🔹 **Oracle PL/SQL Interview Questions and Answers (200 Q&A)**

Section 1: Basics of PL/SQL (Introduction)

1. **What is PL/SQL?**
   PL/SQL (Procedural Language/SQL) is Oracle's extension of SQL with procedural features like loops, conditions, and exception handling.
2. **Difference between SQL and PL/SQL?**
   SQL is declarative (used to query data), PL/SQL is procedural (adds logic like loops, conditions).
3. **What are PL/SQL block types?**

- Anonymous block
- Named block (Procedure/Function/Package)
- Trigger

4. **Structure of PL/SQL block?**

- **DECLARE** (optional): Variables, cursors.
- **BEGIN** (mandatory): Executable statements.
- **EXCEPTION** (optional): Error handling.
- **END** (mandatory).

5. **Advantages of PL/SQL?**

- Modular code
- Error handling
- Better performance with bulk operations
- Portability

6. **What are PL/SQL Datatypes?**
   Scalar (NUMBER, VARCHAR2, DATE), Composite (Record, Table), Large Objects (CLOB, BLOB).
7. **What is %TYPE in PL/SQL?**
   Used to declare variable with same datatype as column. Example: v_name emp.name%TYPE;

8. **What is %ROWTYPE?**
   Used to declare variable with same structure as table row. Example: v_emp emp%ROWTYPE;
9. **Difference between CHAR and VARCHAR2?**

- CHAR: Fixed length, pads with spaces.
- VARCHAR2: Variable length.

10. **What are Literals in PL/SQL?**
    Constants like numbers, strings, or dates directly written in code.

Section 2: Variables and Data Types

11. **How to declare a constant?**

pi CONSTANT NUMBER := 3.14;

12. **What is a bind variable?**
    A variable used in PL/SQL to share values between PL/SQL and SQL*Plus.
13. **What are Composite Data Types?**

- Records
- PL/SQL Tables (Associative Arrays)
- VARRAYs

14. **What is a PL/SQL Record?**
    A composite datatype grouping multiple fields. Example:

TYPE emp_rec IS RECORD (id NUMBER, name VARCHAR2(20));

15. **What is a PL/SQL Table (Associative Array)?**
    An index-by table:

TYPE num_table IS TABLE OF NUMBER INDEX BY PLS_INTEGER;

16. **What is a VARRAY?**
    A variable-size array with maximum size defined.
17. **Difference between Collection Types?**

- Assoc. Array: Unbounded, indexed by number/string.
- Nested Table: Unbounded, stored in DB.
- VARRAY: Fixed size, stored in DB.

18. **What is a Cursor Variable (REF CURSOR)?**
    Pointer to a query result set. Allows dynamic queries.
19. **What is Scope and Lifetime of variables?**

- **Scope**: Part of program where variable is accessible.
- **Lifetime**: Duration variable holds value in memory.

20. **What is PL/SQL Anchor Datatype?**
    Using %TYPE or %ROWTYPE to link variable to table column datatype.

Section 3: Control Structures

21. **Types of PL/SQL Control Structures?**

- Conditional (IF, CASE)
- Iterative (LOOP, FOR, WHILE)
- Sequential (GOTO, NULL)

22. **Syntax of IF statement?**

```
IF condition THEN
  -- statements
ELSE
  -- statements
END IF;
```

23. **What is CASE expression?**
    Alternative to multiple IF-ELSE. Example:

```
CASE grade
  WHEN 'A' THEN dbms_output.put_line('Excellent');
  WHEN 'B' THEN dbms_output.put_line('Good');
END CASE;
```

24. **What are LOOP types in PL/SQL?**

- Simple Loop
- FOR Loop
- WHILE Loop

25. **Difference between FOR and WHILE loop?**

- FOR: Executes fixed number of times.
- WHILE: Executes until condition false.

26. **Example of EXIT statement?**

```
LOOP
  EXIT WHEN counter > 10;
END LOOP;
```

27. **What is CONTINUE statement?**
    Skips current iteration and continues loop.
28. **What is GOTO in PL/SQL?**
    Used to jump to a labeled statement. Avoided for readability.
29. **What is NULL statement?**
    Performs no action, used as placeholder.
30. **What is difference between EXIT and RETURN?**

- EXIT: Exits loop.
- RETURN: Exits entire procedure/function.

