL manage security and transaction integrity.

5 Quality MCQs

Q1. Which SQL command is used to create a new table?

- A) INSERT
- B) CREATE

C) UPDATE

D) GRANT

Answer: B) CREATE

Q2. Which command removes all data from a table but keeps the structure?

A) DELETE

B) DROP

C) TRUNCATE

D) REMOVE

Answer: C) TRUNCATE

Q3. The GRANT and REVOKE commands are part of which SQL category?

A) DML

B) DDL

C) DCL

D) TCL

Answer: C) DCL

Q4. Which SQL command ensures that changes made are saved permanently?

A) ROLLBACK

B) SAVEPOINT

C) COMMIT

D) GRANT

Answer: C) COMMIT

Q5. The command ALTER TABLE students ADD email VARCHAR(50); belongs to:

A) DML

B) DDL

C) TCL

D) DCL

Answer: B) DDL

Would you like to continue with **DDL Commands** next?



DDL (Create, Alter, Drop, Grant, Revoke)

Topic 13: DDL Commands (CREATE, ALTER, DROP, GRANT, REVOKE)

◆ Concept Explanation

1. Definition:

DDL (Data Definition Language) commands are used to **define, modify, and manage database schema and structure** — they affect how data is stored, not the data itself.

2. Key DDL Commands:

- CREATE – Used to create new database objects like tables, views, or databases.

sql

Copy code

```
CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

- ALTER – Used to **modify an existing table** (add, delete, or change columns).

sql

Copy code

```
ALTER TABLE students ADD email VARCHAR(50);
ALTER TABLE students MODIFY age FLOAT;
ALTER TABLE students DROP COLUMN email;
```

- DROP – Used to **delete an entire table or database** permanently.

sql

 Copy code

```
DROP TABLE students;  
DROP DATABASE college;
```

- GRANT – Used to **assign privileges** (permissions) to users.

sql

 Copy code

```
GRANT SELECT, INSERT ON students TO user1;
```

- REVOKE – Used to **remove privileges** previously granted to users.

sql

 Copy code

```
REVOKE INSERT ON students FROM user1;
```

3. Key Notes:

- DDL commands are **auto-committed** — changes are permanent and cannot be rolled back.
- They change the **structure**, not the data.



5 Quality MCQs

Q1. Which of the following commands is used to **create a new table**?

- A) ALTER
- B) CREATE
- C) INSERT
- D) MODIFY



Answer: B) CREATE

Q2. What will happen if you execute `DROP TABLE students;` ?

- A) Only data will be deleted
 - B) Only structure will remain
 - C) Both structure and data will be deleted permanently
 - D) Table will be temporarily deleted
- Answer:** C) Both structure and data will be deleted permanently

Q3. Which command is used to **add a new column** to an existing table?

- A) ADD COLUMN
- B) CREATE COLUMN
- C) ALTER TABLE
- D) UPDATE TABLE

Answer: C) ALTER TABLE

Q4. The `GRANT` command in SQL is used for:

- A) Data insertion
- B) Giving user permissions
- C) Table creation
- D) Data backup

Answer: B) Giving user permissions

Q5. Which of the following statements about DDL commands is **true**?

- A) They can be rolled back using ROLLBACK command
- B) They modify data only
- C) They are auto-committed
- D) They are part of DML

Answer: C) They are auto-committed

Would you like to continue with **DML Commands (SELECT, INSERT, UPDATE, DELETE, TRUNCATE)** next?



DML (Select, Insert, Update, Delete, Truncate)

Topic 14: DML Commands (SELECT, INSERT, UPDATE, DELETE, TRUNCATE)

◆ Concept Explanation

1. Definition:

DML (Data Manipulation Language) commands are used to **manage and manipulate the data** stored inside database tables.

They work on the records (rows) rather than on the schema (structure).

2. Key DML Commands:

- SELECT → Retrieve data from one or more tables.

sql

 Copy code

```
SELECT name, age FROM students WHERE age > 20;
```

- INSERT → Add new records to a table.

sql

 Copy code

```
INSERT INTO students (id, name, age) VALUES (1, 'Riya', 22);
```

- UPDATE → Modify existing records in a table.

sql

 Copy code

```
UPDATE students SET age = 23 WHERE id = 1;
```

- DELETE → Remove records from a table (structure remains).

sql

 Copy code

```
DELETE FROM students WHERE id = 1;
```

- TRUNCATE → Remove all rows from a table, but keep its structure (faster than DELETE).

sql

 Copy code

