Here are some **basic SQL queries** for **DDL (Data Definition Language), DML (Data Manipulation Language), TCL (Transaction Control Language), and DCL (Data Control Language)**.

**1. DDL (Data Definition Language)**

DDL statements are used to **define and modify database structures** (tables, schemas, indexes, etc.).  
These statements include **CREATE, ALTER, DROP, TRUNCATE**.

**1.1 CREATE – Create a Table**

CREATE TABLE worker (

worker\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

first\_name VARCHAR2(25),

last\_name VARCHAR2(25),

salary NUMBER(15),

joining\_date DATE,

department VARCHAR2(25)

);

🔹 Creates a **worker** table.

**1.2 ALTER – Modify Table**

ALTER TABLE worker ADD email VARCHAR2(50);

🔹 Adds a **new column email** to the worker table.

ALTER TABLE worker MODIFY salary NUMBER(20);

🔹 Modifies the salary column to **increase its size**.

ALTER TABLE worker DROP COLUMN email;

🔹 **Removes** the email column.

**1.3 DROP – Delete a Table**

DROP TABLE worker;

🔹 **Deletes** the worker table **permanently**.

**1.4 TRUNCATE – Remove All Data (Faster)**

TRUNCATE TABLE worker;

🔹 **Removes all records** from worker but **keeps the table structure**.

**2. DML (Data Manipulation Language)**

DML statements are used to **manipulate data** in tables. These include **INSERT, UPDATE, DELETE, SELECT**.

**2.1 INSERT – Add Data**

INSERT INTO worker (first\_name, last\_name, salary, joining\_date, department)

VALUES ('John', 'Doe', 50000, TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'IT');

🔹 Adds a new **worker** into the table.

**2.2 UPDATE – Modify Data**

UPDATE worker

SET salary = salary + 5000

WHERE department = 'IT';

🔹 Increases salary by **5000** for all employees in **IT**.

**2.3 DELETE – Remove Data**

DELETE FROM worker WHERE department = 'HR';

🔹 Deletes all **HR department** employees.

**2.4 SELECT – Retrieve Data**

SELECT \* FROM worker;

🔹 Retrieves **all records** from worker.

SELECT first\_name, salary FROM worker WHERE salary > 50000;

🔹 Retrieves **workers with salary > 50000**.

**3. TCL (Transaction Control Language)**

TCL manages transactions in SQL. These include **COMMIT, ROLLBACK, SAVEPOINT**.

**3.1 COMMIT – Save Changes**

INSERT INTO worker (first\_name, last\_name, salary, department)

VALUES ('Mike', 'Smith', 60000, 'Finance');

COMMIT;

🔹 Saves the inserted **record permanently**.

**3.2 ROLLBACK – Undo Changes**

DELETE FROM worker WHERE department = 'IT';

ROLLBACK;

🔹 **Restores** deleted IT department employees.

**3.3 SAVEPOINT – Partial Rollback**

SAVEPOINT before\_delete;

DELETE FROM worker WHERE department = 'HR';

ROLLBACK TO before\_delete;

🔹 Saves a **point in the transaction** and **rolls back** only to that point.

**4. DCL (Data Control Language)**

DCL statements **control database access** and include **GRANT and REVOKE**.

**4.1 GRANT – Provide Privileges**

GRANT SELECT, INSERT, UPDATE ON worker TO user1;

🔹 Grants **SELECT, INSERT, UPDATE** permissions on worker **to user1**.

**4.2 REVOKE – Remove Privileges**

REVOKE INSERT, UPDATE ON worker FROM user1;

🔹 Removes **INSERT, UPDATE** permissions from user1.

