Perfect ✅ — You want **medium-length viva answers (4–5 lines each)** for **all 10 DBMS SQL/PLSQL experiments** — short enough to revise, but detailed enough for viva or lab exam answers.

Below is the complete, balanced **Database Management Systems (DBMS) Viva Question Bank with 4–5 line answers per question**, arranged **Experiment-wise** 👇

---

## 🧪 Experiment 1: SQL – DDL, DML, DCL, and TCL Commands

**1. Difference between DDL, DML, DCL, and TCL commands:**
DDL (Data Definition Language) defines database structure — e.g., CREATE, ALTER.
DML (Data Manipulation Language) handles data modification — e.g., INSERT, UPDATE.
DCL (Data Control Language) manages permissions — GRANT, REVOKE.
TCL (Transaction Control Language) manages transactions — COMMIT, ROLLBACK, SAVEPOINT.

**2. Purpose of CREATE TABLE and ALTER TABLE:**
`CREATE TABLE` is used to define a new table with columns and data types.
`ALTER TABLE` modifies an existing table's structure, such as adding or deleting columns, or changing data types.
Both commands belong to DDL and affect schema definitions directly.

**3. Difference between DELETE, TRUNCATE, and DROP:**
DELETE removes specific rows using a WHERE condition and can be rolled back.
TRUNCATE removes all rows from a table permanently but keeps the structure.
DROP completely removes the table along with its structure from the database.

**4. COMMIT, ROLLBACK, and SAVEPOINT with examples:**
COMMIT permanently saves changes made during a transaction.
ROLLBACK undoes changes since the last commit or savepoint.
SAVEPOINT sets a point within a transaction to roll back partially.
Example: after inserting rows, you can `SAVEPOINT A`, then ROLLBACK TO A if needed.

**5. What are constraints? Types of constraints:**
Constraints are rules applied on columns to maintain data integrity and accuracy.
Types include Primary Key, Foreign Key, Unique, Check, Not Null, and Default.
They prevent invalid data entry and enforce relationships between tables.

**6. Add or remove a column from an existing table:**
We use `ALTER TABLE table_name ADD column_name datatype;` to add a column.
To remove, `ALTER TABLE table_name DROP COLUMN column_name;`.
It helps in modifying the structure of an existing table without dropping it.

**7. What are privileges? GRANT and REVOKE usage:**
Privileges define the rights given to users to perform database operations.
`GRANT` allows a user access, while `REVOKE` removes that access.
Example: `GRANT SELECT ON employee TO user1;` gives read access.

**8. Default values for columns if not specified:**
If a column has a DEFAULT constraint, that value is used.
Otherwise, it automatically stores NULL unless defined otherwise.
This ensures data entry even when some fields are not provided.

---

## Experiment 2: SQL Functions and Operators

**1. Single-row and multiple-row (aggregate) functions:**
Single-row functions operate on one row and return one result (e.g., UPPER, LENGTH).
Aggregate or group functions operate on multiple rows and return a single result (e.g., SUM, AVG).
They help perform data analysis over large datasets.

**2. Five string functions in SQL:**
Common string functions are:

- `UPPER(str)` – converts to uppercase.
- `LOWER(str)` – converts to lowercase.
- `LENGTH(str)` – returns number of characters.
- `SUBSTR(str, start, len)` – extracts substring.
- `CONCAT(a, b)` – joins two strings.

**3. Numeric and date functions with examples:**
Numeric functions include ROUND(), MOD(), and ABS() for number operations.
Date functions like SYSDATE, ADD_MONTHS(), and MONTHS_BETWEEN() handle date calculations.
Example: `ADD_MONTHS(SYSDATE, 2)` adds two months to today's date.

**4. Difference between COUNT(*) and COUNT(column_name):**
`COUNT(*)` counts all rows including those with NULL values.
`COUNT(column_name)` ignores rows where the column is NULL.
Hence, `COUNT(*)` is generally used for total record count.

