	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql

51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9cIHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

Q - 1) What is the difference between “Stored Procedure” and “Function”

1. A procedure can have both input and output parameters, but a function can only have input parameters.
2. Inside a procedure we can use DML (INSERT/UPDATE/DELETE) statements. But inside a function we can’t use DML statements.
3. We can’t utilize a Stored Procedure in a Select statement. But we can use a function in a Select statement.
4. We can use a Try-Catch Block in a Stored Procedure but inside a function we can’t use a Try-Catch block.
5. We can use transaction management in a procedure but we can’t in a function.
6. We can’t join a Stored Procedure but we can join functions.
7. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section. But we can use a function anywhere.
8. A procedure can return 0 or n values (max 1024). But a function can return only 1 value that is mandatory.
9. A procedure can’t be called from a function but we can call a function from a procedure.

Q - 2) What is a trigger?

Database are set of commands that get executed when an event(Before Insert, After Insert, On Update, On delete of a row) occurs on a table, views

Q - 3) What is Nested Trigger

A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

Q - 4) What is Cursor

Cursor is a database object used by applications to manipulate data in a set on a row-by row basis, instead of the typical SQL commands that operate on all the rows in the set at one time. In order to work with a cursor we need to perform some steps in the following order:

1. Declare cursor
2. Open cursor

3. Fetch row from the cursor

```
4. Process fetched row

5. Close cursor

6. De-allocate cursor
```

Q - 5) What is B-Tree

The database server uses a B-tree structure to organize index information. B-Tree generally has following types of index pages or nodes: 1. root node: A root node contains node pointers to branch nodes which can be only one.

```
        2. branch node: A branch node contains pointers to leaf nodes
or other branch nodes which can be two or more.

        3. leaf nodes: A leaf node contains index items and horizontal
pointers to other leaf nodes which can be many.
```

Q - 6) Property of LOV "List from validate" ?

When Validate from List is set to True, Oracle Forms compares the current value of the text item to the values in the first column displayed in the LOV. Whenever the validation event occurs. If the value in the text item matches one of the values in the first column of the LOV, validation succeeds, the LOV is not displayed, and processing continues normally. If the value in the text item does not match one of the values in the first column of the LOV, Oracle Forms displays the LOV and uses the text item value as the search criteria to automatically reduce the list. Q - 7) WRITE A PROCEDURE TO DISPLAY SUM OF TWO NUMBERS. CREATE OR REPLACE PROCEDURE ADD_TWO_NUMBERS

IS

X NUMBER: =10;

Y NUMBER: =20;

BEGIN

DBMS_OUTPUT.PUT_LINE ('SUM OF TWO NUMBERS ='||(X+Y));

END ADD_TWO_NUMBERS;

/

Q - 8) Packages IN PL/SQL

PL/SQL packages are schema objects that groups logically related PL/SQL types, variables and subprograms. A package will have two mandatory parts: x Package specification x Package body or definition Package Specification The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms. All objects placed in the specification are called public objects. Any subprogram not in the package specification but coded in the package body is called a private object. The following code snippet shows a package specification having a single procedure. You can have many global variables defined and multiple procedures or functions inside a package.

```
CREATE PACKAGE cust_sal AS

PROCEDURE find_sal(c_id customers.id%type);

END cust_sal;

CREATE OR REPLACE PACKAGE BODY cust_sal AS PROCEDURE find_sal(c_id customers.id
%TYPE)

IS c_sal customers.salary%TYPE;

BEGIN SELECT salary INTO c_sal

FROM customers

WHERE id = c_id;

dbms_output.put_line('Salary:'|| c_sal);

END find_sal;

END cust_sal;

/

DECLARE

 code customers.id%type := &cc_id;

 BEGIN cust_sal.find_sal(code);

 END;

 /

Enter value for cc_id: 1

Salary: 3000

PL/SQL procedure successfully completed
```

Q - 9) write a program object in PL/SQL

CREATE OR REPLACE TYPE BODY rectangle AS MEMBER FUNCTION enlarge(inc number) return rectangle IS BEGIN return rectangle(self.length + inc, self.width + inc); END enlarge; MEMBER PROCEDURE display IS BEGIN dbms_output.put_line('Length:'|| length); dbms_output.put_line('Width:'|| width); END display; MAP MEMBER FUNCTION measure return number IS BEGIN return (sqrt(length*length + width*width)); END measure; END; / DECLARE r1 rectangle; r2 rectangle; r3 rectangle; inc_factor number := 5; BEGIN r1 := rectangle(3, 4); r2 := rectangle(5, 7); r3 := r1.enlarge(inc_factor); r3.display; IF (r1 > r2) THEN – calling measure function r1.display; ELSE r2.display; END IF; END; / Length: 8 Width: 9 Length: 5 Width: 7 PL/SQL procedure successfully completed.