**Summary of SQL Queries**

| **Category** | **Command** | **Purpose** |
| --- | --- | --- |
| **DDL** | CREATE TABLE | Create a new table |
|  | ALTER TABLE | Modify a table structure |
|  | DROP TABLE | Delete a table permanently |
|  | TRUNCATE TABLE | Remove all data but keep structure |
| **DML** | INSERT INTO | Add new records |
|  | UPDATE | Modify existing records |
|  | DELETE | Remove records |
|  | SELECT | Retrieve records |
| **TCL** | COMMIT | Save changes permanently |
|  | ROLLBACK | Undo changes |
|  | SAVEPOINT | Create a rollback point |
| **DCL** | GRANT | Give access to users |
|  | REVOKE | Remove access from users |

🚀 **These queries will work in Oracle SQL!** Let me know if you need more explanations! 😊

Here are **basic SQL queries** for the worker table using different types of **SQL functions and operators** in **Oracle SQL**.

**1. Aggregate Functions (SUM, AVG, COUNT, MIN, MAX)**

-- Total salary of all workers

SELECT SUM(salary) AS total\_salary FROM worker;

-- Average salary of workers

SELECT AVG(salary) AS avg\_salary FROM worker;

-- Number of workers in the company

SELECT COUNT(\*) AS total\_workers FROM worker;

-- Minimum salary among workers

SELECT MIN(salary) AS min\_salary FROM worker;

-- Maximum salary among workers

SELECT MAX(salary) AS max\_salary FROM worker;

**2. String Functions (LOWER(), UPPER(), LENGTH(), SUBSTR(), INSTR())**

-- Convert first name to lowercase

SELECT LOWER(first\_name) FROM worker;

-- Convert last name to uppercase

SELECT UPPER(last\_name) FROM worker;

-- Find the length of first names

SELECT first\_name, LENGTH(first\_name) AS name\_length FROM worker;

-- Extract first 3 characters from first\_name

SELECT SUBSTR(first\_name, 1, 3) AS short\_name FROM worker;

-- Find position of 'a' in first\_name

SELECT first\_name, INSTR(first\_name, 'a') AS position\_of\_a FROM worker;

**3. Numeric Functions (ROUND(), CEIL(), FLOOR(), MOD())**

-- Round salary to nearest hundred

SELECT salary, ROUND(salary, -2) AS rounded\_salary FROM worker;

-- Get next highest integer value of salary

SELECT salary, CEIL(salary) AS ceil\_salary FROM worker;

-- Get next lowest integer value of salary

SELECT salary, FLOOR(salary) AS floor\_salary FROM worker;

-- Get remainder when salary is divided by 5000

SELECT salary, MOD(salary, 5000) AS remainder FROM worker;

**4. Arithmetic Operators (+, -, \*, /)**

-- Increase salary by 10%

SELECT worker\_id, first\_name, salary, salary \* 1.1 AS new\_salary FROM worker;

-- Decrease salary by 5000

SELECT worker\_id, first\_name, salary, salary - 5000 AS reduced\_salary FROM worker;

-- Calculate yearly salary

SELECT worker\_id, first\_name, salary, salary \* 12 AS yearly\_salary FROM worker;

-- Divide salary by 2

SELECT worker\_id, first\_name, salary, salary / 2 AS half\_salary FROM worker;

**5. Logical Operators (AND, OR, NOT)**

-- Workers who have salary > 50000 AND belong to 'IT' department

SELECT \* FROM worker WHERE salary > 50000 AND department = 'IT';

-- Workers who have salary < 50000 OR belong to 'HR' department

SELECT \* FROM worker WHERE salary < 50000 OR department = 'HR';

-- Workers who do NOT belong to 'Finance' department

SELECT \* FROM worker WHERE NOT department = 'Finance';

**6. Comparison Operators (=, !=, <, >, <=, >=)**

-- Workers with salary equal to 60000

SELECT \* FROM worker WHERE salary = 60000;

-- Workers with salary NOT equal to 60000

SELECT \* FROM worker WHERE salary != 60000;

-- Workers with salary greater than 50000

SELECT \* FROM worker WHERE salary > 50000;

-- Workers with salary less than or equal to 40000

SELECT \* FROM worker WHERE salary <= 40000;

**7. LIKE Operator (Pattern Matching)**

-- Workers whose first name starts with 'J'

SELECT \* FROM worker WHERE first\_name LIKE 'J%';

-- Workers whose last name ends with 'son'

SELECT \* FROM worker WHERE last\_name LIKE '%son';

