PL/SQL Interview Questions and Answers

Basic PL/SQL Interview Questions (1-25)

1. What is PL/SQL?

PL/SQL (Procedural Language/SQL) is Oracle's procedural extension of SQL, allowing procedural programming constructs like loops, conditions, and exceptions.

2. What are the benefits of PL/SQL?

- Supports procedural constructs like loops and conditions.
- Improves performance through block execution.
- Enhances security with stored procedures.
- Supports exception handling.

3. What is the difference between SQL and PL/SQL?

Feature	SQL	PL/SQL
Type	Query language	Procedural extension
Execution	Executes one statement at a time	Executes blocks of code
Control Structures	Not supported	Supports loops, conditions, exceptions

4. What are PL/SQL blocks?

A PL/SQL block consists of:

- **Declaration Section** (`DECLARE`)
- **Executable Section** (`BEGIN ... END;`)
- **Exception Handling Section** (`EXCEPTION`)

```
DECLARE
  v_name VARCHAR2(50);
BEGIN
  v_name := 'PL/SQL';
  DBMS_OUTPUT.PUT_LINE(v_name);
END;
```

5. What are anonymous blocks in PL/SQL?

A block without a name, executed once without storing in the database.

6. What are stored procedures in PL/SQL?

A named PL/SQL block stored in the database and executed with parameters.

```
CREATE PROCEDURE get_employee (emp_id IN NUMBER)
AS
BEGIN
  DBMS_OUTPUT.PUT_LINE(emp_id);
END;
```

7. What is a function in PL/SQL?

A stored PL/SQL block that returns a value.

```
CREATE FUNCTION get_salary(emp_id NUMBER) RETURN NUMBER AS
  v_salary NUMBER;
BEGIN
  SELECT salary INTO v_salary FROM employees WHERE id = emp_id;
  RETURN v_salary;
END;
```

8. What is the difference between a procedure and a function?

Feature	Procedure	Function
Return Type	No return value	Returns a value
Usage	Called independently	Used in SQL expressions

9. What are PL/SQL variables?

Named storage locations holding values of different data types.

10. What is an exception in PL/SQL?

An error-handling mechanism.

11. What are the types of PL/SQL exceptions?

- **Predefined exceptions** (e.g., `NO_DATA_FOUND`, `ZERO_DIVIDE`).
- **User-defined exceptions** (declared using `EXCEPTION` keyword).

```
DECLARE
  my_exception EXCEPTION;
BEGIN
  RAISE my_exception;
EXCEPTION
  WHEN my_exception THEN
    DBMS_OUTPUT.PUT_LINE('User-defined exception occurred.');
END;
```

12. What is a trigger in PL/SQL?

A special procedure that executes automatically in response to database events.

```
CREATE TRIGGER trg_after_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('New employee added!');
END;
```

13. What are the types of triggers?

- **Row-level triggers** (`FOR EACH ROW`).
- **Statement-level triggers** (fires once per statement).
- **BEFORE and AFTER triggers**.
- **INSTEAD OF triggers** (for views).

14. What is the difference between a row-level and statement-level trigger?

Type	Execution
Row-level	Fires for each row affected
Statement-level	Fires once per SQL statement

15. What is a cursor in PL/SQL?

A pointer to a result set in PL/SQL.

16. What are the types of cursors?

- **Implicit cursor** (automatically created for `SELECT` statements).
- **Explicit cursor** (manually declared using `CURSOR` keyword).

17. How do you declare an explicit cursor?

```
DECLARE
  CURSOR emp_cursor IS SELECT * FROM employees;
BEGIN
  FOR emp IN emp_cursor LOOP
    DBMS_OUTPUT.PUT_LINE(emp.name);
  END LOOP;
END;
```

18. What is the difference between implicit and explicit cursors?

Feature	Implicit Cursor	Explicit Cursor
Declaration	Automatic	Manual (`CURSOR` keyword)
Control	Automatically fetched	Requires `OPEN`, `FETCH`, `CLOSE`

19. What is `%ROWTYPE` in PL/SQL?

A composite data type that holds a row from a table.

```
DECLARE
  emp_rec employees%ROWTYPE;
BEGIN
  SELECT * INTO emp_rec FROM employees WHERE id = 1;
  DBMS_OUTPUT.PUT_LINE(emp_rec.name);
END;
```

20. What is %TYPE in PL/SQL?

Assigns a variable the same data type as a column.

```
DECLARE
  emp_name employees.name%TYPE;
BEGIN
  SELECT name INTO emp_name FROM employees WHERE id = 1;
  DBMS_OUTPUT.PUT_LINE(emp_name);
END;
```

21. What are composite data types in PL/SQL?

Collections like **records**, **tables**, and **VARRAYs**.

