1. Inner Join vs. Outer Join

- **Inner Join:** "It retrieves only the rows that have **matching values in both tables**. If a record doesn't have a match in the other table, it is excluded."
- **Outer Join:** "It retrieves all matching rows **plus** the non-matching rows from one or both tables. It ensures data isn't lost just because it doesn't have a partner in the other table."

2. Anomalies (Insertion, Deletion, Update)

- **What are they?** "These are problems that occur in poorly designed (un-normalized) databases when we try to modify data."
- **Insertion Anomaly:** "When we cannot insert a new record because we are missing some mandatory data. For example, not being able to add a new 'Student' because they haven't enrolled in a 'Course' yet."
- **Deletion Anomaly:** "When deleting a record accidentally causes us to lose other important information. For example, if we delete the last student in a course, and the course details are stored in the same row, we lose the course information entirely."
- **Update Anomaly:** "When data is stored in multiple places (redundancy), and we update it in one place but forget the others, leading to inconsistent data."

3. Difference between Procedure & Function

- **Return Value:** "The main difference is that a **Function MUST return a value**, whereas a **Procedure** may or may not return a value (it uses OUT parameters to return data)."
- **Usage:** "A Function can be called directly inside a SQL statement (like SELECT my_func() FROM dual), but a Procedure cannot; it must be executed specifically (using EXEC or inside a PL/SQL block)."

4. Difference between SQL & PL/SQL

- **SQL (Structured Query Language):** "It is a **declarative** language. We tell the database *what* data we want (using SELECT, INSERT), but not *how* to get it. It executes one command at a time."
- **PL/SQL (Procedural Language extension to SQL):** "It is a **procedural** language. It allows us to write code with logic—like loops, variables, IF conditions, and error handling—to process data in a step-by-step manner."

5. What is DISTINCT?

"The DISTINCT keyword is used in a SELECT statement to **remove duplicate rows** from the result set, ensuring that the output contains only unique values."

6. How many types of Joins are there?

"Generally, there are **four main categories**, but we can list more specific types:"

1. **Inner Join**
2. **Outer Joins** (Left Outer, Right Outer, Full Outer)
3. **Cross Join** (Cartesian product)

4. Self Join (Joining a table to itself)

(Note: Mentioning these 4-6 is usually sufficient for a viva).

7. Triggers vs. Procedures vs. Functions (In PL/SQL)

- **Procedure:** "A subprogram that performs a specific action. It is **called explicitly** by the user or code."
- **Function:** "A subprogram used to compute a value. It **must return a value** and is also **called explicitly**."
- **Trigger:** "A stored block of code that **fires automatically** (implicitly) in response to a specific event, like an INSERT, UPDATE, or DELETE. You cannot call a trigger manually; the database does it for you."

I. Core SQL Concepts & Constraints

- **Q: What is a Primary Key vs. a Unique Key?**
    - **Answer:** "Both ensure uniqueness. However, a **Primary Key** uniquely identifies a record and **cannot** contain NULL values (and there can be only one per table). A **Unique Key** also prevents duplicates but **can** accept NULL values (and you can have multiple unique keys in a table)."
- **Q: What is a Foreign Key?**
    - **Answer:** "It is a field in one table that links to the Primary Key of another table. It enforces **Referential Integrity**, ensuring you can't have an 'orphan' record (e.g., an Order for a Customer that doesn't exist)."
- **Q: What is NULL in SQL? Is it the same as zero or a blank space?**
    - **Answer:** "No, NULL is **not** zero or a blank space. NULL represents **unknown, missing, or inapplicable data**. You cannot compare it using = (equals); you must use IS NULL or IS NOT NULL."
- **Q: What is the difference between DELETE, TRUNCATE, and DROP?**
    - **Answer:**
        - **DELETE:** Removes rows one by one (can use a WHERE clause). It is slower and can be rolled back.
        - **TRUNCATE:** Removes *all* rows instantly by resetting the table. It is faster but generally cannot be rolled back.
        - **DROP:** Removes the **entire table structure** and data from the database completely.

II. Aggregate Functions & Clauses

- **Q: Can you use an aggregate function (like SUM or COUNT) in a WHERE clause?**

- **Answer:** "No, you cannot. WHERE filters rows *before* grouping. If you need to filter based on an aggregate result (e.g., 'Show customers with *Total Orders* > 5'), you must use the **HAVING** clause."

- **Q: What is the order of execution in an SQL query?**
  - **Answer:** "It generally follows this order: FROM & JOIN -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY."

III. PL/SQL Specifics

- **Q: What is a Cursor? What are the two types?**
  - **Answer:** "A Cursor is a pointer to a memory area (Context Area) that stores the result of a query.
    - **Implicit Cursor:** Created automatically by Oracle for single-row operations (like INSERT or SELECT INTO).
    - **Explicit Cursor:** Defined manually by the programmer for queries that return multiple rows, allowing us to loop through them."

- **Q: What are the attributes of a Cursor?**
  - **Answer:** "The main attributes are:
    - %FOUND: Returns TRUE if a record was fetched.
    - %NOTFOUND: Returns TRUE if no record was fetched (useful for exit conditions).
    - %ROWCOUNT: Returns the number of rows affected so far.
    - %ISOPEN: Returns TRUE if the cursor is open."

- **Q: What is an Exception in PL/SQL? Name a few pre-defined ones.**
  - **Answer:** "An Exception is an error that occurs during runtime. Common pre-defined exceptions include:
    - NO_DATA_FOUND: When a SELECT INTO returns no rows.
    - TOO_MANY_ROWS: When a SELECT INTO returns more than one row.
    - ZERO_DIVIDE: When you try to divide by zero."

- **Q: What is the difference between %TYPE and %ROWTYPE?**
  - **Answer:**
    - **%TYPE:** Declares a variable with the same data type as a *specific column* (e.g., v_name EMP.Ename%TYPE).
    - **%ROWTYPE:** Declares a record variable that matches the structure of an *entire row* in a table (e.g., v_emp_rec EMP%ROWTYPE).

IV. Advanced / Conceptual

- **Q: What is a View? Why use it?**
    - **Answer:** "A View is a **virtual table** based on the result-set of an SQL statement. It doesn't store data itself. We use it to **simplify complex queries** (save a big join as a view) and for **security** (restrict access to specific columns)."
- **Q: What is Normalization?**
    - **Answer:** "It is the process of organizing data to **reduce redundancy** (duplicates) and improve data integrity. Common forms are 1NF (atomic values), 2NF (no partial dependency), and 3NF (no transitive dependency)."
- **Q: What are ACID properties?**
    - **Answer:** "It stands for **Atomicity** (all or nothing), **Consistency** (data remains valid), **Isolation** (transactions don't interfere), and **Durability** (saved changes stay saved)."