Section 4: Exception Handling

31. **What is Exception in PL/SQL?**
    An error condition during execution.
32. **Types of Exceptions?**

- Predefined (NO_DATA_FOUND, TOO_MANY_ROWS)
- User-defined
- Non-predefined

33. **How to handle exception?**

```
BEGIN
  -- code
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
    dbms_output.put_line('No data found');
END;
```

34. **What is OTHERS exception?**
    Catches all exceptions not explicitly handled.
35. **What is RAISE statement?**
    Used to raise exception explicitly.
36. **What is RAISE_APPLICATION_ERROR?**
    Raise custom exception with error number. Example:

```
RAISE_APPLICATION_ERROR(-20001, 'Custom error');
```

37. **What is Exception Propagation?**
    If exception not handled in block, it propagates to outer block.
38. **What is difference between RAISE and RAISE_APPLICATION_ERROR?**

- RAISE: Raises existing exception.
- RAISE_APPLICATION_ERROR: Creates custom exception.

39. **What is PRAGMA EXCEPTION_INIT?**
    Maps user-defined exception to Oracle error number.
40. **What is Dup_Val_On_Index Exception?**
    Raised when inserting duplicate value in unique index column.

Section 5: Cursors

41. **What is Cursor?**
    Pointer to result of SQL query.
42. **Types of Cursors?**

- Implicit (handled automatically by Oracle)
- Explicit (declared and controlled by user)

43. **Steps in Explicit Cursor usage?**

- Declare
- Open
- Fetch
- Close

44. **What is Cursor FOR loop?**
    Automatically opens, fetches, and closes cursor.

```
FOR rec IN cursor LOOP
  dbms_output.put_line(rec.name);
END LOOP;
```

45. **What are Cursor Attributes?**

- %FOUND
- %NOTFOUND
- %ROWCOUNT
- %ISOPEN

46. **What is a Ref Cursor?**
    Cursor variable pointing to query result dynamically.

47. **Difference between Strong and Weak Ref Cursor?**

- Strong: Bound to specific return type.
- Weak: Can return any query.

48. **What is Bulk Collect with Cursor?**
    Fetch multiple rows into collection in one go.

```
FETCH c1 BULK COLLECT INTO v_tab;
```

49. **What is FOR UPDATE cursor?**
    Locks rows selected by cursor for update.

50. **What is Cursor WITH HOLD?**
    Keeps cursor open across commits.

PL/SQL Interview Questions (51–100)

Control Structures in PL/SQL

51. **What are control structures in PL/SQL?**
    They control the flow of execution: Conditional (IF, CASE), Iterative (LOOP, WHILE, FOR), Sequential (default execution).

52. **Difference between IF-THEN and IF-THEN-ELSE?**

- IF-THEN: Executes code if condition is true.
- IF-THEN-ELSE: Executes one block if true, another block if false.

53. **What is CASE in PL/SQL?**
    CASE provides multiple conditional checks like switch-case in other languages.
54. **When would you use CASE instead of IF?**
    When multiple mutually exclusive conditions need to be checked for better readability.
55. **Syntax of simple CASE expression in PL/SQL?**

```
CASE variable
  WHEN value1 THEN result1
  WHEN value2 THEN result2
  ELSE resultN
END;
```

56. **Syntax of searched CASE expression?**

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  ELSE resultN
END;
```

57. **What are different types of loops in PL/SQL?**
    - Basic LOOP
    - WHILE LOOP
    - FOR LOOP
58. **What is the difference between WHILE and FOR loop?**
    - WHILE executes as long as condition is true.
    - FOR runs for a defined number of iterations.
59. **Give syntax for a FOR loop in PL/SQL.**