22. What is a PL/SQL collection?

A data structure that stores multiple values: **Associative Arrays, Nested Tables, VARRAYs**.

23. What is a nested table in PL/SQL?

A collection that grows dynamically.

```
DECLARE
  TYPE emp_table IS TABLE OF VARCHAR2(50);
  v_emp emp_table := emp_table('John', 'Jane');
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_emp(1));
END;
```

24. What is an associative array in PL/SQL?

A collection of key-value pairs.

```
DECLARE
  TYPE emp_assoc IS TABLE OF VARCHAR2(50) INDEX BY PLS_INTEGER;
  emp emp_assoc;
BEGIN
  emp(1) := 'John';
  emp(2) := 'Jane';
  DBMS_OUTPUT.PUT_LINE(emp(1));
END;
```

25. What is a VARRAY in PL/SQL?

A fixed-size collection.

```
DECLARE
  TYPE emp_varray IS VARRAY(5) OF VARCHAR2(50);
  emp_names emp_varray := emp_varray('Alice', 'Bob');
BEGIN
  DBMS_OUTPUT.PUT_LINE(emp_names(1));
END;
```

Here are **25 Intermediate Level PL/SQL Interview Questions and Answers** (Questions 26-50):

Intermediate PL/SQL Interview Questions (26-50)

26. What are packages in PL/SQL?

Answer: A **package** is a collection of **procedures, functions, variables, cursors, and exceptions** grouped together as a single unit. It consists of:

- **Package Specification** (declares objects)
- **Package Body** (defines objects)

Example:

```
CREATE OR REPLACE PACKAGE emp_pkg AS

  PROCEDURE get_employee(emp_id NUMBER);

END emp_pkg;



CREATE OR REPLACE PACKAGE BODY emp_pkg AS

  PROCEDURE get_employee(emp_id NUMBER) IS

    v_name employees.name%TYPE;

  BEGIN

    SELECT name INTO v_name FROM employees WHERE id = emp_id;
    DBMS_OUTPUT.PUT_LINE(v_name);
  END get_employee;
END emp_pkg;
```

---

27. What is the difference between a procedure and a package?

Feature	Procedure	Package
Definition	A single PL/SQL block	A collection of procedures, functions, variables
Encapsulation	No	Yes
Compilation	Each procedure compiles separately	The whole package compiles at once
Execution	Standalone execution	Must call procedures from the package

---

28. What is the difference between `%TYPE` and `%ROWTYPE`?

Feature	`%TYPE`	`%ROWTYPE`
Usage	Declares a variable with the same data type as a column	Declares a record with the same structure as a table row
Example	`v_name employees.name%TYPE;`	`emp_rec employees%ROWTYPE;`

29. What is the difference between an implicit and explicit cursor?

Feature	Implicit Cursor	Explicit Cursor
Definition	Automatically created by SQL statements	Defined and controlled by the programmer
Usage	`SELECT INTO` statements	`CURSOR` keyword
Example	`SELECT name INTO v_name FROM employees;`	`CURSOR emp_cursor IS SELECT * FROM employees;`

30. How do you declare and use an explicit cursor?

Answer:

```
DECLARE
  CURSOR emp_cursor IS SELECT name FROM employees;
  v_name employees.name%TYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO v_name;
  CLOSE emp_cursor;
  DBMS_OUTPUT.PUT_LINE(v_name);
END;
```

31. What is a cursor FOR loop in PL/SQL?

Answer: A `FOR` loop automatically opens, fetches, and closes a cursor.

```
BEGIN
  FOR emp IN (SELECT name FROM employees) LOOP
    DBMS_OUTPUT.PUT_LINE(emp.name);
  END LOOP;
END;
```

32. What is the difference between `LOOP`, `WHILE`, and `FOR` loops in PL/SQL?

Loop Type	Usage
`LOOP ... EXIT WHEN`	Executes indefinitely until an `EXIT` condition is met
`WHILE` Loop	Executes while a condition is `TRUE`
`FOR` Loop	Iterates a fixed number of times