Q - 10) Find the Largest of Three Numbers By Using Pl/Sql

DECLARE num1 NUMBER := 10; num2 NUMBER := 20; num3 NUMBER := 15; largest NUMBER; BEGIN IF num1 > num2 AND num1 > num3 THEN largest := num1; ELSIF num2 > num1 AND num2 > num3 THEN largest := num2; ELSE largest := num3; END IF; DBMS_OUTPUT.PUT_LINE('The largest number is:' || largest); END; /

Q - 11) Program to Reverse a String Pl/Sql

DECLARE original_str VARCHAR2(50) := 'PLSQL'; reversed_str VARCHAR2(50) := ''; BEGIN FOR i IN REVERSE 1..LENGTH(original_str) LOOP reversed_str := reversed_str || SUBSTR(original_str, i, 1); END LOOP; DBMS_OUTPUT.PUT_LINE('Reversed String:' || reversed_str); END; /

Q - 12) What is the difference between %FOUND and %ROWCOUNT

%FOUND returns TRUE if the most recent DML statement affected at least one row; %ROWCOUNT returns the number of rows affected.

Q - 13) PL/SQL Program with Exception Handling

DECLARE – Declare variables for division numerator NUMBER := 100; – Example value for the numerator denominator NUMBER := 0; – Example value for the denominator, which will cause an exception result NUMBER; BEGIN – Attempt to perform division result := numerator / denominator; DBMS_OUTPUT.PUT_LINE('Result:' || result); EXCEPTION – Handle division by zero error WHEN ZERO_DIVIDE THEN DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.'); – Handle any other unknown errors WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Error: An unexpected error occurred. Error code:' || SQLCODE || ', Message:' || SQLERRM); END; /

Q - 14) PL/SQL Debugging and Profiling

DBMS_OUTPUT: Used to print debugging information. DBMS_DEBUG: Oracle-supplied package that provides debugging tools. PL/SQL Profiler: Collects performance information, such as time spent in each line of code or subprogram.

Q - 15) PL/SQL Architecture

PL/SQL follows a client-server architecture where the client requests the execution of PL/SQL blocks, and the server processes these requests.

The architecture consists of key components:

PL/SQL Engine: Resides in the Oracle server and is responsible for processing PL/SQL code. It separates SQL and PL/SQL parts of the code, optimizing execution. SQL Engine: Executes SQL statements processed by the PL/SQL engine. Data Dictionary Cache and Library Cache: Used to store information about PL/SQL objects, such as procedures, functions, triggers, and packages, for faster access. Performance optimization in PL/SQL involves minimizing the execution time of code and improving its efficiency. Techniques include: Use Bulk Processing: Use BULK COLLECT and FORALL to fetch and manipulate multiple rows in a single context switch between SQL and PL/SQL. Minimize Context Switching: Context switching between the SQL engine and PL/SQL engine can degrade performance. Write SQL statements that perform as much processing as possible in a single pass. Use Bind Variables: Using bind variables reduces parsing time and improves SQL statement reuse. Leverage the PL/SQL Compiler Settings: Use PLSQL_OPTIMIZE_LEVEL to control the optimization level for PL/SQL code. Use Appropriate Data Types: Choosing the right data types for variables can help in minimizing memory usage and improving performance.

Q - 16) PL/SQL Program for Security Considerations

Security Considerations

—-This procedure safely inserts a user's information into a table using bind variables, encrypts sensitive data, and logs the operation.

CREATE OR REPLACE PROCEDURE insert_user_info (p_user_id NUMBER, p_user_name VARCHAR2, p_password VARCHAR2) AUTHID CURRENT_USER – Procedure executes with the privileges of the invoker IS – Declare an encryption key (should be securely stored and managed) l_key RAW(32) := UTL_RAW.CAST_TO_RAW('0123456789ABCDEF0123456789ABCDEF'); l_encrypted_password RAW(2000); BEGIN – Encrypt the password using DBMS_CRYPTO l_encrypted_password := DBMS_CRYPTO.ENCRYPT(UTL_I18N.STRING_TO_RAW(p_password, 'AL32UTF8'), DBMS_CRYPTO.DES_CBC_PKCS5, l_key);

– Use bind variables to safely insert data and prevent SQL injection INSERT INTO users (user_id, user_name, password) VALUES (p_user_id, p_user_name, l_encrypted_password);

– Log the operation INSERT INTO user_logs (log_id, log_message, log_date) VALUES (user_logs_seq.NEXTVAL, 'User' || p_user_name || ' inserted.', SYSDATE);

COMMIT; EXCEPTION WHEN OTHERS THEN – Log any error for debugging while ensuring security INSERT INTO error_logs (log_id, error_message, error_date) VALUES (error_logs_seq.NEXTVAL, SQLERRM, SYSDATE); RAISE; – Reraise the exception to propagate the error END; / AUTHID CURRENT_USER Clause:

The AUTHID CURRENT_USER clause ensures that the procedure executes with the privileges of the invoker, following the least privilege principle. This minimizes the risk of unauthorized data access. Bind Variables to Prevent SQL Injection: Bind variables (p_user_id, p_user_name, and l_encrypted_password) are used in the INSERT statement to prevent SQL injection attacks by not allowing the dynamic concatenation of user input into SQL statements. Encrypt Sensitive Data: The DBMS_CRYPTO.ENCRYPT function encrypts the user's password before inserting it into the database. This ensures that sensitive information is not stored in plain text. Error Handling and Logging: The EXCEPTION block catches all errors, logs them securely, and re-raises the exception to allow the caller to handle it further. This provides both security and transparency for debugging.

