

## PLSQL Developer | Interview Preparation | Previously asked - IT Companies

Subject 2: PLSQL Interview Prep

Chapter 2.1: EASY | MODERATE

Topic 2.1.1: Previously asked Interview Questions

Concept 1: Set - 01

### 1. What is an explicit cursor in PL/SQL? How do you declare, open, fetch, and close one?

**Answer:**

*An explicit cursor is a named pointer to a query result set that you manually control.*

**Steps:**

**DECLARE**

**CURSOR c\_emp IS SELECT empno, ename FROM emp WHERE deptno = 10;**

**v\_emp c\_emp%ROWTYPE;**

**BEGIN**

**OPEN c\_emp;**

**LOOP**

**FETCH c\_emp INTO v\_emp;**

**EXIT WHEN c\_emp%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(v\_emp.empno || ' ' || v\_emp.ename);**

**END LOOP;**

**CLOSE c\_emp;**

**END;**

**Source | Web: Infosys & TCS interviews.**

### 2. What is an implicit cursor? What attributes does it have?

**Answer:**

*\* Created automatically by Oracle for single SQL statements (e.g., INSERT, UPDATE, DELETE, SELECT INTO).*

*\* Attributes: %FOUND, %NOTFOUND, %ROWCOUNT, %ISOPEN.*

**Source | Web:** Wipro interview.

### 3. Difference between Procedure and Function.

**Answer:**

*\* Function: must return a value, can be used in SQL.*

*\* Procedure: may or may not return values (via OUT params), cannot be directly used in SQL.*

**Source | Web:** TCS, Accenture.

### 4. What is a package? What are its parts?

**Answer:**

*A package is a collection of related procedures, functions, cursors, types.*

*\* Specification: public declarations.*

*\* Body: implementations + private elements.*

**Source | Web:** Capgemini.

### 5. Benefits of using Packages.

**Answer:**

- \* **Modularity**
- \* **Encapsulation (hide implementation)**
- \* **Performance (loaded once)**
- \* **Easier maintenance**
- \* **Global shared variables**

**Source | Web: HCL interview.**

#### **6. What is an exception? Types?**

**Answer:**

- \* **An exception is a runtime error.**
- \* **Predefined exceptions: NO\_DATA\_FOUND, TOO\_MANY\_ROWS, ZERO\_DIVIDE.**
- \* **User-defined: declared with EXCEPTION keyword.**

**Source | Web: Infosys.**

#### **7. How do you propagate exceptions from a function/procedure?**

**Answer:**

- \* **Handle locally with EXCEPTION block.**
- \* **Or let it bubble up to caller.**
- \* **Use RAISE to rethrow.**

**Source | Web: Cognizant.**

#### **8. Write a procedure to increase an employee's salary by 10%.**

**Answer:**

```
CREATE OR REPLACE PROCEDURE increase_sal(p_emp_id NUMBER, p_new_sal OUT NUMBER) IS
v_sal NUMBER;
BEGIN
SELECT salary INTO v_sal FROM emp WHERE empno = p_emp_id;
v_sal := v_sal * 1.10;
UPDATE emp SET salary = v_sal WHERE empno = p_emp_id;
p_new_sal := v_sal;
EXCEPTION
WHEN NO_DATA_FOUND THEN
p_new_sal := NULL;
DBMS_OUTPUT.PUT_LINE('Employee not found');
END;
```

**Source | Web: Infosys.**

#### **9. What is %TYPE and %ROWTYPE?**

**Answer:**

**\* %TYPE: variable inherits datatype from a column/variable.**

**Example: v\_name emp.ename%TYPE;**

**\* %ROWTYPE: record inherits structure of entire row.**

**Example: rec\_emp emp%ROWTYPE;**

**Source | Web: TCS.**

#### **10. What is a cursor FOR loop? Difference from explicit cursor loop.**

**Answer:**

*Cursor FOR loop automatically opens, fetches, and closes the cursor.*

```
FOR rec IN (SELECT empno, ename FROM emp) LOOP
DBMS_OUTPUT.PUT_LINE(rec.empno || ' ' || rec.ename);
END LOOP;
```