33. What is bulk processing in PL/SQL?

Answer: Using `BULK COLLECT` and `FORALL` to improve performance in `SELECT` and `DML` operations.

```
DECLARE
  TYPE emp_names IS TABLE OF employees.name%TYPE;
  v_names emp_names;
BEGIN
```

```
  SELECT name BULK COLLECT INTO v_names FROM employees;
END;
```

---

34. What is the difference between `BULK COLLECT` and `FORALL`?

Feature	`BULK COLLECT`	`FORALL`
Usage	Fetch multiple rows into a collection	Performs DML operations in bulk
Example	`SELECT ... BULK COLLECT INTO ...`	`FORALL i IN ... INSERT INTO ...`

---

35. How do you use `FORALL` for bulk updates?

Answer:

```
DECLARE
  TYPE emp_ids IS TABLE OF employees.id%TYPE;
  v_ids emp_ids := emp_ids(1, 2, 3);
BEGIN
  FORALL i IN v_ids.FIRST .. v_ids.LAST
    UPDATE employees SET salary = salary * 1.1 WHERE id = v_ids(i);
END;
```

---

36. What is dynamic SQL in PL/SQL?

Answer: SQL that is constructed and executed at runtime using `EXECUTE IMMEDIATE`.

```
DECLARE
  v_sql VARCHAR2(100);
BEGIN
  v_sql := 'DELETE FROM employees WHERE id = 10';
  EXECUTE IMMEDIATE v_sql;
END;
```

---

37. How do you handle exceptions in dynamic SQL?

Answer: Using `BEGIN ... EXCEPTION` block.

```
BEGIN
  EXECUTE IMMEDIATE 'UPDATE employees SET salary = salary * 1.1';
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
```

---

38. What is the difference between `RAISE` and `RAISE_APPLICATION_ERROR`?

Feature	`RAISE`	`RAISE_APPLICATION_ERROR`
Usage	Raises an exception	Raises an error with a custom message

Example	`RAISE my_exception;`	`RAISE_APPLICATION_ERROR(-20001, 'Error message');`

---

39. What is a pragma in PL/SQL?

Answer: A compiler directive that influences execution behavior.
Example: `PRAGMA AUTONOMOUS_TRANSACTION;`

---

40. What is an autonomous transaction?

Answer: A transaction that runs independently from the main transaction.

```
DECLARE

  PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN

  INSERT INTO logs VALUES ('Error Occurred');

  COMMIT;

END;
```

---

41. What is a mutating table error in PL/SQL?

Answer: Occurs when a row-level trigger tries to modify the same table it is executing on.
Solution: Use a temporary table or package variable.

---

42. What is the difference between `SAVEPOINT`, `COMMIT`, and `ROLLBACK`?

Command	Description
`SAVEPOINT`	Marks a point in a transaction
`COMMIT`	Saves all changes permanently
`ROLLBACK TO SAVEPOINT`	Undoes changes to a savepoint

---

43. What are compound triggers in PL/SQL?

Answer: Triggers that have multiple execution sections for different events (`BEFORE`, `AFTER`, `INSTEAD OF`).

```
CREATE OR REPLACE TRIGGER emp_trg

FOR UPDATE ON employees

COMPOUND TRIGGER

  BEFORE STATEMENT IS BEGIN NULL; END BEFORE STATEMENT;

  AFTER EACH ROW IS BEGIN NULL; END AFTER EACH ROW;

  AFTER STATEMENT IS BEGIN NULL; END AFTER STATEMENT;

END emp_trg;
```

---

44. What is a REF cursor in PL/SQL?

Answer: A cursor that can be passed as a parameter.

```
DECLARE
```

```
  TYPE emp_refcur IS REF CURSOR;
  v_cursor emp_refcur;
BEGIN
  OPEN v_cursor FOR SELECT * FROM employees;
END;
```

---

45. What is the difference between a `PL/SQL Table` and a `Nested Table`?

Feature	PL/SQL Table	Nested Table
Storage	In memory only	Stored in the database
Persistence	Only in PL/SQL block	Persist after block execution

46. What is the difference between `IN`, `OUT`, and `IN OUT` parameters in PL/SQL procedures?

Parameter Type	Usage
`IN`	Passes values into the procedure (read-only)
`OUT`	Returns values from the procedure (write-only)
`IN OUT`	Passes values into and out of the procedure (read/write)