```
TRUNCATE TABLE students;
```

3. Key Differences Between DELETE and TRUNCATE :

Feature	DELETE	TRUNCATE
Removes specific rows	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> No
Can use WHERE clause	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> No
Rollback possible	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> No (auto-commit)
Speed	Slower	Faster

4. Note:

- DML changes **data**, not structure.
 - Changes made by DML commands are **not permanent** until a `COMMIT` is issued.
-



5 Quality MCQs

Q1. Which of the following SQL statements retrieves all data from the `employees` table?

- A) SHOW * FROM employees;
- B) SELECT * FROM employees;
- C) FETCH * FROM employees;
- D) VIEW * FROM employees;

Answer: B) SELECT * FROM employees;

Q2. Which SQL command is used to **add a new record** to a table?

- A) INSERT
- B) UPDATE
- C) ADD
- D) APPEND

Answer: A) INSERT

Q3. Which of the following statements is **true** about the `TRUNCATE` command?

- A) It removes the structure of a table
- B) It can be rolled back
- C) It removes all rows and is faster than `DELETE`

D) It only removes specific rows

Answer: C) It removes all rows and is faster than DELETE

Q4. The `UPDATE` statement in SQL is used to:

- A) Change the structure of the table
- B) Modify existing records
- C) Remove all data
- D) Retrieve data from multiple tables

Answer: B) Modify existing records

Q5. Which of the following commands cannot use the `WHERE` clause?

- A) `DELETE`
- B) `UPDATE`
- C) `SELECT`
- D) `TRUNCATE`

Answer: D) `TRUNCATE`

Would you like to continue with **DCL Commands (COMMIT, ROLLBACK, SAVEPOINT)** next?



DCL (Commit, Rollback, Savepoint)

Here's your **concept explanation + 5 quality MCQs for DCL (Commit, Rollback, Savepoint)** — formatted in your preferred "Big Data prep" style 

◆ **Topic: DCL (Commit, Rollback, Savepoint)**

Concept Explanation (in 4–5 lines)

1. **DCL (Data Control Language)** commands manage **transactions** in SQL and control the state of data changes.
 2. **COMMIT** saves all changes made during the transaction permanently to the database.
 3. **ROLLBACK** undoes changes made in the current transaction before a COMMIT is issued.
 4. **SAVEPOINT** sets a point within a transaction to which you can roll back without affecting prior operations.
 5. These commands ensure **data integrity and recovery** in case of errors during transactions.
-

5 Quality MCQs

Q1. Which of the following SQL commands permanently saves all changes made in the current transaction?

- A) ROLLBACK
- B) SAVEPOINT
- C) COMMIT
- D) TRUNCATE

Answer: C) COMMIT

Q2. Which command is used to undo part of a transaction up to a specific point?

- A) ROLLBACK TO SAVEPOINT
- B) COMMIT
- C) DELETE
- D) REVOKE

Answer: A) ROLLBACK TO SAVEPOINT

Q3. What happens if a COMMIT is issued after a ROLLBACK?

- A) The ROLLBACK is canceled
- B) The transaction restarts automatically
- C) Nothing happens; ROLLBACK already undid changes

D) The COMMIT reverts ROLLBACK

Answer: C) Nothing happens; ROLLBACK already undid changes

Q4. Which command will release all SAVEPOINTS in a transaction?

- A) ROLLBACK TO SAVEPOINT
- B) COMMIT
- C) DELETE SAVEPOINTS
- D) TRUNCATE

Answer: B) COMMIT

Q5. If an error occurs during a transaction and no COMMIT is issued, what will happen when you disconnect from the database?

- A) Changes will be auto-saved
- B) Changes will be rolled back
- C) Changes will be committed automatically
- D) The transaction remains pending

Answer: B) Changes will be rolled back

Would you like me to move to **TCL (Transaction Control Language)** next (it includes these commands in a broader context)?



Grouping and Aggregation (GROUP BY, HAVING)

Here's your **concept explanation + 5 quality MCQs for Grouping and Aggregation (GROUP BY, HAVING)** — formatted exactly in your preferred "Big Data prep" style 

◆ **Topic: Grouping and Aggregation (GROUP BY, HAVING)**

Concept Explanation (in 4–5 lines)

1. **GROUP BY** is used in SQL to **group rows** that have the same values in specified columns into summary rows.
 2. It is often used with **aggregate functions** such as `COUNT()`, `SUM()`, `AVG()`, `MIN()`, and `MAX()` to perform calculations on each group.
 3. **HAVING** is similar to **WHERE**, but it filters **groups after aggregation**, whereas **WHERE** filters **rows before aggregation**.
 4. **GROUP BY** must appear **after the WHERE clause and before the ORDER BY clause** in an SQL query.
 5. These clauses are useful for **data summarization and analytics queries**.
-



5 Quality MCQs

Q1. Which of the following SQL clauses is used to group rows that have the same values?

- A) ORDER BY
- B) GROUP BY
- C) HAVING
- D) DISTINCT



Answer: B) GROUP BY

Q2. Which statement correctly describes the difference between WHERE and HAVING?