**5. What is an operator? Types of operators:**
An operator is a symbol used to perform operations on data values.
Types include arithmetic (+, -), comparison (=, <, >), logical (AND, OR, NOT), and special (LIKE, BETWEEN).
Operators are used to form conditions in SQL queries.

**6. LIKE operator usage:**
The LIKE operator is used for pattern matching in SQL.
`%` matches any sequence of characters and _ matches one character.
Example: `SELECT * FROM emp WHERE name LIKE 'S%';` finds names starting with S.

**7. Use of DISTINCT keyword:**
DISTINCT removes duplicate records from query results.

It ensures only unique values are displayed.
Example: `SELECT DISTINCT dept FROM employee;.`

**8. Can aggregate functions be used with WHERE clause?**
No, aggregate functions cannot be used in WHERE.
We use HAVING after GROUP BY to filter groups based on aggregate conditions.
Example: `HAVING AVG(salary) > 50000;.`

---

## Experiment 3: JOINS and Subqueries

**1. What is a JOIN? Types of JOINs:**
JOIN combines data from two or more tables using a related column.
Types: INNER, LEFT, RIGHT, FULL, and CROSS JOIN.
They help in retrieving related information spread across multiple tables.

**2. INNER JOIN vs OUTER JOIN:**
INNER JOIN returns only matching records from both tables.
OUTER JOIN returns all records from one table and matching ones from the other.
It can be LEFT, RIGHT, or FULL depending on direction.

**3. CROSS JOIN:**
Produces the Cartesian product of two tables.
Each row of the first table joins with every row of the second.
It's rarely used unless all combinations are needed.

**4. Subquery vs JOIN:**
Subquery is a query inside another query; JOIN merges tables directly.
Subqueries are useful for complex filters, while JOINS are better for combining related data efficiently.

**5. Correlated subqueries:**
A correlated subquery uses a value from the outer query.
It executes once for each row of the outer query.
Example: Find employees earning above average salary of their department.

**6. Can subqueries be nested?**
Yes, multiple subqueries can be nested to any level.
Inner subqueries execute first and their results are used by the outer query.

**7. LEFT JOIN with no matching record:**
Returns all rows from the left table with NULLs in unmatched right columns.
Ensures no data loss from the left table.

**8. Aliases in JOINs:**
Aliases are short names for tables, improving readability.
Example: `SELECT s.name FROM student s JOIN course c ON s.cid = c.id;.`

---

## 🧪 Experiment 4: Views, Indexes, and Sequences

**1. What is a view? Difference from a table:**
A view is a virtual table created from a SQL query.
It doesn't store data physically but displays data from one or more tables.
Tables store actual data, while views provide customized access.

**2. Can DML be performed on views?**
Yes, if the view is simple (based on one table, no group functions).
For complex views, DML may not be allowed directly.

**3. Advantages of using views:**
Views provide security by hiding sensitive columns.
They simplify queries and help reuse complex SQL code easily.

**4. What is an index?**
An index improves the speed of data retrieval operations.
It creates a separate data structure to quickly locate records using keys.

**5. Types of indexes:**
Primary, Secondary, Unique, Clustered, and Non-clustered.
Choice depends on data structure and access patterns.

**6. What is a sequence?**
A sequence automatically generates unique numeric values, often for primary keys.
Example: `CREATE SEQUENCE seq1 START WITH 1 INCREMENT BY 1;`.

**7. Can we drop a view or index?**
Yes, using DDL commands:
`DROP VIEW viewname;` and `DROP INDEX indexname;`.

**8. What are materialized views?**
Unlike normal views, materialized views store actual query results.
They improve performance for large datasets and are periodically refreshed.

---

Would you like me to continue with **Experiments 5–10 (PL/SQL, Triggers, Cursors, Transactions, and Project)** in the same 4–5 line detailed format?

Perfect! 💪 Here's the continuation of the **DBMS Viva Q&A (Experiments 5–10)** with **4–5 line detailed answers** per question.