Example:

sql

CopyEdit

```
CREATE PROCEDURE update_salary (emp_id IN NUMBER, new_salary IN OUT
NUMBER) AS

BEGIN

  UPDATE employees SET salary = new_salary WHERE id = emp_id;

  SELECT salary INTO new_salary FROM employees WHERE id = emp_id;

END;
```

---

47. How do you find the second highest salary in a table using PL/SQL?

Answer:

sql

CopyEdit

```
SELECT MAX(salary)

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees);
```

Alternatively, using `ROWNUM`:

sql

CopyEdit

```
SELECT salary FROM (

  SELECT salary FROM employees ORDER BY salary DESC

) WHERE ROWNUM = 2;
```

---

48. What is a materialized view in PL/SQL?

Answer: A **materialized view** stores the results of a query physically in the database and can be refreshed periodically.

sql

CopyEdit

```
CREATE MATERIALIZED VIEW emp_mv

AS SELECT * FROM employees;
```

Types of Refresh Modes:

- **Fast Refresh** (only changes are updated)
- **Complete Refresh** (entire data is rebuilt)
- **Force Refresh** (chooses between fast or complete)

---

49. How do you handle deadlocks in PL/SQL?

Answer:
 Deadlocks occur when two transactions hold locks on resources that the other needs. To prevent them:

- **Order transactions properly**
- **Use `NOWAIT` to avoid waiting indefinitely**
- **Use `COMMIT` frequently**

Example of `NOWAIT`:

sql

CopyEdit

```
SELECT * FROM employees FOR UPDATE NOWAIT;
```

50. What are %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN in PL/SQL?

Attribute	Description
%FOUND	Returns TRUE if a SELECT INTO or FETCH operation found rows
%NOTFOUND	Returns TRUE if no rows were found
%ROWCOUNT	Returns the number of rows affected by the last DML operation
%ISOPEN	Returns TRUE if the cursor is open

Example:

sql

CopyEdit

```
DECLARE

  CURSOR emp_cursor IS SELECT * FROM employees;

  emp_record employees%ROWTYPE;

BEGIN

  OPEN emp_cursor;

  FETCH emp_cursor INTO emp_record;

  IF emp_cursor%FOUND THEN

    DBMS_OUTPUT.PUT_LINE('Row found!');

  END IF;
  CLOSE emp_cursor;

END;
```

Here are **25 Advanced PL/SQL Interview Questions and Answers (51-75):**

Advanced PL/SQL Interview Questions (51-75)

51. What is the difference between `ROWID`, `ROWNUM`, and `DENSE_RANK` in PL/SQL?

Feature	`ROWID`	`ROWNUM`	`DENSE_RANK`
Definition	Unique physical address of a row	Temporary sequence number assigned to a row in query result	Assigns ranking to rows without gaps
Example	`SELECT ROWID FROM employees;`	`SELECT ROWNUM FROM employees;`	`SELECT DENSE_RANK() OVER (ORDER BY salary DESC) FROM employees;`

---

52. How do you optimize PL/SQL queries for performance?

Answer:

- Use **indexes** to speed up searches
- Avoid `SELECT *` (fetch only required columns)
- Use `BULK COLLECT` for fetching multiple rows
- Use `FORALL` for batch updates
- Use **bind variables** to prevent hard parsing

- Use **EXPLAIN PLAN** to analyze queries

Example:

EXPLAIN PLAN FOR SELECT * FROM employees WHERE salary > 50000;

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

---

53. What is the difference between `DBMS_SQL` and `EXECUTE IMMEDIATE`?

Feature	`DBMS_SQL`	`EXECUTE IMMEDIATE`
Usage	Used for dynamic SQL execution	Used for immediate execution of dynamic SQL
Performance	Slightly slower	Faster
Example	`DBMS_SQL.PARSE(cursor_id, sql_query);`	`EXECUTE IMMEDIATE 'SELECT * FROM employees';`

---

54. How do you execute a stored procedure dynamically?

Answer: Using `EXECUTE IMMEDIATE`

EXECUTE IMMEDIATE 'BEGIN my_procedure(:param1, :param2); END;' USING 10, 'John';

---

55. What is an index-by table (associative array) in PL/SQL?

Answer: A PL/SQL collection indexed by a **string or number**, stored in memory.