-- Workers whose first name contains 'an'

SELECT \* FROM worker WHERE first\_name LIKE '%an%';

-- Workers whose first name has exactly 5 letters

SELECT \* FROM worker WHERE first\_name LIKE '\_\_\_\_\_';

**8. IN Operator (Multiple Values)**

-- Workers who belong to IT, HR, or Finance departments

SELECT \* FROM worker WHERE department IN ('IT', 'HR', 'Finance');

-- Workers whose salary is either 50000, 60000, or 70000

SELECT \* FROM worker WHERE salary IN (50000, 60000, 70000);

**9. BETWEEN Operator (Range Search)**

-- Workers with salary between 40000 and 70000

SELECT \* FROM worker WHERE salary BETWEEN 40000 AND 70000;

-- Workers who joined between '2021-01-01' and '2023-12-31'

SELECT \* FROM worker WHERE joining\_date BETWEEN TO\_DATE('2021-01-01', 'YYYY-MM-DD')

AND TO\_DATE('2023-12-31', 'YYYY-MM-DD');

**10. DISTINCT Operator (Remove Duplicates)**

-- Get unique department names

SELECT DISTINCT department FROM worker;

-- Get unique salaries

SELECT DISTINCT salary FROM worker;

These queries cover **all the requested SQL functions and operators** using the worker table. 🚀 Let me know if you need further explanations! 😊

and operators. Do you need more? 😊🚀

**SQL Date Functions with Examples**

SQL provides **date functions** to perform operations like retrieving the current date, extracting parts of a date, adding/subtracting days, and formatting dates.

**1. Getting the Current Date**

**SYSDATE (Oracle) / CURRENT\_DATE (Standard SQL)**

🔹 Returns the **current system date and time**.

SELECT SYSDATE FROM dual; -- Oracle

SELECT CURRENT\_DATE; -- Other SQL databases

📌 **Output Example:** 04-APR-2025

**2. Extracting Parts of a Date**

**EXTRACT()**

🔹 Extracts a **specific part (YEAR, MONTH, DAY, HOUR, etc.)** from a date.

SELECT EXTRACT(YEAR FROM SYSDATE) FROM dual; -- Extracts the Year

SELECT EXTRACT(MONTH FROM SYSDATE) FROM dual; -- Extracts the Month

SELECT EXTRACT(DAY FROM SYSDATE) FROM dual; -- Extracts the Day

📌 **Example Output:** 2025

**TO\_CHAR()**

🔹 Converts a date to a **string format** (YYYY-MM-DD, DD-MON-YYYY, etc.).

SELECT TO\_CHAR(SYSDATE, 'YYYY-MM-DD') FROM dual; -- Output: 2025-04-04

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY') FROM dual; -- Output: 04-APR-2025

SELECT TO\_CHAR(SYSDATE, 'HH24:MI:SS') FROM dual; -- Output: 14:30:15

**TO\_DATE()**

🔹 Converts a string into a **date format**.

SELECT TO\_DATE('2025-04-04', 'YYYY-MM-DD') FROM dual;

**3. Adding & Subtracting Dates**

**ADD\_MONTHS()**

🔹 Adds or subtracts **months** from a date.

SELECT ADD\_MONTHS(SYSDATE, 3) FROM dual; -- Adds 3 months

SELECT ADD\_MONTHS(SYSDATE, -2) FROM dual; -- Subtracts 2 months

**NEXT\_DAY()**

🔹 Finds the **next occurrence of a specific day** of the week.

SELECT NEXT\_DAY(SYSDATE, 'FRIDAY') FROM dual;

📌 **Example Output:** If today is Monday, this will return the date of the next **Friday**.

**LAST\_DAY()**

🔹 Returns the **last day of the month**.

SELECT LAST\_DAY(SYSDATE) FROM dual;

📌 **Example Output:** 30-APR-2025

**4. Date Differences**

**MONTHS\_BETWEEN()**

🔹 Returns the **difference between two dates in months**.