- A) WHERE filters groups; HAVING filters rows
- B) HAVING filters rows; WHERE filters groups
- C) WHERE filters rows; HAVING filters groups
- D) Both filter rows only



Answer: C) WHERE filters rows; HAVING filters groups

Q3. Which query will display the total salary of employees in each department?

- A) `SELECT dept_id, salary FROM employees;`
- B) `SELECT dept_id, SUM(salary) FROM employees GROUP BY dept_id;`
- C) `SELECT dept_id, salary FROM employees GROUP BY dept_id;`

D) `SELECT SUM(salary) FROM employees;`

Answer: B) `SELECT dept_id, SUM(salary) FROM employees GROUP BY dept_id;`

Q4. Which clause is used to restrict the output of a GROUP BY query based on aggregate values?

- A) WHERE
- B) ORDER BY
- C) HAVING
- D) DISTINCT

Answer: C) HAVING

Q5. What will happen if a column is listed in the SELECT clause but not included in the GROUP BY clause or an aggregate function?

- A) SQL executes successfully
- B) SQL ignores that column
- C) SQL returns a syntax error
- D) SQL automatically aggregates that column

Answer: C) SQL returns a syntax error

Would you like me to continue with **JOINS (INNER, LEFT, RIGHT, FULL)** next?



Sorting Data (ORDER BY)

Here's your **concept explanation + 5 quality MCQs** for **Sorting Data (ORDER BY)** in your CDAC CCEE prep format

◆ **Topic: Sorting Data (ORDER BY)**

Concept Explanation (in 4–5 lines)

1. ORDER BY is used in SQL to sort the result set of a query in either ascending (ASC) or descending (DESC) order.
 2. By default, ORDER BY sorts in ascending order if no direction is specified.
 3. It can sort based on one or more columns or even expressions.
 4. Sorting happens after the WHERE, GROUP BY, and HAVING clauses are applied.
 5. It helps in organizing query output for easy analysis and readability.
-



5 Quality MCQs

Q1. Which SQL clause is used to arrange the output rows in ascending or descending order?

- A) SORT
- B) ORDER BY
- C) GROUP BY
- D) FILTER BY



Answer: B) ORDER BY

Q2. What is the default sorting order used by ORDER BY clause in SQL?

- A) Descending
- B) Random
- C) Ascending
- D) Based on primary key



Answer: C) Ascending

Q3. Which of the following queries correctly displays employees ordered by salary from highest to lowest?

- A) SELECT * FROM employees ORDER BY salary;
- B) SELECT * FROM employees ORDER BY salary ASC;
- C) SELECT * FROM employees ORDER BY salary DESC;
- D) SELECT * FROM employees SORT salary DESC;



Answer: C) SELECT * FROM employees ORDER BY salary DESC;

Q4. You want to sort employees first by department (ascending) and then by salary (descending). Which query is correct?

- A) `SELECT * FROM employees ORDER BY dept_id, salary DESC;`
- B) `SELECT * FROM employees ORDER BY dept_id ASC salary DESC;`
- C) `SELECT * FROM employees SORT BY dept_id, salary DESC;`
- D) `SELECT * FROM employees ORDER dept_id, salary;`

Answer: A) `SELECT * FROM employees ORDER BY dept_id, salary DESC;`

Q5. Which statement is true about ORDER BY?

- A) It can only sort numeric columns.
- B) It must always be used with GROUP BY.
- C) It can sort using multiple columns.
- D) It cannot sort NULL values.

Answer: C) It can sort using multiple columns.

Would you like to move to the next SQL topic — **JOINS (INNER, LEFT, RIGHT, FULL)**?



Advanced Subqueries (Correlated, Nested)

Here's your **concept explanation + 5 quality MCQs for Advanced Subqueries (Correlated, Nested)** in your CDAC CCEE prep format

- ◆ **Topic: Advanced Subqueries (Correlated, Nested)**

Concept Explanation (in 4–5 lines)

1. A **subquery** is a query inside another SQL query, usually enclosed in parentheses.

2. A nested subquery is executed once first, and its result is passed to the main (outer) query.
 3. A correlated subquery is executed repeatedly for each row of the outer query because it depends on values from that row.
 4. Subqueries can appear in clauses like WHERE, FROM, or SELECT.
 5. They're powerful for comparisons, filtering, and conditional selections in complex queries.
-



5 Quality MCQs

Q1. Which of the following best defines a *nested subquery*?

- A) A query that executes for each row of the outer query.
- B) A query that runs independently and passes its result to the outer query.
- C) A query that modifies the main query's data.
- D) A query that must include a JOIN.

Answer: B) A query that runs independently and passes its result to the outer query.

Q2. What differentiates a *correlated subquery* from a normal subquery?

- A) It is always faster.
- B) It depends on values from the outer query.
- C) It can only be used in the SELECT clause.
- D) It always returns multiple rows.

Answer: B) It depends on values from the outer query.

Q3. Which of the following SQL examples represents a correlated subquery?

A)

sql