DECLARE

```
  TYPE emp_table IS TABLE OF VARCHAR2(50) INDEX BY PLS_INTEGER;
  v_employees emp_table;
BEGIN
  v_employees(1) := 'John';
  DBMS_OUTPUT.PUT_LINE(v_employees(1));
END;
```

---

56. What are the advantages of using PL/SQL collections?

Answer:

- Improves performance using **BULK COLLECT** and **FORALL**
- Allows storage of multiple values like an array
- Useful for **batch processing**

---

57. What is the difference between `VARRAY`, `Nested Table`, and `Associative Array`?

Feature	`VARRAY`	`Nested Table`	`Associative Array`
Size	Fixed	Dynamic	Dynamic
Storage	Stored in table	Stored in table	Stored in memory
Indexing	Sequential	Can have gaps	String or number

58. How do you pass a collection to a stored procedure?

Answer:

```
CREATE OR REPLACE PROCEDURE process_ids(p_ids IN SYS.ODCINUMBERLIST) AS
BEGIN
  FOR i IN 1 .. p_ids.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(p_ids(i));
  END LOOP;
END;
```

59. What is `DBMS_PROFILER` in PL/SQL?

Answer: A built-in package for **performance profiling**.

```
EXEC DBMS_PROFILER.START_PROFILER('Test Profile');
EXEC DBMS_PROFILER.STOP_PROFILER;
```

60. How do you handle large CLOB or BLOB data in PL/SQL?

Answer: Using `DBMS_LOB`.

```
DECLARE
  v_clob CLOB;
  v_data VARCHAR2(100) := 'Large Text Data';
BEGIN
  DBMS_LOB.WRITEAPPEND(v_clob, LENGTH(v_data), v_data);
END;
```

61. What is the difference between a trigger and a stored procedure?

Feature	Trigger	Stored Procedure
Invocation	Automatically on DML	Manually executed
Execution Scope	Row or statement level	Only when called
Example	`BEFORE INSERT` trigger	`CREATE PROCEDURE`

62. What are compound triggers in PL/SQL?

Answer: Triggers that contain **multiple execution sections**.

```
CREATE OR REPLACE TRIGGER emp_trg
FOR UPDATE ON employees
COMPOUND TRIGGER
  BEFORE STATEMENT IS BEGIN NULL; END BEFORE STATEMENT;
  AFTER EACH ROW IS BEGIN NULL; END AFTER EACH ROW;
END emp_trg;
```

63. How do you track PL/SQL errors using `DBMS_UTILITY.FORMAT_ERROR_STACK`?

Answer:

```
BEGIN
```

```
 RAISE_APPLICATION_ERROR(-20001, 'Custom Error');
EXCEPTION
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE(DBMS_UTILITY.FORMAT_ERROR_STACK);
END;
```

---

64. What is `PRAGMA SERIALLY_REUSABLE` in PL/SQL?

Answer: It marks packages as **stateless** to improve memory usage.

```
PRAGMA SERIALLY_REUSABLE;
```

---

65. How do you improve PL/SQL exception handling?

Answer:

- Use **specific exceptions** (e.g., `NO_DATA_FOUND`)
- Log errors using `DBMS_UTILITY.FORMAT_ERROR_STACK`
- Use `WHEN OTHERS` carefully

---

66. What is the difference between `SYS_REFCURSOR` and a regular cursor?

Feature	`SYS_REFCURSOR`	**Regular Cursor**
Type	Weakly typed	Strongly typed

Flexibility	Can hold any result set	Fixed query

---

67. How do you execute a PL/SQL function inside an SQL query?

Answer:

```
SELECT my_function(salary) FROM employees;
```

---

68. What is a nested cursor in PL/SQL?

Answer: A cursor inside another cursor.

```
DECLARE
  CURSOR emp_cursor IS SELECT id FROM employees;
  CURSOR dept_cursor IS SELECT name FROM departments WHERE emp_id =
emp_cursor.id;
```

---

69. What is `DETERMINISTIC` in PL/SQL functions?

Answer: Ensures a function returns the same result for the same input.

```
CREATE OR REPLACE FUNCTION get_tax (p_salary NUMBER) RETURN NUMBER
DETERMINISTIC AS ...
```

---

70. How do you enable result caching in PL/SQL?

Answer:

```
CREATE OR REPLACE FUNCTION get_salary (p_id NUMBER) RETURN NUMBER
RESULT_CACHE AS ...
```

71. What is a mutating table error?

Answer: Occurs when a trigger modifies the same table it's working on.
Solution: Use an **autonomous transaction**.