---

# 🧪 Experiment 5: PL/SQL Programming – Conditional and Looping Constructs

**1. What is PL/SQL? Difference from SQL:**
PL/SQL is Oracle's procedural extension to SQL that allows variables, loops, and control structures.
SQL is declarative and executes single queries, while PL/SQL can execute blocks of code with logic and error handling.

**2. Different types of loops in PL/SQL:**

- **Basic LOOP:** Executes until an EXIT condition is met.
- **WHILE LOOP:** Executes while a condition is true.
- **FOR LOOP:** Executes a fixed number of times.

**3. IF-THEN-ELSE and CASE statements usage:**
Used for decision-making.
IF-THEN-ELSE handles simple conditions.
CASE handles multiple conditions and works like a switch statement.

**4. Structure of a PL/SQL block:**
`DECLARE` section for variables,
`BEGIN` section for executable statements,
`EXCEPTION` section for handling errors,
`END;` to terminate the block.

**5. Variables and constants:**
Variables store changeable values; constants store fixed values.
They are declared in the DECLARE section with a data type and optionally an initial value.

**6. Data types in PL/SQL:**
Common types: NUMBER, VARCHAR2, CHAR, DATE, BOOLEAN.
They help define the kind of data a variable or column can hold.

**7. Using FOR, WHILE, and LOOP constructs:**
FOR loop iterates fixed times, WHILE iterates based on a condition, LOOP executes indefinitely until EXIT.
They allow repetitive execution of PL/SQL code.

**8. Significance of DBMS_OUTPUT.PUT_LINE():**
Used to display output on the console or debug information during PL/SQL block execution.
Helps in monitoring and testing programs.

---

## Experiment 6: Stored Procedures and Functions

**1. What is a stored procedure?**
A stored procedure is a named PL/SQL block stored in the database, executed on demand.
It can accept parameters and perform multiple SQL operations as a single unit.

**2. What is a function?**
A function is a PL/SQL block that performs a task and returns a single value.
It can be used in SQL statements like SELECT.

**3. Difference between procedure and function:**
Functions return a value and can be called in SQL queries.
Procedures may or may not return values and are executed using CALL or EXECUTE.

**4. IN, OUT, and INOUT parameters:**
IN passes values to the procedure, OUT returns values to the caller, INOUT does both.
They allow dynamic data exchange between caller and procedure.

**5. Calling a procedure from SQL*Plus:**
Use `EXEC proc_name(parameter);` or `CALL proc_name(parameter);`.
It executes the stored procedure stored in the database.

**6. Can a procedure return a value?**
Yes, using OUT parameters.
Functions are preferred if a direct return value is needed.

**7. Exception handling in procedures:**
Handled using the `EXCEPTION` block inside the procedure.
Prevents runtime errors from terminating execution abruptly.

**8. Advantages of stored procedures:**
Code reusability, reduced network traffic, better security, and faster execution.
They encapsulate logic centrally in the database.

---

## Experiment 7: Triggers

**1. What is a trigger?**
A trigger is a stored PL/SQL block that automatically executes when a specified DML event occurs on a table or view.

**2. Types of triggers:**
BEFORE, AFTER, INSTEAD OF, Row-level, Statement-level triggers.

**3. Difference between BEFORE and AFTER triggers:**
BEFORE triggers execute before the DML operation.
AFTER triggers execute after the operation is completed.

**4. Row-level trigger:**
Fires once for each row affected by a DML operation.
Used to validate or modify row-level data.

**5. Statement-level trigger:**
Fires once per SQL statement, regardless of how many rows are affected.
Used for operations that affect the whole table.

**6. Can a trigger call another trigger?**
Yes, but excessive chaining may lead to complexity or performance issues.

**7. Advantages and disadvantages of triggers:**
Advantages: Automatic enforcement of business rules, audit logging.
Disadvantages: Can slow performance and cause complexity if overused.