```
SELECT name FROM students  
WHERE marks > (SELECT AVG(marks) FROM students);
```

B)

sql

```
SELECT name FROM employees e  
WHERE salary > (SELECT AVG(salary) FROM employees d WHERE d.dept_id = e.dept_id);
```

C)

sql

Copy code

```
SELECT * FROM employees WHERE emp_id IN (SELECT emp_id FROM bonus);
```

D) None of the above

Answer: B) — because it references e.dept_id from the outer query.

Q4. Which of the following clauses can contain a subquery?

- A) SELECT
 - B) WHERE
 - C) FROM
 - D) All of the above
- Answer: D) All of the above
-

Q5. Which statement about correlated subqueries is *false*?

- A) They execute once for each row of the outer query.
 - B) They are generally slower than non-correlated subqueries.
 - C) They cannot reference columns from the outer query.
 - D) They are useful for row-wise comparisons.
- Answer: C) They cannot reference columns from the outer query.
-

Would you like to continue with **Joins (INNER, LEFT, RIGHT, FULL)** next or move to **Stored Functions and Procedures in SQL**?



Joins (Inner, Outer, Self Joins)

Here's your **concept explanation + 5 quality MCQs for Joins (Inner, Outer, Self Joins)** — aligned with CDAC CCEE-style prep 

◆ Topic: Joins (Inner, Outer, Self Joins)

Concept Explanation (in 4–5 lines)

1. **Join** combines rows from two or more tables based on a related column between them.
 2. **INNER JOIN** returns only matching rows from both tables.
 3. **OUTER JOIN** returns matched rows plus unmatched rows (LEFT, RIGHT, or FULL).
 4. **SELF JOIN** is when a table is joined with itself, often using aliases.
 5. Joins help in retrieving data spread across multiple relational tables efficiently.
-



5 Quality MCQs

Q1. Which SQL JOIN returns only those records that have matching values in both tables?

- A) LEFT JOIN
- B) RIGHT JOIN
- C) INNER JOIN
- D) FULL OUTER JOIN



Answer: C) INNER JOIN

Q2. What does a **LEFT OUTER JOIN** return?

- A) All rows from both tables
- B) Only rows that match from both tables
- C) All rows from the left table and matching rows from the right table
- D) Only rows from the right table



Answer: C) All rows from the left table and matching rows from the right table

Q3. Which of the following SQL queries demonstrates a **SELF JOIN**?

A)

sql

 Copy code

```
SELECT e1.name, e2.name
FROM employees e1
INNER JOIN employees e2
ON e1.manager_id = e2.emp_id;
```

B)

sql

 Copy code

```
SELECT e.name, d.name
FROM employees e
JOIN departments d
ON e.dept_id = d.id;
```

C)

sql

 Copy code

```
SELECT * FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

D) None of these

Answer: A) — The same table `employees` is joined with itself using aliases.

Q4. Which of the following statements about **FULL OUTER JOIN** is *true*?

- A) It returns only matching rows from both tables.
- B) It returns unmatched rows from both tables along with matched ones.
- C) It returns unmatched rows only from the left table.
- D) It is the same as an INNER JOIN.

Answer: B) It returns unmatched rows from both tables along with matched ones.

Q5. What will happen if the join condition is omitted in an INNER JOIN?

- A) No rows will be returned.
- B) A cross join (Cartesian product) will occur.
- C) Only NULL rows will be displayed.
- D) SQL will throw an error.

Answer: B) A cross join (Cartesian product) will occur.

Would you like to move next to **Stored Functions and Procedures in SQL**, or do you want **Set Operations (UNION, INTERSECT, MINUS)** next?



Joins (INNER, LEFT, RIGHT, FULL)

Here's your **concept explanation + 5 quality MCQs for Joins (INNER, LEFT, RIGHT, FULL)** — formatted for your CDAC CCEE preparation 

◆ Topic: Joins (INNER, LEFT, RIGHT, FULL)

Concept Explanation (in 4–5 lines)

1. **Joins** combine rows from two or more tables based on a related column (usually a foreign key).
 2. **INNER JOIN** returns rows with matching values in both tables.
 3. **LEFT JOIN** returns all rows from the left table and matching rows from the right table; unmatched right-side rows become `NULL`.
 4. **RIGHT JOIN** returns all rows from the right table and matching rows from the left table; unmatched left-side rows become `NULL`.
 5. **FULL JOIN (FULL OUTER JOIN)** returns all rows when there's a match in one of the tables — unmatched rows on both sides are filled with `NULL`.
-



5 Quality MCQs

Q1. Which type of join returns only the rows that have matching values in both tables?