72. What is a self-referencing foreign key in PL/SQL?

Answer: A foreign key that refers to the **same table**.

ALTER TABLE employees ADD CONSTRAINT fk_manager FOREIGN KEY (manager_id) REFERENCES employees(id);

73. How do you optimize joins in PL/SQL?

Answer:

- Use **indexed columns** for joins
- Use **hash joins** for large datasets
- Avoid **Cartesian joins**

74. What is a pipelined function in PL/SQL?

Answer: A function that returns **rows as a stream** instead of all at once.

CREATE FUNCTION get_employees RETURN emp_table PIPELINED AS ...

75. What is `DBMS_OUTPUT.ENABLE` used for?

Answer: It enables output messages in PL/SQL.

EXEC DBMS_OUTPUT.ENABLE(1000000);

Here are **25 Technical PL/SQL Interview Questions and Answers (76-100):**

Technical PL/SQL Interview Questions (76-100)

76. What is the difference between `BULK COLLECT` and `FORALL` in PL/SQL?

Feature	`BULK COLLECT`	`FORALL`
Purpose	Fetch multiple rows into a collection	Execute multiple DML statements in a batch
Performance	Reduces context switching	Improves performance for batch updates
Example	`SELECT salary BULK COLLECT INTO v_salaries FROM employees;`	`FORALL i IN v_ids.FIRST..v_ids.LAST INSERT INTO emp VALUES v_ids(i);`

---

77. How do you improve PL/SQL performance using `BULK COLLECT`?

Answer:
Using `BULK COLLECT` reduces the number of context switches between SQL and PL/SQL.

```
DECLARE
  TYPE emp_table IS TABLE OF employees%ROWTYPE;
  v_emps emp_table;
BEGIN
  SELECT * BULK COLLECT INTO v_emps FROM employees WHERE department_id = 10;
END;
```

78. What are autonomous transactions in PL/SQL?

Answer: Independent transactions that run within another transaction.

```
CREATE OR REPLACE PROCEDURE log_error(p_msg VARCHAR2) IS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO error_log (message) VALUES (p_msg);
  COMMIT;
END;
```

79. What are the types of triggers in PL/SQL?

Trigger Type	Description
BEFORE INSERT	Fires before an insert operation
AFTER UPDATE	Fires after an update operation
INSTEAD OF	Used for views

Example:

```
CREATE OR REPLACE TRIGGER trg_before_insert
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
```

```
  :NEW.salary := :NEW.salary + 1000;

END;
```

---

80. What is a mutating table error in PL/SQL? How do you fix it?

Answer:
 A **mutating table error** occurs when a row-level trigger queries or modifies the table it is based on.
 Solution: Use a compound trigger or an autonomous transaction.

Example of using a **compound trigger**:

```
CREATE OR REPLACE TRIGGER trg_emp_mutating

FOR UPDATE ON employees

COMPOUND TRIGGER

  TYPE t_salary_tab IS TABLE OF employees.salary%TYPE;

  g_salary_tab t_salary_tab;

  BEFORE STATEMENT IS BEGIN g_salary_tab := t_salary_tab(); END BEFORE
STATEMENT;

  AFTER EACH ROW IS BEGIN g_salary_tab.EXTEND; g_salary_tab(g_salary_tab.LAST)
:= :NEW.salary; END AFTER EACH ROW;

END trg_emp_mutating;
```

---

81. How do you fetch multiple rows in PL/SQL without using a cursor?

Answer: Using `BULK COLLECT`.

```
DECLARE

  TYPE emp_table IS TABLE OF employees%ROWTYPE;

  v_emps emp_table;

BEGIN
```

```
  SELECT * BULK COLLECT INTO v_emps FROM employees WHERE department_id = 10;
END;
```

---

82. How do you improve batch updates using FORALL?

```
DECLARE
  TYPE num_table IS TABLE OF NUMBER;
  v_ids num_table := num_table(1, 2, 3);
BEGIN
  FORALL i IN v_ids.FIRST..v_ids.LAST
    UPDATE employees SET salary = salary + 500 WHERE id = v_ids(i);
END;
```