SELECT MONTHS\_BETWEEN(TO\_DATE('2025-12-01', 'YYYY-MM-DD'), SYSDATE) FROM dual;

📌 **Example Output:** 8.9 (8 months and some days)

**ROUND() and TRUNC() on Dates**

🔹 ROUND() rounds the date to the nearest day, month, or year.  
🔹 TRUNC() removes the time portion and rounds **down** to the nearest day, month, or year.

SELECT ROUND(SYSDATE, 'MONTH') FROM dual; -- Rounds to the nearest month

SELECT TRUNC(SYSDATE, 'MONTH') FROM dual; -- Truncates to the start of the month

📌 **Example Output:** 01-APR-2025

**5. Formatting Dates**

| **Format** | **Example** | **Output** |
| --- | --- | --- |
| YYYY | SELECT TO\_CHAR(SYSDATE, 'YYYY') FROM dual; | 2025 |
| MM | SELECT TO\_CHAR(SYSDATE, 'MM') FROM dual; | 04 |
| DD | SELECT TO\_CHAR(SYSDATE, 'DD') FROM dual; | 04 |
| DAY | SELECT TO\_CHAR(SYSDATE, 'DAY') FROM dual; | FRIDAY |
| DY | SELECT TO\_CHAR(SYSDATE, 'DY') FROM dual; | FRI |

**6. Finding Leap Year**

SELECT CASE WHEN MOD(EXTRACT(YEAR FROM SYSDATE), 4) = 0

AND MOD(EXTRACT(YEAR FROM SYSDATE), 100) <> 0

OR MOD(EXTRACT(YEAR FROM SYSDATE), 400) = 0

THEN 'Leap Year' ELSE 'Not a Leap Year'

END AS Leap\_Year\_Status

FROM dual;

📌 **Example Output:** 'Leap Year' or 'Not a Leap Year'

**✅ Final Summary**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| SYSDATE | Returns current date & time | SELECT SYSDATE FROM dual; |
| EXTRACT() | Extracts YEAR, MONTH, DAY | EXTRACT(YEAR FROM SYSDATE) |
| TO\_CHAR() | Formats date as string | TO\_CHAR(SYSDATE, 'YYYY-MM-DD') |
| TO\_DATE() | Converts string to date | TO\_DATE('2025-04-04', 'YYYY-MM-DD') |
| ADD\_MONTHS() | Adds months to date | ADD\_MONTHS(SYSDATE, 3) |
| NEXT\_DAY() | Finds next occurrence of a weekday | NEXT\_DAY(SYSDATE, 'FRIDAY') |
| LAST\_DAY() | Finds last day of the month | LAST\_DAY(SYSDATE) |
| MONTHS\_BETWEEN() | Finds months between two dates | MONTHS\_BETWEEN('2025-12-01', SYSDATE) |
| ROUND() | Rounds date to the nearest unit | ROUND(SYSDATE, 'MONTH') |
| TRUNC() | Truncates date to a unit | TRUNC(SYSDATE, 'MONTH') |

Would you like more queries? 🚀

Here are **SQL queries** using additional **string functions** and **numeric functions** in **Oracle SQL**.

**1. String Functions**

These functions help manipulate string values.

**1.1 ASCII() – Get ASCII value of a character**

SELECT first\_name, ASCII(SUBSTR(first\_name, 1, 1)) AS ascii\_value FROM worker;

🔹 Extracts the **ASCII value** of the **first character** in first\_name.

**1.2 REPLACE() – Replace part of a string**

SELECT first\_name, REPLACE(first\_name, 'a', '@') AS replaced\_name FROM worker;

🔹 Replaces **'a'** with **'@'** in first\_name.

**1.3 CONCAT() – Concatenate two strings**

SELECT CONCAT(first\_name, last\_name) AS full\_name FROM worker;

🔹 Joins first\_name and last\_name into **one string**.  
💡 Use || for **more than two** strings:

SELECT first\_name || ' ' || last\_name AS full\_name FROM worker;

**1.4 TRIM() – Remove spaces from both sides**

SELECT TRIM(' John Doe ') AS trimmed\_name FROM dual;

🔹 Removes **leading and trailing spaces**.