- A) LEFT JOIN
- B) RIGHT JOIN
- C) INNER JOIN
- D) FULL JOIN

Answer: C) INNER JOIN

Q2. What will a **LEFT JOIN** return if there is no matching record in the right table?

- A) Only matching rows
- B) All rows from the right table
- C) All rows from the left table with NULLs for right table columns
- D) No rows

Answer: C) All rows from the left table with NULLs for right table columns

Q3. Which SQL JOIN is used to return all records from both tables, whether or not there is a match?

- A) FULL JOIN
- B) LEFT JOIN
- C) INNER JOIN
- D) CROSS JOIN

Answer: A) FULL JOIN

Q4. Which of the following SQL queries correctly performs a **RIGHT JOIN**?

A)

sql

Copy code

```
SELECT e.name, d.dept_name
FROM employees e
RIGHT JOIN departments d
ON e.dept_id = d.dept_id;
```

B)

sql

Copy code

```
SELECT e.name, d.dept_name
FROM employees e
LEFT JOIN departments d
ON e.dept_id = d.dept_id;
```

C)

sql

Copy code

```
SELECT * FROM employees, departments;
```

D) None of these

Answer: A) — because it returns all rows from the right table (departments).

Q5. When performing a **FULL OUTER JOIN**, how are non-matching rows handled?

- A) They are ignored completely.
- B) They are included with NULL values in columns of the missing table.
- C) They are deleted automatically.
- D) SQL raises an error.

Answer: B) They are included with NULL values in columns of the missing table.

Would you like to continue with **Stored Functions and Procedures in SQL** next, or go for **Views and Triggers**?



Conditional Constructs (CASE statements, IF logic)

Here's your concept explanation + 5 quality MCQs for **Conditional Constructs (CASE statements, IF logic)** — in perfect CDAC CCEE prep format

◆ Topic: Conditional Constructs (CASE statements, IF logic)

Concept Explanation (in 4–5 lines)

1. **Conditional constructs** in SQL allow performing decision-making within queries.
 2. The **CASE statement** works like an `if-else` ladder — it checks conditions sequentially and returns a value when the first condition is true.
 3. The **IF() function** (supported in MySQL) returns one value if a condition is true and another if it's false.
 4. These constructs are mainly used in the **SELECT, UPDATE, or ORDER BY clauses** for conditional output.
 5. They help create logic-based computed columns or conditional transformations in SQL queries.
-



5 Quality MCQs

Q1. Which SQL statement is used for conditional logic similar to "if-else" in programming languages?

- A) SWITCH
- B) CASE
- C) CHOOSE
- D) SELECT



Answer: B) CASE

Q2. What will be the result of this query?

sql

Copy code

```
SELECT name,
CASE
    WHEN marks >= 90 THEN 'A'
    WHEN marks >= 75 THEN 'B'
    ELSE 'C'
END AS grade
FROM students;
```

- A) Assigns grades based on marks
 - B) Shows only students with grade 'A'
 - C) Produces an error
 - D) Returns NULL for all rows
- Answer:** A) Assigns grades based on marks
-

Q3. Which of the following SQL functions performs a simple conditional check returning one value if true and another if false?

- A) CHOOSE()
- B) IF()
- C) CASE()
- D) IIF()

Answer: B) IF()

Q4. What will be the output of this query?

sql

 Copy code

```
SELECT salary,  
CASE  
    WHEN salary > 50000 THEN 'High'  
    WHEN salary BETWEEN 30000 AND 50000 THEN 'Medium'  
    ELSE 'Low'  
END AS Category  
FROM employees;
```

- A) Classifies salaries as High, Medium, or Low
 - B) Only displays High salaries
 - C) Raises syntax error
 - D) Returns only NULL values
- Answer:** A) Classifies salaries as High, Medium, or Low
-

Q5. Which clause can a **CASE** expression be used in?

- A) SELECT
- B) WHERE
- C) ORDER BY
- D) All of the above

Answer: D) All of the above

Would you like to move next to **Views or Stored Functions and Procedures in SQL?**



Views – Creation and Usage

Here's your **concept explanation + 5 quality MCQs** for **Views – Creation and Usage** — in your CDAC CCEE-style prep format 

◆ Topic: Views – Creation and Usage

Concept Explanation (in 4–5 lines)

1. A View is a **virtual table** created from one or more base tables using an SQL query.
 2. It **does not store data physically**; instead, it stores the SQL query that dynamically fetches data when accessed.
 3. Views are used to **simplify complex queries, enhance security, and restrict data access** to certain columns or rows.
 4. A view can be created using the `CREATE VIEW` command and deleted using `DROP VIEW`.
 5. Some views are **updatable** (if based on a single table), while **complex views** (with joins, group by) are **read-only**.
-



5 Quality MCQs

Q1. What is a view in SQL?