```
FOR i IN 1..10 LOOP
  DBMS_OUTPUT.PUT_LINE(i);
END LOOP;
```

60. **What is EXIT in PL/SQL loops?**
    EXIT immediately terminates the loop.

61. **Difference between EXIT and EXIT WHEN?**

- EXIT: unconditional termination.
- EXIT WHEN: terminates loop when condition is true.

62. **What is CONTINUE in PL/SQL?**
    CONTINUE skips the current iteration and proceeds to the next loop cycle.

63. **What is NULL statement in PL/SQL?**
    It performs no action; used as a placeholder.

64. **What is GOTO in PL/SQL?**
    Transfers control to a labeled block in the code.

65. **Is GOTO recommended? Why?**
    Not recommended, reduces readability and maintainability (spaghetti code).

Exception Handling in PL/SQL

66. **What are exceptions in PL/SQL?**
    Exceptions are runtime errors that disrupt normal program execution.

67. **Types of exceptions in PL/SQL?**

- Predefined exceptions
- Non-predefined (Named) exceptions
- User-defined exceptions

68. **Examples of predefined exceptions?**
    NO_DATA_FOUND, TOO_MANY_ROWS, ZERO_DIVIDE, INVALID_CURSOR, DUP_VAL_ON_INDEX.

69. **How to declare a user-defined exception?**

my_exception EXCEPTION;

70. **How to raise a user-defined exception?**

RAISE my_exception;

71. **What is RAISE_APPLICATION_ERROR?**
    A procedure to raise custom error messages with numbers (-20000 to -20999).

72. **Syntax of RAISE_APPLICATION_ERROR?**

```
RAISE_APPLICATION_ERROR(-20001, 'Custom error message');
```

73. **What is WHEN OTHERS exception?**
    A catch-all handler for all exceptions not explicitly handled.

74. **Why should WHEN OTHERS always be last?**
    Because it catches all exceptions, and would otherwise block specific exception handlers.

75. **How to propagate exceptions to the caller?**
    By not handling them, or by re-raising using RAISE;.

76. **What is exception propagation?**
    If not handled in a block, the exception moves outward to the calling block.

77. **Can exceptions be re-raised?**
    Yes, using RAISE; in the exception block.

78. **What is PRAGMA EXCEPTION_INIT?**
    Associates a user-defined exception with an Oracle error code.

79. **Example of PRAGMA EXCEPTION_INIT?**

```
PRAGMA EXCEPTION_INIT(my_exception, -2292);
```

80. **Can you log exceptions in PL/SQL?**
    Yes, using DBMS_OUTPUT, DBMS_ERRLOG, or logging into a custom error table.

81. **What is the difference between predefined and user-defined exceptions?**

- Predefined: Oracle provides (e.g., NO_DATA_FOUND).
- User-defined: Created by programmers.

82. **What happens if an exception is not handled?**
    The program terminates abnormally and returns an error to the user.

83. **Can you nest exception handlers?**
    Yes, inner blocks can have their own exception handlers.

84. **What is exception scope in PL/SQL?**
    An exception is visible only in the block where it is declared, unless propagated.

85. **Can exceptions be stored in variables?**
    No, but exception error codes and messages can be stored using SQLCODE and SQLERRM.

86. **What are SQLCODE and SQLERRM?**

- SQLCODE: returns numeric error code.

- SQLERRM: returns error message.

87. **Can you raise multiple exceptions in one block?**
    Yes, but only one exception is raised at runtime.

88. **What is DUP_VAL_ON_INDEX exception?**
    Occurs when inserting a duplicate value into a unique index.

89. **What is TOO_MANY_ROWS exception?**
    Occurs when SELECT INTO returns more than one row.

90. **What is NO_DATA_FOUND exception?**
    Occurs when SELECT INTO returns no rows.

91. **What is ZERO_DIVIDE exception?**
    Raised when dividing a number by zero.

92. **What is INVALID_CURSOR exception?**
    Occurs when cursor operations are invalid (e.g., fetching from a closed cursor).

93. **What is STORAGE_ERROR exception?**
    Raised when PL/SQL runs out of memory or stack space.

94. **What is LOGIN_DENIED exception?**
    Occurs when a program attempts to log in with invalid credentials.

95. **What is PROGRAM_ERROR exception?**
    Occurs due to an internal error in PL/SQL engine.

96. **Can you use SAVEPOINT with exception handling?**
    Yes, to roll back partially in case of errors.

97. **What is autonomous transaction in exception handling?**
    A transaction that is independent of the main transaction, often used for logging errors.

98. **Can exceptions be logged asynchronously?**
    Yes, using autonomous transactions to insert error logs into a table.

99. **What is COLLECTION_IS_NULL exception?**
    Occurs when trying to access a null collection.

100. **What is VALUE_ERROR exception?**
    Raised when an arithmetic, conversion, truncation, or constraint error occurs.

Part 3: Procedures, Functions, Triggers, Packages (Q101–Q150)

Procedures & Functions

101. **What is a Procedure in PL/SQL?**
A named PL/SQL block that performs a task but may or may not return a value.

102. **What is a Function in PL/SQL?**
A named block that always returns a single value using RETURN.

103. **Difference between Procedure and Function?**

- **Procedure**: May not return a value, called independently.
- **Function**: Must return a value, can be used in SQL.

104. **Can a function call a procedure?**
Yes, functions can call procedures.

105. **Can a procedure return multiple values?**
Yes, via OUT parameters or records.

106. **How to execute a procedure?**