Q - 17) Write a program Fibonacci Series in Pl/Sql

DECLARE n NUMBER := 10; a NUMBER := 0; b NUMBER := 1; temp NUMBER; BEGIN DBMS_OUTPUT.PUT_LINE(a); DBMS_OUTPUT.PUT_LINE(b); FOR i IN 3..n LOOP temp := a + b; DBMS_OUTPUT.PUT_LINE(temp); a := b; b := temp; END LOOP; END; /

Q - 18) Program with Ref Cursors

—–A program that uses a ref cursor to return a result set to the calling environment.

DECLARE TYPE ref_cursor_type IS REF CURSOR; emp_cursor ref_cursor_type; emp_record employees%ROWTYPE; BEGIN OPEN emp_cursor FOR SELECT * FROM employees WHERE department_id = 10;

LOOP FETCH emp_cursor INTO emp_record; EXIT WHEN emp_cursor%NOTFOUND; DBMS_OUTPUT.PUT_LINE('Employee Name:' || emp_record.first_name || ', Salary:' || emp_record.salary); END LOOP;

```
CLOSE emp_cursor;
END;
```

/

Q - 19) I will create a comprehensive example that touches on many of the key topics, such as: Basic PL/SQL Block Structure Variables and Data Types Control Structures (IF, LOOP, WHILE, FOR) Cursors (Implicit and Explicit) Exception Handling Procedures and Functions Triggers Packages Records and Collections Dynamic SQL

– Step 1: Create a sample table for demonstration CREATE TABLE employees (employee_id NUMBER PRIMARY KEY, first_name VARCHAR2(50), last_name VARCHAR2(50), department_id NUMBER, salary NUMBER);

– Step 2: Insert some sample data INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (1, 'John', 'Doe', 10, 5000); INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (2, 'Jane', 'Smith', 20, 6000); INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (3, 'Alice', 'Brown', 10, 7000); COMMIT;

– Step 3: Create a package to handle employee operations CREATE OR REPLACE PACKAGE employee_pkg IS – Function to calculate bonus FUNCTION calculate_bonus(p_salary NUMBER) RETURN NUMBER;

– Procedure to display employee details PROCEDURE display_employee_details;

– Exception for employee not found employee_not_found EXCEPTION; END employee_pkg; /

CREATE OR REPLACE PACKAGE BODY employee_pkg IS – Function to calculate bonus FUNCTION calculate_bonus(p_salary NUMBER) RETURN NUMBER IS BEGIN RETURN p_salary * 0.1; END calculate_bonus;

– Procedure to display employee details PROCEDURE display_employee_details IS CURSOR emp_cursor IS SELECT * FROM employees; emp_record employees%ROWTYPE; BEGIN OPEN emp_cursor; LOOP FETCH emp_cursor INTO emp_record; EXIT WHEN emp_cursor %NOTFOUND; DBMS_OUTPUT.PUT_LINE('Employee:' || emp_record.first_name || ' ' || emp_record.last_name || ', Salary:' || emp_record.salary || ', Bonus:' || calculate_bonus(emp_record.salary)); END LOOP; CLOSE emp_cursor; END display_employee_details; END employee_pkg; /

– Step 4: Create a trigger to log salary changes CREATE OR REPLACE TRIGGER trg_salary_change AFTER UPDATE OF salary ON employees FOR EACH ROW BEGIN IF :OLD.salary != :NEW.salary THEN DBMS_OUTPUT.PUT_LINE('Salary changed for Employee ID' || :NEW.employee_id || ' from ' || :OLD.salary || ' to ' || :NEW.salary); END IF; END trg_salary_change; /

– Step 5: PL/SQL block to test the package and exception handling DECLARE v_bonus NUMBER; emp_count NUMBER; BEGIN – Using the package function SELECT employee_pkg.calculate_bonus(7000) INTO v_bonus FROM DUAL; DBMS_OUTPUT.PUT_LINE('Calculated Bonus:' || v_bonus);

– Using the package procedure employee_pkg.display_employee_details;

– Handling exceptions SELECT COUNT(*) INTO emp_count FROM employees WHERE employee_id = 999; IF emp_count = 0 THEN RAISE employee_pkg.employee_not_found; END IF;

