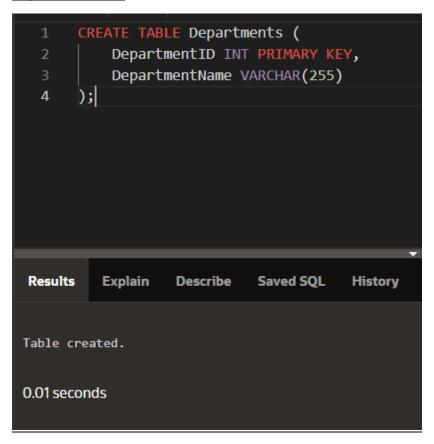
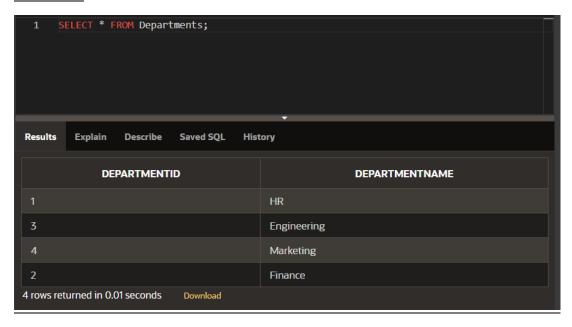
Question 2:

<u>Create Required tables and insert data to perform the below operations.</u>

Department table:



Insert data:



Employees Table:

```
CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY,
EmployeeName VARCHAR(255),
DepartmentID INT,
Salary DECIMAL(10, 2),
FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

7 );

Results Explain Describe Saved SQL History

Table created.

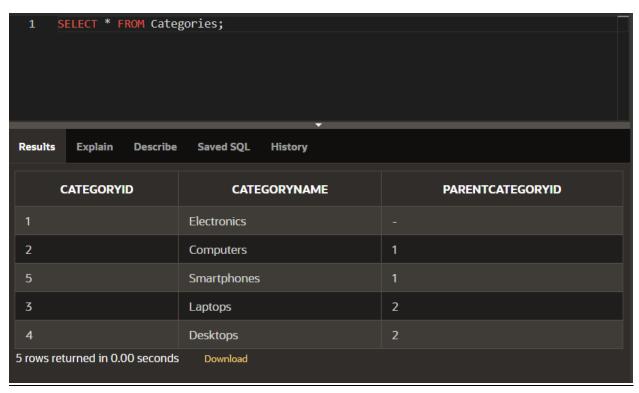
0.02 seconds
```

Insert data:



Categories Table:

Insert data:



And also create location customers and order tables.

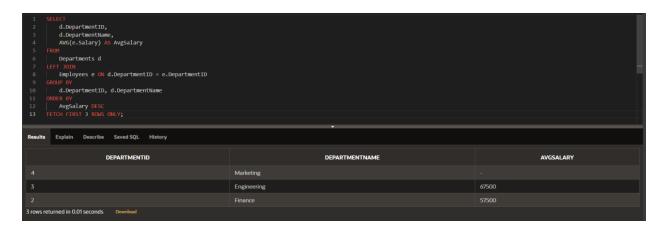
Question 1: Top 3 Departments with Highest Average Salary

Task:

1. Write a SQL query to find the top 3 departments with the highest average salary of employees. Ensure departments with no employees show an average salary of NULL.

Deliverables:

- 1. SQL query that retrieves DepartmentID, DepartmentName, and AvgSalary for the top 3 departments.
- 2. Explanation of how the query handles departments with no employees and calculates average salary.

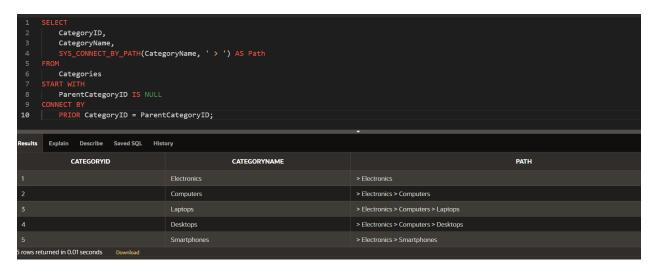


Question 2: Retrieving Hierarchical Category Paths

Task:

 Write a SQL query using recursive Common Table Expressions (CTE) to retrieve all categories along with their full hierarchical path (e.g., Category > Subcategory > Sub-subcategory).

- 1. SQL query that uses recursive CTE to fetch CategoryID, CategoryName, and hierarchical path.
- 2. Explanation of how the recursive CTE works to traverse the hierarchical data.



Question 3: Total Distinct Customers by Month

Task:

1. Design a SQL query to find the total number of distinct customers who made a purchase in each month of the current year. Ensure months with no customer activity show a count of 0.

Deliverables:

- 1. SQL query that retrieves MonthName and CustomerCount for each month.
- 2. Explanation of how the query ensures all months are included and handles zero customer counts.



Question 4: Finding Closest Locations

Task:

1. Write a SQL query to find the closest 5 locations to a given point specified by latitude and longitude. Use spatial functions or advanced mathematical calculations for proximity.

Deliverables:

- 1. SQL query that calculates the distance and retrieves LocationID, LocationName, Latitude, and Longitude for the closest 5 locations.
- 2. Explanation of the spatial or mathematical approach used to determine proximity.

```
| SELECT | LocationID, | LocationID, | LocationMane, | Latitude, | Longitude, | Longitude, | LocationMane, | L
```

Question 5: Optimizing Query for Orders Table

Task:

1. Write a SQL query to retrieve orders placed in the last 7 days from a large Orders table, sorted by order date in descending order.