*\* Explicit loop requires manual OPEN, FETCH, CLOSE.*

**Source | Web:** Wipro.

#### 11. What is a REF CURSOR?

**Answer:**

*\* A pointer to a query result set that can be passed between programs.*

*\* Useful for dynamic queries and returning results to client apps.*

**Source | Web:** Oracle Corporation interviews.

#### 12. How do you handle “no rows found” in SELECT INTO?

**Answer:**

*By handling NO\_DATA\_FOUND exception:*

```
BEGIN
SELECT ename INTO v_name FROM emp WHERE empno = 999;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No such employee');
END;
```

**Source | Web:** Infosys.

#### 13. What is PRAGMA EXCEPTION\_INIT?

**Answer:**

*Associates a user-defined exception with a specific Oracle error code.*

```
e_invalid_salary EXCEPTION;  
PRAGMA EXCEPTION_INIT(e_invalid_salary, -20001);
```

**Source | Web:** IBM.

**14. Can we overload functions/procedures in packages?**

**Answer:**

*Yes, same name but different parameter lists.*

*Declared in package spec and implemented in body.*

**Source | Web:** Capgemini.

**15. What happens if you perform DML inside a function called from SQL?**

**Answer:**

*Not allowed (causes errors like mutating table), because functions used in SQL must not have side effects.*

**Source | Web:** Infosys.

**16. Cursor attributes for explicit cursors.**

**Answer:**

- \* %ISOPEN
- \* %FOUND
- \* %NOTFOUND
- \* %ROWCOUNT

**Source | Web:** Wipro.

**17. What is a mutating table error? How to avoid?**

**Answer:**

*Occurs when a trigger queries the same table it modifies.*

**Solutions:**

- \* *Use statement-level trigger*
- \* *Use package variables*
- \* *Use compound triggers*

**Source | Web:** TCS.

#### 18. How to run package initialization once per session?

**Answer:**

*Include initialization code in the package body's BEGIN ... END; section.*

**Source | Web:** Oracle.

#### 19. How to handle exceptions in an autonomous transaction?

**Answer:**

*Declare with PRAGMA AUTONOMOUS\_TRANSACTION. Handle exceptions within that block, commit/rollback independently.*

**Source | Web:** Infosys.

#### 20. What is WHEN OTHERS? Precautions?

**Answer:**

*Catch-all exception handler.*

*Precaution: Always log and re-raise, otherwise it hides real errors.*

**Source | Web:** TCS.

#### 21. Can package spec have a cursor declaration?

**Answer:**

*Yes. Cursor declared in spec is public, can be opened/fetched in body or other units.*

**Source | Web:** Capgemini.

## 22. What is BULK COLLECT and FORALL?

**Answer:**

*\* BULK COLLECT: fetch multiple rows into collection in one go.*

*\* FORALL: perform bulk DML using collections.*

*Improves performance by reducing context switches.*

**Source | Web:** Infosys.

## 23. Row-level vs statement-level triggers?

**Answer:**

*\* Row-level: fires once per row.*

*\* Statement-level: fires once per statement.*

**Source | Web:** Accenture.

## 24. How to raise custom user-defined exception with a message?

**Answer:**

*Use RAISE\_APPLICATION\_ERROR.*

*IF v\_salary < 0 THEN*

*RAISE\_APPLICATION\_ERROR(-20001, 'Invalid salary');*

*END IF;*

**Source | Web:** HCL.

## 25. What is a function-based index?

**Answer:**

***Index on the result of a function applied to a column (e.g., UPPER(name)).***

***Improves performance for queries using that function.***

**Source | Web: Infosys.**

Subject 2: PLSQL Interview Prep

Chapter 2.1: EASY | MODERATE

Topic 2.1.1: Previously asked Interview Questions

Concept 2: Set - 02

**Set 2: 25 PL/SQL Questions & Model Answers | Similar to previous questions**

**1. How would you design a PL/SQL function that returns a REF CURSOR, based on dynamic filters (where clause is dynamic) provided by the caller?**