---

83. What are the different ways to handle exceptions in PL/SQL?

Exception Type	Example
Predefined Exception	`WHEN NO_DATA_FOUND THEN ...`
User-defined Exception	`DECLARE my_exception EXCEPTION;`
Others	`WHEN OTHERS THEN ...`

---

84. How do you log errors in PL/SQL?

```
EXCEPTION
  WHEN OTHERS THEN
    INSERT INTO error_log (message) VALUES (SQLERRM);
    COMMIT;
```

---

85. What is the difference between `SAVEPOINT`, `ROLLBACK`, and `COMMIT`?

Statement	Description
`SAVEPOINT`	Marks a point for rollback
`ROLLBACK`	Undo changes
`COMMIT`	Saves changes permanently

Example:

```
SAVEPOINT sp1;
UPDATE employees SET salary = 5000 WHERE id = 1;
ROLLBACK TO sp1;
```

---

86. What is dynamic SQL in PL/SQL?

Answer: It allows SQL statements to be built at runtime using `EXECUTE IMMEDIATE`.

```
EXECUTE IMMEDIATE 'DELETE FROM employees WHERE id = :id' USING 10;
```

87. How do you return a table from a function in PL/SQL?

```
CREATE OR REPLACE FUNCTION get_employees RETURN SYS_REFCURSOR AS
  v_cursor SYS_REFCURSOR;
BEGIN
  OPEN v_cursor FOR SELECT * FROM employees;
  RETURN v_cursor;
END;
```

88. What are pipelined functions in PL/SQL?

Answer: Functions that return rows as a stream.

```
CREATE FUNCTION get_salaries RETURN emp_table PIPELINED AS ...
```

89. How do you improve PL/SQL query performance using indexing?

Use indexes on frequently queried columns.

```
CREATE INDEX emp_salary_idx ON employees(salary);
```

90. What is a `REF CURSOR` in PL/SQL?

Answer: A cursor that allows dynamic query execution.

```
TYPE emp_cursor IS REF CURSOR;
```

91. How do you use SYS_REFCURSOR in PL/SQL?

```
DECLARE
  v_cursor SYS_REFCURSOR;
  v_record employees%ROWTYPE;
BEGIN
  OPEN v_cursor FOR SELECT * FROM employees;
  FETCH v_cursor INTO v_record;
  CLOSE v_cursor;
END;
```

---

92. How do you delete duplicate rows in PL/SQL?

```
DELETE FROM employees WHERE rowid NOT IN (
  SELECT MIN(rowid) FROM employees GROUP BY name, salary
);
```

---

93. What are materialized views in PL/SQL?

A **materialized view** stores query results.

```
CREATE MATERIALIZED VIEW emp_mv AS SELECT * FROM employees;
```

---

94. How do you refresh a materialized view?

```
BEGIN
  DBMS_MVIEW.REFRESH('emp_mv');
END;
```

95. What are the different types of indexes in PL/SQL?

Index Type	Description
B-Tree Index	Default index type
Bitmap Index	Used for low-cardinality columns
Function-based Index	Index on expressions

Example:

```
CREATE INDEX idx_salary ON employees(salary);
```

96. What is `PRAGMA EXCEPTION_INIT`?

Answer: Maps user-defined errors to SQL error codes.

```
PRAGMA EXCEPTION_INIT(my_exception, -20001);
```

97. What is `%TYPE` and `%ROWTYPE`?

Feature	`%TYPE`	`%ROWTYPE`
Stores	Column type	Entire row

Example:

```
v_salary employees.salary%TYPE;

v_emp employees%ROWTYPE;
```

---

98. How do you call a PL/SQL procedure from Java?

Using JDBC CallableStatement:

```
CallableStatement stmt = conn.prepareCall("{call my_procedure(?)}");

stmt.setInt(1, 10);

stmt.execute();
```

---

99. What is DBMS_SCHEDULER?

Used for scheduling jobs in PL/SQL.

```
BEGIN

  DBMS_SCHEDULER.CREATE_JOB(

    job_name => 'my_job',

    job_type => 'PLSQL_BLOCK',

    job_action => 'BEGIN my_procedure; END;',

    start_date => SYSTIMESTAMP,

    enabled => TRUE);

END;
```

---

100. What is DBMS_STATS?

Collects optimizer statistics.

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('employees');
```