- A) A physical copy of a table
- B) A virtual table based on a query
- C) A backup of a table
- D) A table containing only metadata

Answer: B) A virtual table based on a query

Q2. Which of the following is the correct syntax to create a view?

A)

sql

 Copy code

```
CREATE VIEW view_name AS  
SELECT columns FROM table_name WHERE condition;
```

B)

sql

 Copy code

```
MAKE VIEW view_name FROM table_name;
```

C)

sql

 Copy code

```
VIEW CREATE view_name SELECT ...;
```

D)

sql

 Copy code

```
SELECT INTO view_name FROM table_name;
```

Answer: A) Correct syntax using CREATE VIEW .

Q3. Which of the following is true about SQL views?

- A) Views always store data permanently.

- B) Views can simplify complex queries by encapsulating logic.
- C) Views cannot be based on joins.
- D) Views cannot be dropped.

Answer: B) Views can simplify complex queries by encapsulating logic.

Q4. What will happen if you execute `DROP VIEW employee_view;` ?

- A) The base table will be deleted.
- B) Only the view definition will be deleted.
- C) The entire database will be dropped.
- D) It will raise an error.

Answer: B) Only the view definition will be deleted.

Q5. Which of the following statements is **false** about views?

- A) A view can restrict users from accessing specific data.
- B) A view can be created on multiple tables using joins.
- C) A view always allows insert and update operations.
- D) A view can be used just like a table in queries.

Answer: C) A view always allows insert and update operations.

Would you like to continue with **Triggers** next or go for **Stored Procedures and Functions in SQL**?



Window Functions

Here's your **concept explanation + 5 quality MCQs for Window Functions** — in perfect CDAC CCEE exam-prep format 

◆ Topic: Window Functions

Concept Explanation (in 4–5 lines)

1. **Window functions** perform calculations **across a set of rows related to the current row**, without collapsing them into a single output (unlike `GROUP BY`).
 2. They use the `OVER()` clause to define a “window” or range of rows to operate on.
 3. Common window functions include `ROW_NUMBER()` , `RANK()` , `DENSE_RANK()` , `LEAD()` , `LAG()` , `SUM()` , `AVG()` , etc.
 4. They are used for **ranking, running totals, moving averages, and comparisons between rows**.
 5. The `PARTITION BY` clause divides data into groups, and `ORDER BY` defines the sequence of rows in each group.
-



5 Quality MCQs

Q1. What distinguishes a **window function** from a regular aggregate function?

- A) It can be used only in `GROUP BY`.
- B) It collapses rows into a single result.
- C) It performs calculations without collapsing rows.
- D) It cannot use `ORDER BY`.



Answer: C) It performs calculations without collapsing rows.

Q2. What does the `OVER()` clause in a window function specify?

- A) The alias of the result column
- B) The condition for `WHERE` clause
- C) The set of rows (window) used for the calculation
- D) The join condition



Answer: C) The set of rows (window) used for the calculation

Q3. Which of the following assigns a unique sequential number to each row in a result set?

- A) `RANK()`
- B) `ROW_NUMBER()`
- C) `DENSE_RANK()`

D) COUNT()

Answer: B) ROW_NUMBER()

Q4. What is the difference between RANK() and DENSE_RANK() ?

- A) RANK() skips ranks after ties, while DENSE_RANK() does not.
- B) RANK() gives same rank to all, while DENSE_RANK() gives unique ranks.
- C) Both behave identically.
- D) DENSE_RANK() cannot be used with PARTITION BY.

Answer: A) RANK() skips ranks after ties, while DENSE_RANK() does not.

Q5. Which SQL statement correctly uses a window function?

A)

sql

```
SELECT name, salary,  
       RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS rnk  
    FROM employees;
```

B)

sql

```
SELECT name, salary,  
       RANK(salary) FROM employees;
```

C)

sql

```
SELECT RANK salary FROM employees;
```

D)

sql

```
SELECT salary, RANK OVER employees;
```

Answer: A) Correct use of RANK() with OVER() and PARTITION BY .

Would you like to move next to **Stored Procedures and Functions or Triggers?**



Window Functions in detailed

Here's a **detailed explanation of Window Functions in SQL**

1. What are Window Functions?

- **Window functions** perform calculations **across a set of rows related to the current row**, without collapsing them into a single output (unlike GROUP BY).
- They are used for **ranking, running totals, moving averages, and comparisons between rows**.