**Answer:**

- \* Use a weak-typed REF CURSOR or define a strong one with a record type.*
- \* Build the SQL statement as a string depending on which filters are provided (e.g. only those not null).*
- \* OPEN refcur FOR sql\_stmt; inside the function.*
- \* Handle exceptions.*
- \* Example skeleton:*

```
CREATE OR REPLACE FUNCTION get_emps_dyn(p_deptno NUMBER, p_job VARCHAR2)
RETURN SYS_REFCURSOR
IS
  l_ref SYS_REFCURSOR;
  l_sql VARCHAR2(1000) := 'SELECT * FROM emp WHERE 1=1';
BEGIN
  IF p_deptno IS NOT NULL THEN
    l_sql := l_sql || ' AND deptno = :deptno';
  END IF;
  IF p_job IS NOT NULL THEN
    l_sql := l_sql || ' AND job = :job';
  END IF;
  OPEN l_ref FOR l_sql USING p_deptno, p_job;
  RETURN l_ref;
END;
```

**2. When you fetch bulk rows using BULK COLLECT, what are the memory/performance pitfalls and how do you mitigate them?**

**Answer:**

- \* Fetching too many rows at once can consume too much PGA memory.*
- \* Use LIMIT clause with BULK COLLECT to fetch in batches.*
- \* Use BULK COLLECT INTO collection LIMIT 1000 etc, process, then continue.*
- \* Release or re-initialize collections where possible.*

**3. Explain how you would implement error logging in PL/SQL packages for production code in a large scale system.**

**Answer:**

- \* Have central logging package (e.g. pkg\_error\_log) to insert error details into audit/error tables.*
- \* Log: error code, error message, stack trace, time, user, module name, input parameters.*
- \* Use WHEN OTHERS THEN handlers that call this log package, and optionally rethrow or wrap exceptions.*
- \* Possibly use DBMS\_UTILITY.FORMAT\_ERROR\_BACKTRACE to get stack trace.*

**4. What are compound PL/SQL triggers? How do they help in resolving mutating table errors?**

**Answer:**

- \* Compound trigger (available in Oracle 11g+) allows declaring sections for BEFORE STATEMENT, BEFORE EACH ROW, AFTER EACH ROW, AFTER STATEMENT in one trigger.*
- \* You can collect row data in BEFORE EACH ROW or AFTER EACH ROW into package/trigger-level collections and then act in AFTER STATEMENT so you avoid querying the mutating table during row processing.*

**5. How do you ensure transactional integrity when using autonomous transactions inside packages/procedures?**

**Answer:**

- \* Use PRAGMA AUTONOMOUS\_TRANSACTION for subprograms that must commit independently (e.g. logging).*
- \* Be careful: autonomous transaction commits are visible even if outer transaction rolls back.*
- \* Handle exceptions inside autonomous transaction properly, ensure rollback inside it if needed.*

**6. You have a procedure which many users will call concurrently; how to ensure that they do not step on each other when updating shared resources?**

**Answer:**

- \* Use locking: either explicit locking (e.g. `SELECT FOR UPDATE`) or `DBMS_LOCK` or use `LOCK TABLE` depending on requirement.*
- \* Use transactions with appropriate isolation levels.*
- \* Use optimistic locking if possible (version numbers) to avoid deadlocks.*

**7. How would you optimize PL/SQL code that's too slow when processing large volumes of rows?**

**Answer:**

- \* Reduce context switches: fewer SQL/PLSQL round-trips (use `BULK COLLECT`, `FORALL`).*
- \* Minimize usage of SQL inside loops.*
- \* Use indexes or function-based indexes.*
- \* Analyze execution plan for SQL parts.*
- \* Avoid unnecessary select statements; only select required columns.*
- \* Use proper exception handling so that exceptions do not degrade performance.*

**8. Design a package which tracks its own initialization (once per session or once per database), and supports versioning of its logic.**