EXCEPTION WHEN employee_pkg.employee_not_found THEN DBMS_OUTPUT.PUT_LINE('Employee not found.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An unexpected error occurred:' || SQLERRM); END; /

– Step 6: Execute a dynamic SQL statement DECLARE v_sql VARCHAR2(200); v_count NUMBER; BEGIN v_sql := 'SELECT COUNT(*) FROM employees WHERE salary > :salary';

EXECUTE IMMEDIATE v_sql INTO v_count USING 5000;
DBMS_OUTPUT.PUT_LINE('Number of employees with salary above 5000:' || v_count);
END; /

Explanation of the Program Creating a Sample Table and Inserting Data: Sets up a basic employees table with sample data. Package Definition and Body: employee_pkg is a package that contains a function calculate_bonus and a procedure display_employee_details. The function calculates a 10% bonus based on the salary. The procedure displays employee details using an explicit cursor. Trigger: trg_salary_change is an after-update trigger that logs salary changes. PL/SQL Block for Testing: Tests the package's function and procedure. Demonstrates exception handling by raising a custom exception if an employee is not found. Dynamic SQL: Shows how to use dynamic SQL to execute a query with a bind variable. Basic PL/SQL Block Structure: Using DECLARE, BEGIN, EXCEPTION, and END. Control Structures: IF statement and loop control (LOOP). Cursors: Explicit cursor (emp_cursor) for fetching employee details. Exception Handling: Custom exceptions and the WHEN OTHERS clause. Procedures and Functions: Defined within the package (employee_pkg). Triggers: Capturing events (salary changes) with AFTER UPDATE triggers. Packages: Encapsulation of related procedures and functions. Dynamic SQL: Using EXECUTE IMMEDIATE for runtime SQL execution. Records and Collections: Using %ROWTYPE to define a record type that matches the structure of a table.

Q - 20) PL/SQL Program to Create a Table, Insert, and Update Values

BEGIN EXECUTE IMMEDIATE 'DROP TABLE employees';—WE CAN NOT HAVE SAME TABLE_NAME(S) IN SAME DB EXCEPTION WHEN OTHERS THEN NULL; – Ignore if the table does not exist END; /

——————––First, you need to create a table in your database.

BEGIN EXECUTE IMMEDIATE ' CREATE TABLE employees (employee_id NUMBER PRIMARY KEY, first_name VARCHAR2(50), last_name VARCHAR2(50), department_id NUMBER, salary NUMBER)'; END; /

————————Next, insert some initial values into the table. BEGIN EXECUTE IMMEDIATE ' INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (1, ''John'',''Doe'', 10, 5000)';

EXECUTE IMMEDIATE ' INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (2, ''Jane'',''Smith'', 20, 6000)';

EXECUTE IMMEDIATE ' INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (3, ''Alice'',''Brown'', 10, 7000)';

COMMIT; END; /

—————Finally, update the values in the table using a PL/SQL block

DECLARE v_employee_id NUMBER := 2; v_new_salary NUMBER := 6500; BEGIN – Update salary for a specific employee UPDATE employees SET salary = v_new_salary WHERE employee_id = v_employee_id;

COMMIT;

DBMS_OUTPUT.PUT_LINE('Updated salary for employee_id' || v_employee_id || ' to ' || v_new_salary); EXCEPTION WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred:' || SQLERRM); END; /

Q - 21) Differences between MySQL, Oracle SQL, and PL/SQL:

1. MySQL Overview:

Type: Relational Database Management System (RDBMS) Usage: Widely used in web applications, open-source projects, and small to medium-sized enterprises. Key Features:

SQL Language: MySQL uses standard SQL (Structured Query Language) for querying and managing data. Storage Engines: Supports multiple storage engines, including with varying features and performance characteristics. Transactions: Supports ACID (Atomicity, Consistency, Isolation, Durability) transactions with. Replication: Provides various replication methods including master-slave and group replication. Performance: Known for its speed and efficiency in read-heavy workloads. Limitations:

Advanced Features: May lack some advanced features found in other RDBMS systems like Oracle, such as advanced analytical functions and sophisticated transaction management. Stored Procedures: While MySQL supports stored procedures and functions, its capabilities are less advanced compared to Oracle PL/SQL. 2. Oracle SQL Overview:

Type: SQL dialect used within Oracle Database Usage: Enterprise-level applications, large-scale databases, and mission-critical systems. Key Features:

SQL Language: Oracle SQL is an extension of the standard SQL with additional features specific to Oracle databases. Advanced Features: Includes advanced analytical functions, hierarchical queries, and robust data types. Performance Tuning: Includes sophisticated tools and features for performance optimization and tuning. Security: Advanced security features such as Virtual Private Database (VPD) and advanced encryption. Limitations:

Complexity: Some advanced features can add complexity to database management and require specialized knowledge. 3. PL/SQL Overview:

Type: Procedural Language extension for SQL Vendor: Oracle Corporation Usage: Used within Oracle Database for creating complex scripts, stored procedures, functions, triggers, and packages. Key Features:

Procedural Language: Extends SQL by adding procedural constructs such as variables, loops, and conditionals. Stored Procedures and Functions: Allows for the creation of reusable code blocks that can be executed within the database. Error Handling: Provides robust exception handling mechanisms for managing errors and exceptions. Triggers:

Supports creating triggers that automatically execute in response to specific database events. Limitations:

Oracle-Specific: PL/SQL is specific to Oracle databases and cannot be used with other RDBMS systems like MySQL. Summary of Differences MySQL: An open-source RDBMS using standard SQL. Suitable for many web applications and smaller projects. Less advanced in some features compared to Oracle. Oracle SQL: A SQL dialect specific to Oracle Database with advanced features for enterprise-level needs. Part of the broader Oracle ecosystem and used for high-performance, high-security applications. PL/SQL: A procedural extension to SQL used within Oracle databases for creating stored procedures, functions, triggers, and packages. It adds procedural programming capabilities to SQL, enhancing Oracle's database management capabilities. In essence:

MySQL and Oracle SQL are both RDBMS systems, but Oracle SQL is specific to Oracle's database system with advanced features. PL/SQL is a procedural language used in Oracle databases to extend SQL with programming constructs.

Q - 22) The difference between a procedure and a function in PL/SQL

Procedure Definition:

A procedure is a PL/SQL block that performs a specific task or set of tasks. It does not return a value. Key Characteristics:

No Return Value: Procedures do not return a value directly. They perform operations such as inserting, updating, or deleting data, or other tasks, but they do not return a result. Parameters: Procedures can have IN, OUT, and IN OUT parameters, allowing them to accept input values and return output values or modify parameters. Usage: Procedures are used for executing a sequence of statements that perform a particular action. They are often used for operations that do not require a return value. CREATE OR REPLACE PROCEDURE update_salary

(p_employee_id IN NUMBER, p_new_salary IN NUMBER)

AS BEGIN UPDATE employees SET salary = p_new_salary WHERE employee_id = p_employee_id; COMMIT; DBMS_OUTPUT.PUT_LINE('Salary updated successfully.'); END; / Usage in SQL:

Procedures are called using EXECUTE or CALL statements in SQL*Plus or other PL/SQL environments. Function Definition:

A function is a PL/SQL block that performs a specific task and returns a single value. Functions can be used in SQL statements, expressions, and other PL/SQL blocks. Key Characteristics:

Returns a Value: Functions must return a value of a specific data type. They are used when you need to compute and return a value based on input parameters. Parameters: Functions can have IN parameters (input values) and must return a value. They cannot have OUT or IN OUT parameters. Usage: Functions are often used to compute and return a value, which can be directly used in SQL queries, expressions, or other PL/SQL code. CREATE OR REPLACE FUNCTION get_salary

(p_employee_id IN NUMBER)

RETURN NUMBER IS v_salary NUMBER; BEGIN SELECT salary INTO v_salary FROM employees WHERE employee_id = p_employee_id; RETURN v_salary; EXCEPTION WHEN NO_DATA_FOUND THEN RETURN NULL; END; /

Functions can be used in SQL queries and expression

SELECT get_salary(101)

FROM dual;

procedures are used to perform tasks and operations without returning a value, while functions are designed to return a value and can be used in SQL statements and expressions.

Q - 23) Write a PL/SQL code to find whether a given string is palindrome or not.

—-A palindrome string is a sequence of characters that reads the same backward as forward, ignoring spaces, punctuation, and capitalization.

——Examples of Palindrome Strings

———-Single words: “madam,” “racecar,” “level”

DECLARE input_string VARCHAR2(50); – The input string to check reversed_string VARCHAR2(50); – To store the reversed string BEGIN – Assign your input string here input_string := ‘madam’; – You can change this to any string

– Reverse the input string using SQL functions reversed_string := REVERSE(input_string);

– Check if the original string is equal to the reversed string IF input_string = reversed_string THEN DBMS_OUTPUT.PUT_LINE(input_string || ’ is a palindrome.’); ELSE DBMS_OUTPUT.PUT_LINE(input_string || ’ is not a palindrome.’); END IF; END; /

Ex:-2

DECLARE

– Declared variables string, letter, reverse_string where string is the original string. string VARCHAR2(10) := ‘abccba’; letter VARCHAR2(20); reverse_string VARCHAR2(10); BEGIN

FOR i IN REVERSE 1..LENGTH(string) LOOP letter := SUBSTR(string, i, 1); – concatenate letter to reverse_string variable reverse_string := reverse_string ||“||letter; END LOOP; IF reverse_string = string THEN dbms_output.Put_line(reverse_string||“||’ is palindrome’); ELSE

dbms_output.Put_line(reverse_string ||” ||’ is not palindrome’); END IF;