**1.5 LTRIM() – Remove leading spaces**

SELECT LTRIM(' John Doe') AS left\_trimmed FROM dual;

🔹 Removes **spaces from the left**.

**1.6 RTRIM() – Remove trailing spaces**

SELECT RTRIM('John Doe ') AS right\_trimmed FROM dual;

🔹 Removes **spaces from the right**.

**1.7 LPAD() – Left pad with a character**

SELECT LPAD(first\_name, 10, '\*') AS left\_padded FROM worker;

🔹 Pads first\_name on the **left** with \* until **length = 10**.

**1.8 RPAD() – Right pad with a character**

SELECT RPAD(first\_name, 10, '-') AS right\_padded FROM worker;

🔹 Pads first\_name on the **right** with - until **length = 10**.

**2. Numeric Functions**

These functions are used for **mathematical operations**.

**2.1 ABS() – Absolute value**

SELECT salary, ABS(salary - 50000) AS abs\_difference FROM worker;

🔹 Returns **absolute value** of salary difference from 50000.

**2.2 COS() – Cosine of a number**

SELECT COS(0) AS cosine\_zero FROM dual;

🔹 Returns **cos(0) = 1**.

**2.3 SIN() – Sine of a number**

SELECT SIN(90 \* (3.141592/180)) AS sine\_90 FROM dual;

🔹 Returns **sine of 90 degrees** (converted to radians).

**2.4 TAN() – Tangent of a number**

SELECT TAN(45 \* (3.141592/180)) AS tan\_45 FROM dual;

🔹 Returns **tan(45 degrees) = 1**.

**2.5 ACOS() – Arc cosine**

SELECT ACOS(1) AS acos\_one FROM dual;

🔹 Returns the **inverse cosine** of 1 (which is 0 radians).

**2.6 ASIN() – Arc sine**

SELECT ASIN(0) AS asin\_zero FROM dual;

🔹 Returns **inverse sine** of 0 (which is 0 radians).

**2.7 ATAN() – Arc tangent**

SELECT ATAN(1) AS atan\_one FROM dual;

🔹 Returns **inverse tangent** of 1 (which is π/4 radians).

**2.8 POWER() (or POW()) – Exponentiation**

SELECT POWER(2, 3) AS two\_cubed FROM dual;

🔹 Returns **2³ = 8**.

**2.9 EXP() – Exponential function**

SELECT EXP(1) AS exp\_one FROM dual;

🔹 Returns **e¹ (Euler’s number ≈ 2.718)**.

**2.10 SIGN() – Sign of a number**

SELECT salary, SIGN(salary - 50000) AS sign\_value FROM worker;

🔹 Returns:

* 1 if **positive**
* -1 if **negative**
* 0 if **zero**

**Summary of Queries**

| **Function** | **Purpose** |
| --- | --- |
| ASCII() | Get ASCII value of a character |
| REPLACE() | Replace part of a string |
| CONCAT() | Concatenate strings |
| TRIM() | Remove spaces from both sides |
| LTRIM() | Remove left spaces |
| RTRIM() | Remove right spaces |
| LPAD() | Left pad with characters |
| RPAD() | Right pad with characters |
| ABS() | Absolute value |
| COS(), SIN(), TAN() | Trigonometric functions |
| ACOS(), ASIN(), ATAN() | Inverse trigonometric functions |
| POWER() or POW() | Raise to a power |
| EXP() | Exponential function |
| SIGN() | Sign of a number |

These queries will work in **Oracle SQL**. Let me know if you need any modifications! 🚀

Here are some **viva questions** along with **detailed explanations** related to **SQL commands, functions, and operators**.

**1. SQL COMMANDS**

**DDL (Data Definition Language)**

**1. What is DDL, and why is it used?**  
🔹 **DDL (Data Definition Language)** is used to **define and modify** database structures. It includes commands like **CREATE, ALTER, DROP, and TRUNCATE**.