**Answer:**

- \* In package spec, maintain a version constant.*
- \* Package body has an initialization section that runs once per session (first use).*
- \* Use a package global boolean `g_initialized`; in body block you can check and do setup only once.*
- \* To version logic: maintain conditionals based on version constant or table that has logic version, so that you can migrate behaviour.*

**9. What is the difference between strong vs weak REF CURSORS? When does Oracle enforce strong typing?**

**Answer:**

- \* Strong REF CURSOR: cursor type has fixed return columns EXACTLY matching some query. Parameters and column types are fixed.*
- \* Weak REF CURSOR: open for any query; column structure is not enforced at compile time.*
- \* Enforced when declared explicitly (e.g. TYPE strong\_refcur IS REF CURSOR RETURN emp%ROWTYPE;).*

**10. How to handle nested exceptions (one exception raised inside another handler) and ensuring that outer procedure catches meaningful message?**

**Answer:**

- \* Use RAISE to bubble up after logging.*
- \* Use WHEN OTHERS in inner block but wrap and re-raise with meaningful custom exception (or using RAISE\_APPLICATION\_ERROR).*
- \* Use EXCEPTION\_INIT for custom Oracle error codes if needed.*

**11. Explain “pipelined table functions”. How do you build one and when to use?**

**Answer:**

- \* A pipelined function returns rows incrementally as it generates them, like a virtual table. Useful for streaming large result sets.*
- \* Implement using PIPE ROW(...) inside the loop, and define function with PIPELINED keyword.*
- \* Can be used in SQL from TABLE(some\_pipelined\_fn(...)).*

**12. If a function is invoked from SQL and that function modifies database (DML), what problems can occur?**

**Answer:**

- \* It may violate purity levels => Oracle may disallow it (depending on version).*
- \* Could cause unexpected side effects.*
- \* Could lead to mutating table issues or integrity issues.*
- \* Generally best practice: functions called from SQL should be read-only.*

**13. How to manage version control / deployment of PL/SQL packages/functions in a team environment?**

**Answer:**

- \* Keep source in version control (Git etc).*
- \* Use scripts that can drop/compile in correct order.*
- \* Maintain change logs.*
- \* Use automated CI/CD pipeline for PL/SQL object deployment.*

**14. How do you trace performance problems in PL/SQL (profiling)?**

**Answer:**

- \* Use DBMS\_PROFILER or Oracle Trace tools.*
- \* Use DBMS\_UTILITY.FORMAT\_CALL\_STACK or DBMS\_APPLICATION\_INFO to instrument code.*
- \* Use SQL trace / TKPROF for the SQL inside PL/SQL.*

**15. What is the difference between exceptions with named PL/SQL variables vs Oracle error codes vs RAISE\_APPLICATION\_ERROR?**

**Answer:**

- \* Named PL/SQL exceptions defined in code are local, handled internally.*
- \* Oracle error codes are pre-defined or user defined with EXCEPTION\_INIT.*
- \* RAISE\_APPLICATION\_ERROR lets you throw a user-defined error code (-20000 to -20999), message; this can be caught by caller.*

**16. Write a PL/SQL procedure (or demonstrate) that copies rows from one table to another in batches of size 1000 using BULK COLLECT & FORALL, and commits between batches.**

**Answer:**

```
CREATE OR REPLACE PROCEDURE copy_in_batches IS
TYPE emp_tab IS TABLE OF emp%ROWTYPE;
l_batch emp_tab;
CURSOR c1 IS SELECT * FROM emp_source;
l_limit CONSTANT PLS_INTEGER := 1000;
BEGIN
OPEN c1;
LOOP
FETCH c1 BULK COLLECT INTO l_batch LIMIT l_limit;
EXIT WHEN l_batch.COUNT = 0;
FORALL i IN 1..l_batch.COUNT
INSERT INTO emp_target VALUES l_batch(i);
COMMIT;
END LOOP;
CLOSE c1;
END;
```

**17. What strategies do you use to handle schema changes for PL/SQL objects when underlying tables change (e.g. columns added, dropped)?**