END;

/

Q - 24) Write PL/SQL program to convert each digit of a given number into its corresponding word format

DECLARE

```
 -- declare necessary variables

 -- num represents the given number

 -- number_to_word represents the word format of the number

 -- str, len and digit are the intermediate variables used for program
execution

     num INTEGER;

     number_to_word VARCHAR2(100);

    digit_str VARCHAR2(100);

   len INTEGER;

   digit INTEGER;
```

BEGIN

```
num := 123456;

len := LENGTH(num);

dbms_output.PUT_LINE('Input: ' ||num);
```

– Iterate through the number one by one FOR i IN 1..len LOOP

```
digit := SUBSTR(num, i, 1);
```

– Using DECODE, get the str representation of the digit

```
  SELECT Decode

 (

 digit, 0, 'Zero ',

          1, 'One ',
            2, 'Two ',

            3, 'Three ',

            4, 'Four ',

            5, 'Five ',

            6, 'Six ',

            7, 'Seven ',

            8, 'Eight ',

            9, 'Nine '

        )
```

INTO digit_str

FROM dual; – Append the str representation of digit to final result.

number_to_word := number_to_word || digit_str; END LOOP;

dbms_output.PUT_LINE('Output:' ||number_to_word);

END;

/

Input: 12345 Output: One Two Three Four Five

example 2

DECLARE input_number NUMBER := 12345; – Input number to convert input_string VARCHAR2(10); – To hold the input number as a string digit_word VARCHAR2(10); – To hold the final output in word format current_digit CHAR(1); – To hold the current digit being processed BEGIN – Convert the input number to a string input_string := TO_CHAR(input_number);

– Initialize the output string digit_word := ";

– Loop through each character (digit) of the input string FOR i IN 1..LENGTH(input_string) LOOP current_digit := SUBSTR(input_string, i, 1);

```
   -- Use a CASE statement to convert each digit to its word
equivalent
   CASE current_digit
       WHEN '0' THEN digit_word := digit_word || 'Zero '
       WHEN '1' THEN digit_word := digit_word || 'One '
       WHEN '2' THEN digit_word := digit_word || 'Two '
```

```
        WHEN '3' THEN digit_word := digit_word || 'Three '
        WHEN '4' THEN digit_word := digit_word || 'Four '
        WHEN '5' THEN digit_word := digit_word || 'Five '
        WHEN '6' THEN digit_word := digit_word || 'Six '
        WHEN '7' THEN digit_word := digit_word || 'Seven '
        WHEN '8' THEN digit_word := digit_word || 'Eight '
        WHEN '9' THEN digit_word := digit_word || 'Nine '
    END CASE;
```

END LOOP;

– Print the result DBMS_OUTPUT.PUT_LINE('The number in words is:' || digit_word); END; /

Q - 25) Program to Swap Two Numbers Without Using a Third Variable

DECLARE a NUMBER := 5; b NUMBER := 10; BEGIN a := a + b; – a becomes 15 b := a - b; – b becomes 5 a := a - b; – a becomes 10

DBMS_OUTPUT.PUT_LINE('After swapping: a =' || a || ', b =' || b); END; /

Q - 26) What is the difference between IN, OUT, and IN OUT parameters in PL/SQL

IN: Passes a value into a procedure or function (read-only). OUT: Returns a value from a procedure or function (write-only). IN OUT: Passes a value into and returns a value from a procedure or function (read-write).

Q - 27) PL/SQL Function to Find the Area of a Square

– Create a function to calculate the area of a square

CREATE OR REPLACE FUNCTION calculate_square_area(side_length NUMBER) RETURN NUMBER IS area NUMBER; – Variable to store the area of the square BEGIN – Calculate the area of the square (side_length * side_length) area := side_length * side_length;

– Return the calculated area RETURN area; END; /

DECLARE side NUMBER := 5; – Example side length of the square area_of_square NUMBER; – Variable to store the result BEGIN – Call the function to calculate the area area_of_square := calculate_square_area(side);

– Display the result DBMS_OUTPUT.PUT_LINE('The area of the square with side length' || side || ' is: ' || area_of_square); END; /

The area of the square with side length 5 is: 25

Q - 28) pattern program in pl/sql

- ****

PLSQL

BEGIN FOR i IN 1..5 LOOP FOR j IN 1..i LOOP
dbms_output.put('*'); END LOOP;
dbms_output.put_line(''); END LOOP; END; /

SQL QUERY

- **** SELECT RPAD('', *LEVEL,* '') "*" FROM dual CONNECT BY LEVEL <= 5;

Q - 29) Print the Pattern in PL/SQL

1 11 111 1111 11111

DECLARE V NUMBER := 1; BEGIN FOR i IN 1..5 LOOP FOR j IN 1..i LOOP
dbms_output.put(V); END LOOP; dbms_output.put_line(''); END LOOP; END; /

Q - 30) Show some predefined exceptions.