Syntax:

sql

Copy code

```
function_name(expression)
OVER (PARTITION BY column_name ORDER BY column_name)
```

2. Components of a Window Function

1. Function Name:

e.g., ROW_NUMBER(), RANK(), SUM(), AVG()

2. OVER() Clause:

Defines the **window** of rows the function operates on.

3. PARTITION BY:

Divides data into groups (like GROUP BY).

4. ORDER BY:

Defines the order of rows within each partition.

3. Common Types of Window Functions

a) Ranking Functions

Function	Description
ROW_NUMBER()	Assigns a unique number to each row in order.
RANK()	Gives rank with gaps for ties.
DENSE_RANK()	Gives rank without gaps for ties.
NTILE(n)	Divides rows into <code>n</code> equal buckets.

Example:

sql

 Copy code

```
SELECT emp_id, department, salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank_in_dept
FROM employees;
```

b) Aggregate Functions as Window Functions

Function	Description
SUM()	Cumulative or total value across window
AVG()	Average value across window
MIN() / MAX()	Minimum or maximum value

Example:

sql

 Copy code

```
SELECT emp_id, department, salary,  
       SUM(salary) OVER (PARTITION BY department ORDER BY emp_id) AS running_total  
FROM employees;
```

⌚ c) Value Functions

Function	Description
LAG(column, n)	Fetches value from n rows before current.
LEAD(column, n)	Fetches value from n rows after current.
FIRST_VALUE()	Returns the first value in window.
LAST_VALUE()	Returns the last value in window.

Example:

sql

Copy code

```
SELECT emp_id, department, salary,  
       LAG(salary, 1) OVER (PARTITION BY department ORDER BY salary) AS prev_salary,  
       LEAD(salary, 1) OVER (PARTITION BY department ORDER BY salary) AS next_salary  
FROM employees;
```

▀ 4. FRAME Clause (Optional)

Defines which rows are included in the window frame relative to the current row.

Example:

sql

Copy code

```
SUM(salary) OVER (  
    ORDER BY emp_id
```

```
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
```

```
) AS rolling_sum
```

👉 Calculates the sum of the current row and previous 2 rows.

5. Key Differences vs Aggregate Functions

Feature	Aggregate	Window Function
Output rows	Reduces multiple rows into one	Returns all rows
Usage	Uses GROUP BY	Uses OVER()
Example	SELECT dept, SUM(salary)	SUM(salary) OVER (PARTITION BY dept)

6. Real-Life Use Cases

- Ranking students by marks.
 - Calculating running totals of sales.
 - Comparing a month's sales with previous month.
 - Finding top-N salaries per department.
-

Would you like me to give 5 quality MCQs with answers on this topic next (for interview or test prep)?



Triggers – Concepts and Implementation

Here's a detailed explanation of Triggers – Concepts and Implementation 

⚙️ 1. What is a Trigger?

- A Trigger is a special stored program in a database that automatically executes (fires) when a specified event occurs on a table.
 - It is often used to enforce business rules, maintain audit trails, or validate data before or after changes.
-

✳️ 2. Types of Triggers

Based on the **timing** and **event**, triggers are categorized as:

◆ By Timing

Type	Description
BEFORE Trigger	Executes before the event (INSERT/UPDATE/DELETE). Used for validation.
AFTER Trigger	Executes after the event has occurred. Used for logging or auditing.

◆ By Event

Event	Description
INSERT Trigger	Fires when a new record is inserted.
UPDATE Trigger	Fires when an existing record is updated.
DELETE Trigger	Fires when a record is deleted.

🗣️ 3. Syntax

sql

Copy code

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
```

```
-- Trigger Logic  
END;
```

💡 4. Example: AFTER INSERT Trigger

Suppose we want to **log all employee insertions** into a separate table:

sql

 Copy code

```
CREATE TABLE employee_audit (  
    emp_id INT,  
    action_type VARCHAR(50),  
    action_time DATETIME  
);  
  
CREATE TRIGGER after_employee_insert  
AFTER INSERT ON employees  
FOR EACH ROW  
BEGIN  
    INSERT INTO employee_audit(emp_id, action_type, action_time)  
    VALUES (NEW.emp_id, 'INSERT', NOW());  
END;
```

✓ Explanation:

- NEW refers to the new record being inserted.
- Trigger runs automatically **after** a new employee is added.
- A log is created in `employee_audit`.

💻 5. Example: BEFORE UPDATE Trigger

Used to **validate or modify data** before it's updated.

sql

 Copy code

```
CREATE TRIGGER before_salary_update  
BEFORE UPDATE ON employees  
FOR EACH ROW  
BEGIN
```