**2. What is the difference between DROP and TRUNCATE?**

| **Command** | **Description** |
| --- | --- |
| DROP TABLE table\_name; | **Deletes** the table **completely** (structure + data). |
| TRUNCATE TABLE table\_name; | **Removes all rows** from the table but **keeps structure**. |

**3. What is the purpose of the ALTER command?**  
🔹 The ALTER command is used to **modify an existing table**.

ALTER TABLE worker ADD email VARCHAR2(50);

🔹 Adds a **new column email** to the worker table.

**DML (Data Manipulation Language)**

**4. What are the main DML commands?**  
🔹 **DML (Data Manipulation Language)** is used to **manage data** inside tables:

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

**5. How does DELETE differ from TRUNCATE?**

| **Command** | **Effect** | **Rollback?** |
| --- | --- | --- |
| DELETE FROM worker WHERE department = 'HR'; | Removes specific records | ✅ Yes |
| TRUNCATE TABLE worker; | Removes **all records** | ❌ No |

**TCL (Transaction Control Language)**

**6. What is the purpose of COMMIT, ROLLBACK, and SAVEPOINT?**  
🔹 **TCL (Transaction Control Language)** is used for **managing transactions**.

| **Command** | **Purpose** |
| --- | --- |
| COMMIT; | Saves all changes permanently |
| ROLLBACK; | Reverts all uncommitted changes |
| SAVEPOINT sp1; | Creates a **rollback point** |

SAVEPOINT before\_delete;

DELETE FROM worker WHERE department = 'HR';

ROLLBACK TO before\_delete;

🔹 Rolls back only the **HR department deletion**, keeping other transactions.

**DCL (Data Control Language)**

**7. How is GRANT different from REVOKE?**

| **Command** | **Purpose** |
| --- | --- |
| GRANT SELECT, INSERT ON worker TO user1; | Grants **permissions** to user1 |
| REVOKE INSERT ON worker FROM user1; | Removes **INSERT permission** from user1 |

**2. FUNCTIONS IN SQL**

**AGGREGATE FUNCTIONS**

**8. What are Aggregate Functions? Give examples.**  
🔹 Aggregate functions perform calculations on multiple rows and return **a single value**.

SELECT AVG(salary), SUM(salary), COUNT(worker\_id) FROM worker;

| **Function** | **Purpose** |
| --- | --- |
| AVG(salary) | Returns **average salary** |
| SUM(salary) | Returns **total salary** |
| COUNT(worker\_id) | Counts **number of employees** |

**STRING FUNCTIONS**

**9. What does REPLACE() do in SQL?**  
🔹 REPLACE() replaces a **part of a string**.

SELECT REPLACE('Hello World', 'World', 'SQL') FROM dual;

🔹 Output: **"Hello SQL"**

**10. What is the use of LTRIM() and RTRIM()?**  
🔹 LTRIM() removes **leading spaces**  
🔹 RTRIM() removes **trailing spaces**

SELECT LTRIM(' Hello') FROM dual; -- Output: 'Hello'

SELECT RTRIM('Hello ') FROM dual; -- Output: 'Hello'

**NUMERIC FUNCTIONS**

**11. How does ABS() work?**  
🔹 Returns the **absolute value**.

SELECT ABS(-100) FROM dual; -- Output: 100

**12. What is POWER() in SQL?**  
🔹 Raises a number to a power.

SELECT POWER(2, 3) FROM dual; -- Output: 8

**DATE FUNCTIONS**

**13. How do you get the current date in Oracle SQL?**  
🔹 Use SYSDATE.

SELECT SYSDATE FROM dual;

🔹 Returns **today’s date**.

**14. How do you extract the year from a date?**

SELECT EXTRACT(YEAR FROM SYSDATE) FROM dual;

🔹 Returns **current year**.