DUP_VAL_ON_INDEX ZERO_DIVIDE NO_DATA_FOUND TOO_MANY_ROWS
CURSOR_ALREADY_OPEN INVALID_NUMBER INVALID_CURSOR PROGRAM_ERROR
TIMEOUT _ON_RESOURCE STORAGE_ERROR LOGON_DENIED VALUE_ERROR,,,,,,,,etc…

Q - 31) PL/SQL CASE Statement

statement uses a selector instead of using a combination of multiple Boolean expressions.

SYNTAX

[<>] CASE [TRUE | selector] WHEN expression1 THEN sequence_of_statements1; WHEN expression2 THEN sequence_of_statements2; … WHEN expressionN THEN sequence_of_statementsN; [ELSE sequence_of_statementsN+1;] END CASE [label_name];

————-SET SERVEROUTPUT ON

SET LINESIZE 10000;

SET PAGESIZE 10000;

DECLARE n_pct employees.commission_pct%TYPE; v_eval varchar2(10); n_emp_id employees.employee_id%TYPE := 145; BEGIN ———- get commission percentage SELECT commission_pct INTO n_pct FROM employees WHERE employee_id = n_emp_id;

——- evalutate commission percentage CASE n_pct

WHEN 0 THEN v_eval := 'N/A';

WHEN 0.1 THEN v_eval := 'Low';

WHEN 0.4 THEN v_eval := 'High';

ELSE v_eval := ‘Fair’;

END CASE; – print commission evaluation DBMS_OUTPUT.PUT_LINE(‘Employee’ || n_emp_id || ’ commission ’ || TO_CHAR(n_pct) ||’ which is ’ || v_eval); END; /

Q - 32) How to write comments in PL/SQL

Comments help us to understand the logic of the code. There are two ways to write comments in Pl/SQL:

Using (–): This is used to write single-line comments. Using (/…./): This is used to write multi-line comments.

Q - 33) Differentiate between compile-time errors and run-time errors?

A PL/SQL compiler can readily discover a compile-time error. Take, for example, poor spelling. An exception-handling section in a PL/SQL block is used to handle a run-time error. Consider the SELECT INTO query, which returns no rows.

Q - 34) Why do we use SYS.ALL_DEPENDENCIES

All dependencies between procedures, packages, triggers, and functions that are accessible to the current user are described in SYS.ALL DEPENDENCIES. Name, dependency type, type, referenced owner, and other columns are returned.

Q - 35) What are autonomous transactions in PL SQL

The independent transactions in a vast transaction are known as autonomous transactions. They are useful when there is a requirement for logging or changes in the track that needs to be made without the result of the main transition being affected. This helps to main data consistency. You can use it for performing operations that are separated from the main transactions. They must be rolled back independently or committed.

Q - 36) Create a stored procedure that takes an employee ID as input and gives them a 10% raise(hike)

CREATE OR REPLACE PROCEDURE give_raise(p_emp_id IN NUMBER) IS BEGIN

```
UPDATE employees
SET salary = salary * 1.1
WHERE employee_id = p_emp_id;
IF SQL%ROWCOUNT = 0 THEN
```

RAISE_APPLICATION_ERROR(-20001, ‘No employee found with ID’ || p_emp_id); ELSE

```
 COMMIT;
DBMS_OUTPUT.PUT_LINE('Raise given successfully to employee ' ||
p_emp_id);
```

END IF; EXCEPTION WHEN OTHERS THEN ROLLBACK; DBMS_OUTPUT.PUT_LINE('Error:' || SQLERRM); END give_raise; END;

/

Q - 37) What are the basic elements of PL SQL

The basic elements of PL/SQL include variables, loops and conditional statements, exception handling, cursors for database interaction, and procedures/functions forming its fundamental components for organized and reusable code

Q - 38) Which of the following is a correct PL/SQL block structure

A) DECLARE, BEGIN, END

B) BEGIN, EXCEPTION, END

C) BEGIN, DECLARE, END

D) DECLARE, EXCEPTION, END

Answer: A) DECLARE, BEGIN, END

Q - 39) Which of the following is used to store the entire row of a table?

A) %TYPE

B) %ROWTYPE

C) ROWID

D) TABLE

Answer: B) %ROWTYPE

Q - 40) What happens when you issue a COMMIT in PL/SQL?

A) CURSOR ... IS

B) DECLARE CURSOR

C) OPEN CURSOR

D) SET CURSOR

Answer: A) CURSOR ... IS

Q - 41) What is the purpose of EXCEPTION_INIT in PL/SQL?

A) To handle predefined exceptions

B) To associate an exception with an error code

C) To declare user-defined exceptions

D) To exit the exception block

Answer: B) To associate an exception with an error code

Q - 42) Which PL/SQL collection type stores key-value pairs?