```
IF NEW.salary < 0 THEN
    SET NEW.salary = 0;
END IF;
END;
```

Explanation:

- Ensures salary cannot be negative.
- The trigger modifies the new value before saving.



6. Accessing OLD and NEW Values

Keyword	Description
OLD	Refers to the value before the change.
NEW	Refers to the new value being inserted or updated.



7. Advantages of Triggers

- Automatically maintains **data integrity**.
- Useful for **auditing** and **security**.
- Reduces repetitive application-level code.



8. Disadvantages

- Can make debugging harder (executes automatically).
- May **impact performance** if too many triggers fire.
- Difficult to track hidden logic if overused.



9. Dropping a Trigger

sql

 Copy code

```
DROP TRIGGER trigger_name;
```

10. Real-Life Use Cases

- Audit trail of employee updates.
 - Auto-update stock quantities after a sale.
 - Prevent deletion of important data.
 - Validate inputs before database changes.
-

Would you like me to give 5 quality MCQs with answers on *Triggers* next for practice?



please explain in topic and also give 5 quality as you behaved above please dont misbehave

Perfect, Devansh

Here's the **topic-wise structured explanation + 5 quality MCQs on Triggers – Concepts and Implementation**, exactly in your preferred format.

Topic: Triggers – Concepts and Implementation

◆ 1. Concept of Triggers

- A **Trigger** is a **stored database object** that **automatically executes** (fires) when a specific **event** occurs on a table or view.
 - It helps automate actions like **validation, logging, or enforcing business rules**.
-

◆ 2. Types of Triggers

(-) By Timing

Type	Description
BEFORE Trigger	Executes before the triggering event (e.g., before inserting data).
AFTER Trigger	Executes after the event (e.g., after a record is deleted).

⚡ By Event

Event	When It Fires
INSERT	When a new row is inserted.
UPDATE	When a record is updated.
DELETE	When a record is deleted.

◆ 3. Syntax

sql

 Copy code

```

CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
    -- Trigger body
END;

```

◆ 4. Example 1 – AFTER INSERT Trigger

Used for auditing newly added records.

sql

 Copy code

```

CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN

```

```
INSERT INTO employee_audit(emp_id, action, time)
VALUES (NEW.emp_id, 'INSERTED', NOW());
END;
```

Here:

- NEW refers to the newly inserted row.
 - This trigger logs every new employee addition.
-

◆ 5. Example 2 – BEFORE UPDATE Trigger

Used for data validation.

sql

 Copy code

```
CREATE TRIGGER before_salary_update
BEFORE UPDATE ON employees
FOR EACH ROW
BEGIN
    IF NEW.salary < 0 THEN
        SET NEW.salary = 0;
    END IF;
END;
```

Ensures salary never goes below zero.

◆ 6. OLD and NEW Keywords

Keyword	Meaning
OLD	Refers to the existing data before the change.
NEW	Refers to the new data being added or updated.

Example:

sql

 Copy code

```
INSERT INTO audit_log VALUES (OLD.emp_id, NEW.salary);
```

◆ 7. Advantages

- Ensures **automatic data consistency**.
 - Useful for **security, validation, and auditing**.
 - Reduces need for manual or application-side logic.
-

◆ 8. Disadvantages

- Performance **overhead** when too many triggers fire.
 - **Hidden logic** — may make debugging difficult.
 - Complex triggers can cause **recursive updates**.
-

◆ 9. Dropping a Trigger

sql

 Copy code

```
DROP TRIGGER trigger_name;
```

💡 5 Quality MCQs on Triggers

Q1. Which of the following best describes a database trigger?

- A) A command to back up data manually
- B) A block of code that executes automatically when an event occurs
- C) A stored function that runs only on request
- D) A scheduled batch job

 **Answer:** B

 *Triggers execute automatically when an event (like INSERT, UPDATE, or DELETE) occurs.*

Q2. Which SQL keywords are used to define when a trigger executes relative to its event?

- A) START and END
- B) OPEN and CLOSE
- C) BEFORE and AFTER
- D) NEW and OLD

Answer: C

 *BEFORE and AFTER specify when the trigger should fire.*

Q3. What does the NEW keyword refer to in a trigger body?

- A) The old row before modification
- B) The new row after modification
- C) The database schema name
- D) The trigger name

Answer: B

 *NEW holds values of the new record being inserted or updated.*

Q4. Which of the following statements about triggers is FALSE?

- A) Triggers can enforce data integrity.
- B) Triggers can be set for specific events like INSERT or DELETE.
- C) Triggers can be manually executed by users.
- D) Triggers can automatically maintain audit logs.

Answer: C

 *Triggers are automatically executed, not manually invoked.*

Q5. Which of the following can cause performance issues in a database?

- A) Too many triggers firing on frequently updated tables
- B) Using BEFORE triggers for validation
- C) Using triggers for auditing
- D) Using OLD and NEW references