**3. OPERATORS IN SQL**

**ARITHMETIC OPERATORS**

**15. What are arithmetic operators in SQL?**  
🔹 Arithmetic operators perform **mathematical calculations**.

| **Operator** | **Example** | **Result** |
| --- | --- | --- |
| + (Addition) | SELECT 10 + 5 FROM dual; | 15 |
| - (Subtraction) | SELECT 10 - 5 FROM dual; | 5 |
| \* (Multiplication) | SELECT 10 \* 5 FROM dual; | 50 |
| / (Division) | SELECT 10 / 5 FROM dual; | 2 |

**LOGICAL OPERATORS**

**16. What are logical operators in SQL?**  
🔹 Logical operators are used in **WHERE conditions**.

| **Operator** | **Example** |
| --- | --- |
| AND | SELECT \* FROM worker WHERE salary > 50000 AND department = 'IT'; |
| OR | SELECT \* FROM worker WHERE department = 'HR' OR salary > 70000; |
| NOT | SELECT \* FROM worker WHERE NOT department = 'HR'; |

**COMPARISON OPERATORS**

**17. What is the difference between = and <>?**

| **Operator** | **Example** | **Purpose** |
| --- | --- | --- |
| = | SELECT \* FROM worker WHERE department = 'IT'; | Matches **exact value** |
| <> | SELECT \* FROM worker WHERE department <> 'HR'; | Not equal |

**LIKE OPERATOR**

**18. What is the LIKE operator used for?**  
🔹 The LIKE operator is used for **pattern matching**.

SELECT \* FROM worker WHERE first\_name LIKE 'A%';

🔹 Finds **names starting with 'A'**.

| **Pattern** | **Matches** |
| --- | --- |
| 'A%' | Starts with 'A' |
| '%A' | Ends with 'A' |
| '%A%' | Contains 'A' |

**IN & BETWEEN OPERATORS**

**19. What is the difference between IN and BETWEEN?**

| **Operator** | **Example** | **Purpose** |
| --- | --- | --- |
| IN | SELECT \* FROM worker WHERE department IN ('IT', 'HR'); | Matches **multiple values** |
| BETWEEN | SELECT \* FROM worker WHERE salary BETWEEN 40000 AND 70000; | Matches **range of values** |

**Final Thoughts**

These viva questions cover **all important SQL commands, functions, and operators**. Let me know if you need more details! 🚀

**Viva Questions with Answers on SQL Commands and Functions**

**1️⃣ SQL Commands (DDL, DML, DCL, TCL)**

**Q1: What are the different types of SQL commands?**

**A:** SQL commands are categorized into:

* **DDL (Data Definition Language)**: CREATE, ALTER, DROP, TRUNCATE
* **DML (Data Manipulation Language)**: INSERT, UPDATE, DELETE, SELECT
* **DCL (Data Control Language)**: GRANT, REVOKE
* **TCL (Transaction Control Language)**: COMMIT, ROLLBACK, SAVEPOINT

**Q2: What is the difference between DELETE and TRUNCATE?**

**A:**

* DELETE: Removes records **one by one** and can be rolled back.
* TRUNCATE: Removes **all rows at once** and **cannot be rolled back**.

📌 **Example:**

DELETE FROM worker WHERE department = 'HR'; -- Removes specific records

TRUNCATE TABLE worker; -- Removes all records from the table

**Q3: What is the difference between COMMIT, ROLLBACK, and SAVEPOINT?**

**A:**

* COMMIT: Saves the transaction permanently.
* ROLLBACK: Reverts all uncommitted changes.
* SAVEPOINT: Creates a temporary point to which a transaction can be rolled back.

📌 **Example:**

INSERT INTO worker VALUES (101, 'John', 'Doe', 50000, SYSDATE, 'IT');

SAVEPOINT A;

UPDATE worker SET salary = 55000 WHERE worker\_id = 101;

ROLLBACK TO A; -- Reverts only the update

COMMIT; -- Saves changes permanently

**2️⃣ Aggregate Functions**

**Q4: What are aggregate functions? Name a few.**

**A:** Aggregate functions perform calculations on a set of values and return a **single value**. Examples:

* COUNT(): Counts records
* SUM(): Adds up values
* AVG(): Calculates average
* MAX(): Finds maximum value
* MIN(): Finds minimum value

📌 **Example:**

SELECT department, AVG(salary) FROM worker GROUP BY department;

**3️⃣ String Functions**

**Q5: What is the use of the CONCAT() function?**

**A:** It **joins two or more strings together**.  
📌 **Example:**

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name FROM worker;

**Q6: What is the difference between LTRIM() and RTRIM()?**

**A:**

* LTRIM(): Removes leading (left) spaces.
* RTRIM(): Removes trailing (right) spaces.