- 1. SQL query optimized for performance, considering indexing, query rewriting, or other techniques.
- 2. Discussion of strategies used to optimize the query and improve performance

```
1 SELECT
2 OrderID,
3 CustomerID,
4 OrderDate,
5 TotalAmount
6 FROM
7 Orders
8 WHERE
9 OrderDate >= CURRENT_DATE - INTERVAL '7 days'
10 ORDER BY
11 OrderDate DESC;
```

Question 3:

PL/SQL Questions

Question 1: Handling Division Operation

Task:

 Write a PL/SQL block to perform a division operation where the divisor is obtained from user input. Handle the ZERO_DIVIDE exception gracefully with an appropriate error message.

Deliverables:

- 1. PL/SQL block that performs the division operation and handles exceptions.
- 2. Explanation of error handling strategies implemented.

```
1 DECLARE
2     dividend NUMBER := 100;
3     divisor NUMBER;
4     result NUMBER;
5     BEGIN
6     divisor := &divisor;
7     result := dividend / divisor;
8     DBMS_OUTPUT.PUT_LINE('Result: ' || result);
9     EXCEPTION
10     WHEN ZERO_DIVIDE THEN
11     DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
12     END;
13
```

Question 2: Updating Rows with FORALL

Task:

1. Use the FORALL statement to update multiple rows in the Employees table based on arrays of employee IDs and salary increments.

- 1. PL/SQL block that uses FORALL to update salaries efficiently.
- 2. Description of how FORALL improves performance for bulk updates.

```
DECLARE
TYPE emp_id_array IS TABLE OF employees.employee_id%TYPE;
TYPE salary_inc_array IS TABLE OF NUMBER;
emp_ids emp_id_array := emp_id_array(101, 102, 103);
salary_incs salary_inc_array := salary_inc_array(500, 1000, 1500);

BEGIN
FORALL i IN 1..emp_ids.COUNT
UPDATE employees
SET salary = salary + salary_incs(i)
WHERE employee_id = emp_ids(i);
DBMS_OUTPUT.PUT_LINE('Salaries updated successfully.');
END;
```

Question 3: Implementing Nested Table Procedure

Task:

1. Implement a PL/SQL procedure that accepts a department ID as input, retrieves employees belonging to the department, stores them in a nested table type, and returns this collection as an output parameter.

- 1. PL/SQL procedure with nested table implementation.
- 2. Explanation of how nested tables are utilized and returned as output.

Question 4: Using Cursor Variables and Dynamic SQL

Task:

 Write a PL/SQL block demonstrating the use of cursor variables (REF CURSOR) and dynamic SQL. Declare a cursor variable for querying EmployeeID, FirstName, and LastName based on a specified salary threshold.

Deliverables:

- 1. PL/SQL block that declares and uses cursor variables with dynamic SQL.
- 2. Explanation of how dynamic SQL is constructed and executed.

Question 5: Designing Pipelined Function for Sales Data

Task:

 Design a pipelined PL/SQL function get_sales_data that retrieves sales data for a given month and year. The function should return a table of records containing OrderID, CustomerID, and OrderAmount for orders placed in the specified month and year.

- 1. PL/SQL code for the pipelined function get_sales_data.
- 2. Explanation of how pipelined table functions improve data retrieval efficiency.

```
OR REPLACE TYPE sales_record AS OBJECT (
order_id NUMBER,
customer_id NUMBER,
order_amount NUMBER
);
               CREATE OR REPLACE TYPE sales_table AS TABLE OF sales_record;
               CREATE OR REPLACE FUNCTION get_sales_data (
p_month IN MAMBER,
p_year IN NAMBER
) RETURN sales_table PIPELINED IS
v_sales_rec_sales_record;
BEGGIN_
                              FROM sales
                                                      WHERE EXTRACT(MONTH FROM order_date) = p_month
AND EXTRACT(YEAR FROM order_date) = p_year)
                                                                               TYPE ref_cursor IS REF CURSOR;
     c_ref_cursor ref_cursor;
     v_employee_id employees.employee_id%TYPE;
     v_first_name employees.first_name%TYPE;
     v_last_name employees.last_name%TYPE;
     v_salary_threshold NUMBER := &salary_threshold;
     OPEN c_ref_cursor FOR
               'SELECT employee_id, first_name, last_name FROM employees WHERE salary > ' || v_salary_threshold;
     FETCH c_ref_cursor INTO v_employee_id, v_first_name, v_last_name;
     EXIT WHEN c_ref_cursor%NOTFOUND;
     DBMS_OUTPUT_PUT_LINE('Employee ID: ' || v_employee_id || ', Name: ' || v_first_name || ' ' || v_last_name);
END LOOP;
CLOSE c_ref_cursor;
```

Rubrics

Criteria	Description	Percentage
Conceptual Understanding	Demonstrates clear understanding of the problem domain (e.g., traffic flow management for ER Diagram, data retrieval and manipulation for SQL/PLSQL).	25%
Technical Accuracy	Accuracy in designing the ER Diagram or writing SQL/PLSQL queries, ensuring they meet requirements and handle edge cases effectively.	30%
Documentation and Clarity	Quality of documentation, including clarity of explanations, use of appropriate terminology, and organization of diagrams or code.	
Design and Solution Justification	Justification of design choices (e.g., normalization in ER Diagram, query optimization in SQL/PLSQL) with clear reasoning and considerations for scalability or efficiency.	20%