**Answer:**

- \* Use %ROWTYPE or %TYPE so types adapt when column definitions change (with care).*
- \* Maintain good testing around PL/SQL objects.*
- \* Use exception handlers for invalid objects or re-compile dependent PL/SQL after schema change.*
- \* Possibly use versioning or backward compatibility layers.*

**18. How do you manage or prevent PL/SQL dependency/invalid-object problems over deployment?**

**Answer:**

- \* Establish proper compile order.*
- \* Use CREATE OR REPLACE so that new objects compile.*
- \* Use DEPENDENCIES views to check.*
- \* Use scripts / tools to find invalid objects and recompile them.*

**19. Explain how you could use PL/SQL to enforce complex business rules (e.g. multi-table consistency) that cannot be enforced purely via database constraints.**

**Answer:**

- \* Use stored procedures/triggers/packages.*
- \* Use BEFORE INSERT/UPDATE triggers to check conditions across multiple tables.*
- \* Possibly raise exceptions if violations.*
- \* Or build a validation layer in packages invoked by front-end or mid-tier.*

**20. How to manage exceptions in nested PL/SQL calls (procedure calls another procedure which may exception) such that error context is preserved?**

**Answer:**

- \* Inner procedure catches but RAISEs again (or wraps in another exception) so caller has context of where it failed.*
- \* Use DBMS\_UTILITY.FORMAT\_ERROR\_BACKTRACE and DBMS\_UTILITY.FORMAT\_ERROR\_STACK.*

**21. How can you use PL/SQL to process large XML or JSON payloads stored in the database?**

**Answer:**

- \* Use Oracle's XMLTYPE / JSON support.*
- \* Use DBMS\_LOB or streaming LOB reading.*
- \* Possibly use pipelined table functions to parse and return rows.*

**22. What are the pitfalls of using %ROWTYPE when table structure changes?**

**Answer:**

- \* If a column is dropped, code must be recompiled; if code references dropped column, it fails.*
- \* %ROWTYPE includes all columns; grabbing more data than needed. Might cause inefficiency.*

**23. What is PL/SQL Function Purity Levels? Why important if functions are used in SQL statements?**

**Answer:**

- \* Purity levels indicate what side effects or database state changes the function is allowed (reads, writes).*
- \* For functions used in SQL select statements etc, Oracle may enforce that they do not do DML, commit, etc.*

**24. Explain how to implement scheduler jobs invoking PL/SQL procedures, including retry logic in case of failures.**

**Answer:**

- \* Use DBMS\_SCHEDULER to schedule jobs.*
- \* The job can call a stored procedure.*
- \* In procedure, wrap logic in exception handlers; retry logic inside code or schedule job with retry attributes.*

**25. Test-case: You need to migrate data between two large Oracle databases; some records may violate constraints in target schema. How would you write a PL/SQL solution to attempt insert, log failures, continue with rest, and finally report summary?**

**Answer:**

- \* Use a loop over source cursor, inside try/except (PL/SQL exception block) for each record.*
- \* For successful, insert; for failure, catch exception, log into error\_log table with record key, error message.*
- \* Possibly commit in batches.*
- \* After full run, report success/fail counts.*