```
EXEC procedure_name(parameters);
```

107. **How to execute a function?**
Used inside SQL or PL/SQL:

```
SELECT function_name(param) FROM dual;
```

108. **What is the advantage of using stored procedures?**
Reusability, better performance, modularization, and security.

109. **Can we call a function in SELECT statements?**
Yes, if it doesn't modify data (deterministic).

110. **What are IN, OUT, IN OUT parameters?**

- IN: Input only
- OUT: Output only
- IN OUT: Both input and output

Triggers

111. **What is a Trigger?**
A stored block executed automatically when an event occurs (INSERT, UPDATE, DELETE).

112. **Types of Triggers in PL/SQL?**

- Row-level
- Statement-level
- BEFORE trigger
- AFTER trigger
- INSTEAD OF trigger

113. **What is a Row-level trigger?**
Executes for each row affected.

114. **What is a Statement-level trigger?**
Executes once per SQL statement.

115. **What is an INSTEAD OF trigger?**
Defined on views to perform DML operations.

116. **Can we create a trigger on a view?**
Yes, but only INSTEAD OF triggers.

117. **Can triggers call procedures?**
Yes, triggers can invoke procedures and functions.

118. **What is the difference between BEFORE and AFTER triggers?**

- BEFORE: Executes before DML.
- AFTER: Executes after DML.

119. **What is a Mutating Table error?**
Occurs when a row-level trigger tries to query/modify the same table.

120. **How to avoid mutating table error?**
Use statement-level triggers, compound triggers, or packages.

Packages

121. **What is a Package in PL/SQL?**
A group of related procedures, functions, variables, and cursors stored together.

122. **Components of a Package?**

- Package Specification (declaration)
- Package Body (implementation)

123. **Advantages of Packages?**
Encapsulation, reusability, better performance, modular code.

124. **What is Package Specification?**
Declares functions, procedures, constants, and variables (interface).

125. **What is Package Body?**
Contains implementation of procedures/functions declared in spec.

126. **Can a package exist without a body?**
Yes, if it only declares variables, constants, or cursors.

127. **Can two packages have procedures with the same name?**
Yes, as long as they are in different packages.

128. **What is the difference between public and private procedures in packages?**

- Public: Declared in package spec.
- Private: Declared in package body only.

129. **Can a package be overloaded?**
Yes, functions/procedures inside a package can be overloaded.

130. **What are Forward Declarations in packages?**
Declaring subprograms in package spec before their definition in body.

Advanced Procedures & Triggers

131. **What are Autonomous Transactions?**
Independent transactions inside PL/SQL (using PRAGMA AUTONOMOUS_TRANSACTION).

132. **When to use Autonomous Transactions?**
For logging, auditing, or separate commits/rollbacks.

133. **Can triggers perform COMMIT or ROLLBACK?**
No, except in autonomous transactions.

134. **What is a Compound Trigger?**
A trigger that combines multiple timing points (before/after row/statement).

135. **Use cases of Compound Triggers?**
Avoiding mutating table errors, handling multiple events together.

136. **What is a DDL Trigger?**
A trigger that fires on DDL statements (CREATE, DROP, ALTER).

137. **What is a Database Trigger?**
Trigger defined at database level for auditing or logging.

138. **Can a trigger call another trigger?**
Yes, indirectly, but may cause recursion.

139. **What is a FOLLOWS clause in triggers?**
Specifies execution order of multiple triggers on the same event.

140. **Difference between Database Triggers and Application Triggers?**

- DB Trigger: Executes at DB level.
- App Trigger: Executed via application logic.

Practical Usage

141. **Why use stored functions instead of SQL built-in functions?**
For customized business logic not covered by built-ins.

142. **What is Exception Propagation in procedures/functions?**
If an exception is not handled, it propagates to the caller.

143. **What is Deterministic Function in PL/SQL?**
A function that returns the same result for the same inputs.

144. **Why mark functions as DETERMINISTIC?**
So Oracle can cache results for performance.

145. **Can triggers modify :NEW and :OLD values?**

- BEFORE INSERT/UPDATE: :NEW can be modified.
- AFTER: Only read access allowed.

146. **Difference between Procedure Overloading and Function Overloading?**
Both allow multiple subprograms with the same name but different parameters.

147. **What is NOCOPY in procedures?**
An optimization hint that passes OUT/IN OUT parameters by reference instead of by value.

148. **What is DBMS_UTILITY package used for?**
Provides utility functions like formatting, analyzing dependencies.

149. **What is DBMS_OUTPUT package used for?**
Displaying output/debugging messages in PL/SQL.

150. **What is PRAGMA in PL/SQL?**
Compiler directives (e.g., PRAGMA AUTONOMOUS_TRANSACTION, PRAGMA EXCEPTION_INIT).

Part 4: Dynamic SQL, Error Handling, Bulk Processing (Q151–Q200)

Dynamic SQL (EXECUTE IMMEDIATE, DBMS_SQL)

151. **What is Dynamic SQL in PL/SQL?**
SQL statements built and executed at runtime.

152. **Why use Dynamic SQL?**
When SQL structure (table/column names) is unknown at compile time.

153. **What are the ways to execute Dynamic SQL in PL/SQL?**

- EXECUTE IMMEDIATE (preferred)
- DBMS_SQL package

154. **Example of EXECUTE IMMEDIATE:**