📌 **Example:**

SELECT LTRIM(' Hello') FROM dual; -- Output: 'Hello'

SELECT RTRIM('Hello ') FROM dual; -- Output: 'Hello'

**4️⃣ Numeric Functions**

**Q7: What are some numeric functions in SQL?**

**A:**

* ABS(x): Returns absolute value
* POW(x, y): Calculates x raised to the power y
* EXP(x): Returns e^x
* SIN(x), COS(x), TAN(x): Trigonometric functions

📌 **Example:**

SELECT ABS(-10), POW(2, 3), EXP(1) FROM dual;

**5️⃣ Date Functions**

**Q8: How do you find the difference between two dates in SQL?**

**A:** Use MONTHS\_BETWEEN() or subtract dates.  
📌 **Example:**

SELECT MONTHS\_BETWEEN(TO\_DATE('2025-12-01', 'YYYY-MM-DD'), SYSDATE) FROM dual;

**Q9: What is the purpose of LAST\_DAY()?**

**A:** It returns the **last day of the given month**.  
📌 **Example:**

SELECT LAST\_DAY(SYSDATE) FROM dual;

**6️⃣ Operators**

**Q10: What are SQL comparison operators?**

**A:** Comparison operators compare values and return a boolean result. Examples:

* = (equal to)
* <> or != (not equal to)
* <, >, <=, >= (less than, greater than, etc.)

📌 **Example:**

SELECT \* FROM worker WHERE salary > 50000;

**Q11: What is the difference between LIKE and IN operators?**

**A:**

* **LIKE**: Used for pattern matching with % and \_.
* **IN**: Used to match multiple values in a list.

📌 **Example:**

SELECT \* FROM worker WHERE first\_name LIKE 'J%'; -- Names starting with 'J'

SELECT \* FROM worker WHERE department IN ('IT', 'HR'); -- IT or HR department

**Q12: How does the BETWEEN operator work?**

**A:** The BETWEEN operator checks if a value **falls within a range** (inclusive).  
📌 **Example:**

SELECT \* FROM worker WHERE salary BETWEEN 40000 AND 80000;

**7️⃣ Miscellaneous SQL Viva Questions**

**Q13: What is the difference between PRIMARY KEY and UNIQUE?**

**A:**

* **PRIMARY KEY**: Uniquely identifies a row, **cannot be NULL**.
* **UNIQUE**: Ensures unique values but allows **one NULL**.

📌 **Example:**

CREATE TABLE employee (

emp\_id INT PRIMARY KEY,

email VARCHAR(50) UNIQUE

);

**Q14: What is the difference between HAVING and WHERE?**

**A:**

* WHERE: Filters **before** grouping.
* HAVING: Filters **after** grouping.

📌 **Example:**

SELECT department, AVG(salary) FROM worker GROUP BY department HAVING AVG(salary) > 50000;

**Q15: What is the difference between JOIN and UNION?**

**A:**

* JOIN: Combines columns from multiple tables based on a relationship.
* UNION: Combines rows from multiple queries.

📌 **Example:**

SELECT w.worker\_id, w.first\_name, b.bonus\_amount

FROM worker w INNER JOIN bonus b ON w.worker\_id = b.worker\_ref\_id;

**Q16: What is a foreign key, and why is it used?**

**A:** A **foreign key** establishes a relationship between two tables by referencing the primary key of another table.  
📌 **Example:**

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

worker\_id INT,

FOREIGN KEY (worker\_id) REFERENCES worker(worker\_id)

);

**Q17: How can you prevent SQL Injection?**

**A:** Use **prepared statements** and **parameterized queries** instead of direct input.  
📌 **Example (Safe Query):**

PREPARE stmt FROM 'SELECT \* FROM worker WHERE first\_name = ?';

These are **some of the most expected viva questions** related to SQL commands, functions,