Subject 2: PLSQL Interview Prep

Chapter 2.2: MODERATE | DIFFICULT

Topic 2.2.1: Previously asked Interview Questions

Concept 1: Set - 03

**PL/SQL Advanced Interview Questions – Set 2 (25) | Previously asked questions - IT Companies**

**26. Google: How do you optimize a PL/SQL block that processes millions of rows?**

**Answer:**

**Use BULK COLLECT + FORALL to minimize context switches.**

**DECLARE**

**TYPE t\_emp IS TABLE OF emp%ROWTYPE;**

**l\_emps t\_emp;**

**BEGIN**

**SELECT \* BULK COLLECT INTO l\_emps FROM emp;**

**FORALL i IN 1..l\_emps.COUNT**

**UPDATE emp SET sal = sal \* 1.1 WHERE empno = l\_emps(i).empno;**

**END;**

**\* Reduces switching between SQL & PL/SQL engines.**

**\* Use LIMIT clause for memory safety.**

**27. Amazon: How do you debug a procedure in PL/SQL?**

**Answer:**

**\* Use DBMS\_OUTPUT.PUT\_LINE for logging.**

**\* Use DBMS\_TRACE or DBMS\_DEBUG package for line-by-line debugging.**

**\* Exception logging table + autonomous transactions for production systems.**

**28. Toptal: How do you implement error logging for a package that is used across multiple**

**modules?**

**Answer:**

**Create a centralized error logging package:**

```
CREATE OR REPLACE PACKAGE error_pkg IS
PROCEDURE log_error(p_module VARCHAR2, p_msg VARCHAR2);
END;
/
CREATE OR REPLACE PACKAGE BODY error_pkg IS
PROCEDURE log_error(p_module VARCHAR2, p_msg VARCHAR2) IS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
INSERT INTO error_log(module_name, error_message, log_date)
VALUES(p_module, p_msg, SYSDATE);
COMMIT;
END;
END;
```

**\* Reusable**

**\* Safe due to autonomous transaction**

**29. Schlumberger: Explain cursor variables (REF CURSOR) vs cursor expressions.**

**Answer:**

**\* REF CURSOR: pointer to a query result, flexible, can return dynamic queries.**

**\* Cursor Expression: used in SELECT to return a cursor per row, useful in hierarchical queries.**

**Example:**

```
SELECT deptno, CURSOR(SELECT ename FROM emp e WHERE e.deptno = d.deptno)
FROM dept d;
```

**30. McKinsey: How would you implement retry logic in a PL/SQL procedure?**

**Answer:**

**Use loop + exception handling:**

```
DECLARE
v_attempts NUMBER := 0;
BEGIN
LOOP
BEGIN
v_attempts := v_attempts + 1;
UPDATE emp SET sal = sal + 100 WHERE empno = 1001;
EXIT; -- success
EXCEPTION
WHEN OTHERS THEN
IF v_attempts < 3 THEN
DBMS_LOCK.SLEEP(1); -- wait and retry
ELSE
RAISE;
END IF;
END;
END LOOP;
END;
```

**31. Amazon: Can you return a cursor from a function?**

**Answer:**

**Yes, using REF CURSOR:**

```
CREATE OR REPLACE FUNCTION get_emp_cursor(p_dept NUMBER)
RETURN SYS_REFCURSOR IS
c SYS_REFCURSOR;
BEGIN
OPEN c FOR SELECT empno, ename FROM emp WHERE deptno = p_dept;
RETURN c;
END;
```

**32. Toptal: Explain deterministic functions in PL/SQL.**

**Answer:**

- \* Marked with DETERMINISTIC keyword.*
- \* Ensures Oracle that function returns same output for same input.*
- \* Useful in function-based indexes.*

**33. Google: How do you implement pagination with cursors in PL/SQL?**

**Answer:**

*Use OFFSET ... FETCH in SQL or ROWNUM logic.*

```
OPEN c FOR
SELECT empno, ename FROM emp
ORDER BY empno
OFFSET (p_page-1)*p_page_size ROWS
FETCH NEXT p_page_size ROWS ONLY;
```

**34. McKinsey: What is difference between RAISE and RAISE\_APPLICATION\_ERROR?**

**Answer:**

- \* RAISE: rethrows existing exception.*
- \* RAISE\_APPLICATION\_ERROR: raises custom error with number (-20000 to -20999).*

**35. Amazon: How to implement transaction savepoints in PL/SQL?**

**Answer:**

```
SAVEPOINT before_update;
UPDATE emp SET sal = sal*2 WHERE deptno=10;
-- Rollback to savepoint if needed
ROLLBACK TO before_update;

Useful in long transactions.
```

**36. Schlumberger: Explain invoker's rights vs definer's rights in PL/SQL.**

**Answer:**

- \* Definer's Rights (default): procedure runs with privileges of its creator.*
- \* Invoker's Rights (AUTHID CURRENT\_USER): runs with privileges of caller.*