```
EXECUTE IMMEDIATE 'DELETE FROM employees WHERE dept_id = :1' USING 10;
```

155. **What is the difference between Static and Dynamic SQL?**

- Static: Known at compile time.
- Dynamic: Built and run at runtime.

156. **What is DBMS_SQL package used for?**
To parse, bind, execute, and fetch SQL dynamically (older method).

157. **Which is better: EXECUTE IMMEDIATE or DBMS_SQL?**
EXECUTE IMMEDIATE (simpler, faster). DBMS_SQL is used for complex cases.

158. **Can Dynamic SQL be used in Functions?**
Yes, but must follow restrictions (deterministic usage).

159. **How to fetch rows using Dynamic SQL?**
Using EXECUTE IMMEDIATE ... INTO or BULK COLLECT.

160. **Can DDL statements be executed in PL/SQL?**
Yes, but only via Dynamic SQL. Example:

```
EXECUTE IMMEDIATE 'CREATE TABLE test(id NUMBER)';
```

Error Handling & Exceptions

161. **What is Exception Handling in PL/SQL?**
Mechanism to handle runtime errors.

162. **Types of Exceptions in PL/SQL?**

- Predefined (e.g., NO_DATA_FOUND)
- User-defined
- Non-predefined

163. **Example of Predefined Exception:**

```
BEGIN
  SELECT salary INTO v_sal FROM emp WHERE emp_id = 100;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found');
END;
```

164. **What is OTHERS in Exception Handling?**
A catch-all handler for unhandled exceptions.

165. **What is SQLCODE and SQLERRM?**

- SQLCODE: Numeric error code.
- SQLERRM: Error message text.

166. **How to raise a user-defined exception?**