A) Nested Table

B) VARRAY

C) Associative Array

D) RECORD

Answer: C) Associative Array

Q - 43) How do you raise a user-defined exception in PL/SQL?

A) RAISE_APPLICATION_ERROR

B) EXCEPTION RAISE

C) RAISE

D) THROW

Answer: C) RAISE

Q - 44) What is the maximum size of VARCHAR2 in PL/SQL?

A) 1000

B) 2000

C) 4000

D) 8000

Answer: C) 4000

Q - 45) What is the purpose of SAVEPOINT in PL/SQL?

A) To commit a transaction

B) To rollback to a point in a transaction

C) To restart the transaction

D) To define the end of a block

Answer: B) To rollback to a point in a transaction

Q - 46) Implicit Cursor, Explicit Cursor In Pl/Sql

Implicit Cursor: While invoking any of the commands SELECT INTO, INSERT, DELETE, or UPDATE, Oracle implicitly creates a cursor. Oracle handles the cursor execution cycle internally and returns the cursor's information and status using the cursor attributes ROWCOUNT, ISOPEN, FOUND, and NOTFOUND.

Explicit Cursor: This cursor is a SELECT statement declared in the declaration block. The programmer must control the cursors' execution cycle, which begins with OPEN and ends with FETCH and CLOSE. Oracle defines the SQL statement execution cycle as well as the cursor that is associated with it.

Q - 47) What is wrong in the following code?

```
DECLARE
    c_id := 1;
     c_name  customers.name%type;
     c_addr customers.address%type;
BEGIN
       SELECT name, address INTO c_name, c_addr
       FROM customers
       WHERE id = c_id;
       END;
       /
```

a)You cannot use the SELECT INTO statement of SQL to assign values to PL/SQL variables

b)The SELECT INTO statement here is wrong. It should be: SELECT c_name, c_address INTO name, addr

c) The WHERE statement is wrong. It should be: WHERE id := c_id;

d) The variable c_id should be declared as a type-compatible variable as -c_id customers.id%type := 1;

ANSWER) d)The variable c_id should be declared as a type-compatible variable as -c_id customers.id%type := 1;

Q - 48) What would be the output of the following code?

```
DECLARE a number;

b number;

c number;

FUNCTION fx(x IN number, y IN number) RETURN number IS

z number; BEGIN IF x > 2*y THEN z:= x;

ELSE

      z:= 2*y;

END IF;

        RETURN z;

END;

BEGIN a:= 23; b:= 47;

       c := fx(a, b);

       dbms_output.put_line(c);

END;

/
```

A - 46

B - 47

C - 94

D - 23

Answer) C - 94

Q - 49) This set of PL/SQL Multiple Choice Questions & Answers (MCQs) focuses on "PL/SQL Basics".

. What is the full form of PL in PL/SQL? a) Programming Language b) Procedural Language c) Programming Logic d) Procedural Logic

Answer: b Explanation: PL in PL/SQL stands for Procedural Language. It is a block structured language that enables developers to combine the power of SQL with procedural language.

PL/SQL can execute a number of queries in one single block using a single command. a) True b) False

Answer: a Explanation: Yes, PL/SQL can execute a number of queries in one single block using a single command. All the statements in the block are passed to the oracle engine all at once to increase the processing speed and reduce traffic.

Which of the following feature is provided by PL/SQL to handle the exceptions occurred in the PL/SQL block? a) Error Handling Block b) Error Removing Block c) Exception Handling Block d) Exception Removing Block

Answer: c Explanation: The Exception handling block is the feature that helps the user to find and remove any exceptions that are found in the PL/SQL block.

Q - 50) Which of the following is the correct is the correct syntax for exception handling?

WHEN exception1 THEN { exception1_handling_statement};

WHEN exception2 THEN { exception2_handling_statement};

……..

WHEN others THEN{ Exception_handling_statement};

WHEN exception1 exception1_handling_statement

WHEN exception2 exception2_handling_statement

……..

WHEN others Exception_handling_statement

WHEN exception1 THEN

{ exception1_handling_statement};

WHEN exception2 THEN

{ exception2_handling_statement};

……..

WHEN others THEN

{ exception_handling_statement};

WHEN exception1

```
 exception1_handling_statement
```

WHEN exception2

```
   exception2_handling_statement
```

……..

WHEN others Exception_handling_statement

Answer: b Explanation: The correct syntax for exception handling is –

WHEN exception1 THEN

```
  exception1_handling_statement
```

WHEN exception2 THEN

```
   exception2_handling_statement

   ........
```

WHEN others THEN

```
      Exception_handling_statement
```

Raising an exception can be performed under which of the following PL/SQL program block?

a) DECLARE
b) BEGIN
c) EXCEPTION
d) END

Answer: b Explanation: If we are manually/explicitly raising any exception then it should come under the BEGIN block of a PL/SQL program. It is done before any exception handling process and the EXCEPTION block.

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project