**37. Toptal: How do you prevent SQL injection in PL/SQL dynamic SQL?**

**Answer:**

- \* Use DBMS\_ASSERT to sanitize inputs.*
- \* Example:*

```
v_sql := 'SELECT * FROM emp WHERE ename = ''' || DBMS_ASSERT.ENQUOTE_LITERAL(p_name) ||
''';
EXECUTE IMMEDIATE v_sql;
```

**38. Amazon: Explain compound triggers and their use case.**

**Answer:**

- \* Triggers that combine row-level + statement-level sections in one.*
- \* Helps avoid mutating table errors.*

**39. Google: How to improve performance of a function called inside SQL?**

**Answer:**

- \* Make it deterministic.*
- \* Avoid DML inside.*
- \* Use pipelined functions for large result sets.*

**40. McKinsey: What is a pipelined table function?**

**Answer:**

**Function returning rows as a table, one at a time, using PIPE ROW.**

```
FUNCTION get_emps RETURN emp_tab PIPELINED IS
BEGIN
FOR r IN (SELECT * FROM emp) LOOP
PIPE ROW(r);
END LOOP;
END;
```

**Used in SELECT \* FROM TABLE(get\_emps).**

#### **41. Schlumberger: How to log execution time of a procedure?**

**Answer:**

**Use DBMS\_UTILITY.GET\_TIME or SYSTIMESTAMP.**

```
v_start NUMBER := DBMS_UTILITY.GET_TIME;
-- procedure logic
DBMS_OUTPUT.PUT_LINE('Elapsed time: ' || (DBMS_UTILITY.GET_TIME - v_start));
```

#### **42. Amazon: How to avoid "ORA-06502: numeric or value error"?**

**Answer:**

- \* Ensure variable size matches data length.**
- \* Use %TYPE for safer declaration.**
- \* Example: v\_name emp.ename%TYPE;**

#### **43. Toptal: How do you handle deadlocks in PL/SQL?**

**Answer:**

- \* Oracle automatically detects deadlocks, raises exception ORA-00060.*
- \* Solution: catch exception, rollback, retry logic.*

**44. Google: Explain PRAGMA AUTONOMOUS\_TRANSACTION with example.**

**Answer:**

*Lets procedure commit independently of caller.*

```
PRAGMA AUTONOMOUS_TRANSACTION;  
BEGIN  
INSERT INTO audit_log VALUES(...);  
COMMIT;  
END;
```

**45. McKinsey: How to implement audit logging for all DML operations in PL/SQL?**

**Answer:**

*Create DML triggers or use Fine-Grained Auditing.*

```
CREATE OR REPLACE TRIGGER emp_audit_trg  
AFTER INSERT OR UPDATE OR DELETE ON emp  
FOR EACH ROW  
BEGIN  
INSERT INTO emp_audit VALUES(SYSDATE, USER, ORA_SYSEVENT);  
END;
```

**46. Schlumberger: What is DBMS\_SQL package and when to use it over EXECUTE IMMEDIATE?**

**Answer:**

- \* EXECUTE IMMEDIATE: simple, compile-time known queries.*
- \* DBMS\_SQL: when column list is dynamic, unknown at compile time.*

**47. Toptal: How to safely drop a package that is in use?**

**Answer:**

- \* Check dependency using USER\_DEPENDENCIES.*
- \* Disable package before dropping.*
- \* Use edition-based redefinition in critical apps.*

**48. Amazon: Can you call a procedure inside SELECT?**

**Answer:**

*No, procedures cannot return values in SQL.  
Only functions can be used in SELECT.*

**49. Google: How to avoid too many open cursors error?**

**Answer:**

- \* Always close explicit cursors.*
- \* Use cursor FOR loop where possible.*
- \* Increase OPEN\_CURSORS parameter only after fixing leaks.*

**50. McKinsey: Write a function that safely divides two numbers and returns NULL if denominator is zero.**

**Answer:**

```
CREATE OR REPLACE FUNCTION safe_div(p_num NUMBER, p_den NUMBER)
RETURN NUMBER IS
BEGIN
  IF p_den = 0 THEN
    RETURN NULL;
  ELSE
    RETURN p_num / p_den;
  END IF;
END;
```