```
RAISE my_exception;
```

167. **Difference between RAISE and RAISE_APPLICATION_ERROR?**

- RAISE: Raises declared exception.
- RAISE_APPLICATION_ERROR: Raises custom Oracle error with error number/message.

168. **Syntax of RAISE_APPLICATION_ERROR:**

```
RAISE_APPLICATION_ERROR(-20001, 'Invalid Input');
```

169. **What is PRAGMA EXCEPTION_INIT?**
Maps Oracle error codes to user-defined exceptions.

170. **What happens if an exception is not handled?**
It propagates to the calling block and may terminate execution.

Bulk Processing (FORALL, BULK COLLECT)

171. **What is Bulk Collect in PL/SQL?**
Fetches multiple rows into collections in one go.

172. **Example of Bulk Collect:**

```
SELECT emp_id, name BULK COLLECT INTO v_ids, v_names FROM employees;
```

173. **What is FORALL in PL/SQL?**
Performs bulk DML operations using collections.

174. **Example of FORALL:**

```
FORALL i IN 1..emp_ids.COUNT
  INSERT INTO emp_backup VALUES (emp_ids(i), emp_names(i));
```

175. **Difference between FOR Loop and FORALL?**

- FOR loop: Executes row by row (slow).
- FORALL: Executes in bulk (faster).

176. **Can we use FORALL with DELETE/UPDATE?**
Yes, FORALL supports INSERT, UPDATE, DELETE, MERGE.

177. **What is LIMIT clause in Bulk Collect?**
Restricts number of rows fetched at a time (avoids memory issues).

178. **Example with LIMIT:**

```
FETCH cur BULK COLLECT INTO v_data LIMIT 100;
```

179. **What is SAVE EXCEPTIONS in FORALL?**
Allows bulk DML to continue even if some operations fail.

180. **How to get failed records in SAVE EXCEPTIONS?**
Using SQL%BULK_EXCEPTIONS.

Performance & Tuning

181. **How to improve PL/SQL performance?**

- Use Bulk Collect / FORALL

- Minimize context switching
- Use bind variables
- Avoid unnecessary commits

182. **What is Context Switching?**
The overhead of switching between SQL and PL/SQL engines.

183. **How to reduce Context Switching?**
Use bulk operations instead of row-by-row processing.

184. **What are Bind Variables?**
Placeholders that improve performance and security in SQL.

185. **What is Pipelined Function?**
A function that returns rows incrementally (like a table).

186. **What is Result Cache in PL/SQL?**
Caches function results for faster execution.

187. **What is DBMS_PROFILER used for?**
Performance analysis of PL/SQL code.

188. **What is DBMS_APPLICATION_INFO used for?**
To trace PL/SQL modules for monitoring performance.

189. **What is the difference between SQL and PL/SQL engine?**

- SQL engine executes SQL.
- PL/SQL engine executes procedural code + calls SQL engine.

190. **Why is PL/SQL faster than SQL in loops?**
Because of procedural features + bulk processing.

Security, Advanced Features, Miscellaneous

191. **What is Definer's Rights in PL/SQL?**
Code executes with privileges of the creator.

192. **What is Invoker's Rights?**
Code executes with privileges of the caller (AUTHID CURRENT_USER).

193. **Why use Invoker's Rights?**
For sharing code across multiple users with their privileges.

194. **What is DBMS_ASSERT package used for?**
To validate dynamic SQL inputs (avoid SQL injection).

195. **What is Edition-Based Redefinition in PL/SQL?**
Feature to upgrade PL/SQL objects without downtime.

196. **What are PL/SQL Collections?**
Data structures: Associative Arrays, Nested Tables, VARRAYs.

197. **What is Difference between Nested Table and VARRAY?**

- Nested Table: Unbounded, stored separately.
- VARRAY: Fixed size, stored inline.

198. **What are Weakly vs Strongly typed ref cursors?**

- Strong: Return type is fixed.
- Weak: Return type flexible, any query.

199. **What are Autonomous Transactions useful for?**
Independent logging, auditing, retry logic.

200. **Why choose PL/SQL over just SQL?**

- Adds procedural features
- Exception handling
- Modular programming
- Performance optimization
- Security & encapsulation