**8. Mutating table errors:**
Occurs when a row-level trigger tries to read or modify the table that fired the trigger.
Handled by using statement-level triggers or temporary storage structures.

---

## 🧪 Experiment 8: Cursors

**1. What is a cursor?**
A cursor is a pointer to the result set of a query in PL/SQL.
It allows row-by-row processing of query results.

**2. Why are cursors used?**
Used when SQL operations need to process individual rows rather than entire tables at once.

**3. Types of cursors:**
Implicit (auto-managed by Oracle) and Explicit (user-declared for custom queries).

**4. Steps in cursor processing:**
Declare → Open → Fetch → Process → Close.
Each step ensures proper handling of result sets.

**5. Difference between implicit and explicit cursors:**
Implicit cursors are automatically created for single-row queries.
Explicit cursors are user-defined for multi-row queries.

**6. Exception handling in cursors:**
Handled using the `EXCEPTION` block to catch errors during open, fetch, or close operations.

**7. %ROWCOUNT and %FOUND:**
%ROWCOUNT returns number of rows processed.
%FOUND returns TRUE if last fetch returned a row, otherwise FALSE.

**8. Can we update a record through a cursor?**
Yes, using `FOR UPDATE` clause and `WHERE CURRENT OF cursor_name`.
Allows row-level updates during cursor iteration.

**9. Cursor attributes:**
`%FOUND`, `%NOTFOUND`, `%ROWCOUNT`, `%ISOPEN` are used to check cursor status and control logic.

---

## 🧪 Experiment 9: Transactions and Exception Handling

**1. What is a transaction?**
A transaction is a logical unit of work containing one or more SQL operations that must execute completely or not at all.

**2. ACID properties:**
Atomicity – all or nothing execution.
Consistency – database remains valid.
Isolation – concurrent transactions don't interfere.
Durability – committed changes are permanent.

**3. Concurrency control:**
Ensures multiple transactions execute safely without conflicts.
Techniques include locking, timestamps, and serializability.

**4. COMMIT and ROLLBACK usage:**
COMMIT saves all changes permanently.
ROLLBACK undoes changes to the last commit or savepoint.

**5. Exception handling:**
Allows graceful handling of runtime errors using `EXCEPTION` block in PL/SQL.

**6. Predefined exceptions:**
NO_DATA_FOUND, TOO_MANY_ROWS, ZERO_DIVIDE, VALUE_ERROR, INVALID_NUMBER.

**7. User-defined exceptions:**
Declared by the programmer using `EXCEPTION` and triggered with `RAISE` statement.

**8. If exceptions are not handled:**
PL/SQL terminates execution, and uncommitted changes may be rolled back automatically.

---

## 🧪 Experiment 10: Case Study / Mini Project

**1. Explain project's database design:**
Entities, attributes, and relationships are identified based on system requirements.
Tables are created with proper data types and keys.

**2. Entities and relationships identified:**
Example: Student, Course, Enrollment with relations like Student ENROLLS in Course.

**3. Normalization forms applied:**
Applied 1NF, 2NF, 3NF to remove redundancy and maintain integrity.

**4. Queries or reports implemented:**
Used SELECT, JOIN, GROUP BY, and aggregate functions to generate reports.

**5. Constraints used:**
Primary key, Foreign key, Not Null, Unique, and Check constraints ensure correct data.

**6. Ensuring data integrity:**
Through keys, constraints, triggers, and transactions to maintain consistent data.

**7. Challenges faced:**
Mapping relationships, handling large data sets, and query optimization.

**8. Possible improvements:**
Adding indexes, materialized views, advanced reporting, or improving normalization.

---

✅ **All 10 Experiments — 4–5 Line Detailed Answers Ready for Viva or Lab Exam**

---

If you want, I can **compile all 10 experiments into a neat, formatted PDF** with **bold headings, proper spacing, and page breaks** so it's ready to print or study.

Do you want me to do that next?