**CSA09: DATABASE MANAGEMENT SYSTEMS-ASSIGNMENT QUESTIONS**

**Due Date: 31 July 2024**

**Question 1:**

**ER Diagram Question: Traffic Flow Management System (TFMS)**

**Scenario**

You are tasked with designing an Entity-Relationship (ER) diagram for a Traffic Flow Management System (TFMS) used in a city to optimize traffic routes, manage intersections, and control traffic signals. The TFMS aims to enhance transportation efficiency by utilizing real-time data from sensors and historical traffic patterns.

The city administration has decided to implement a TFMS to address growing traffic congestion issues. The system will integrate real-time data from traffic sensors, cameras, and historical traffic patterns to provide intelligent traffic management solutions. Key functionalities include:

1. **Road Network Management**:
   * **Roads**: The city has a network of roads, each identified by a unique RoadID. Roads have attributes such as RoadName, Length (in meters), and SpeedLimit (in km/h).
2. **Intersection Control**:
   * **Intersections**: These are key points where roads meet and are crucial for traffic management. Each intersection is uniquely identified by IntersectionID and has attributes like IntersectionName and geographic Coordinates (Latitude, Longitude).
3. **Traffic Signal Management**:
   * **Traffic Signals**: Installed at intersections to regulate traffic flow. Each signal is identified by SignalID and has attributes such as SignalStatus (Green, Yellow, Red) indicating current state and Timer (countdown to next change).
4. **Real-Time Data Integration**:
   * **Traffic Data**: Real-time data collected from sensors includes TrafficDataID, Timestamp, Speed (average speed on the road), and CongestionLevel (degree of traffic congestion).
5. **Functionality Requirements**:
   * **Route Optimization**: Algorithms will be implemented to suggest optimal routes based on current traffic conditions.
   * **Traffic Signal Control**: Adaptive control algorithms will adjust signal timings dynamically based on real-time traffic flow and congestion data.
   * **Historical Analysis**: The system will store historical traffic data for analysis and planning future improvements.

**ER Diagram Design Requirements**

1. **Entities and Attributes**:
   * Clearly define entities (Roads, Intersections, Traffic Signals, Traffic Data) and their attributes based on the scenario provided.
   * Include primary keys (PK) and foreign keys (FK) where necessary to establish relationships between entities.
2. **Relationships**:
   * Illustrate relationships between entities (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).
   * Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).
3. **Normalization Considerations**:
   * Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

**Tasks**

**Task 1: Entity Identification and Attributes**

Identify and list the entities relevant to the TFMS based on the scenario provided (e.g., Roads, Intersections, Traffic Signals, Traffic Data).

Define attributes for each entity, ensuring clarity and completeness.

Op:

Entities:

1. Roads

- RoadID (PK)

- RoadName

- Length

- SpeedLimit

2. Intersections

- IntersectionID (PK)

- IntersectionName

- Latitude

- Longitude

3. Traffic Signals

- SignalID (PK)

- IntersectionID (FK)

- SignalStatus

- Timer

4. Traffic Data

- TrafficDataID (PK)

- Timestamp

- RoadID (FK)

- Speed

- CongestionLevel

**Task 2: Relationship Modeling**

Illustrate the relationships between entities in the ER diagram (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).

Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).

Op:’

Relationships:

1. Roads - Intersections (many-to-many)

- A road can have multiple intersections, and an intersection can be connected to multiple roads.

2. Intersections - Traffic Signals (one-to-many)

- An intersection can have multiple traffic signals, but a traffic signal is associated with only one intersection.

3. Roads - Traffic Data (one-to-many)

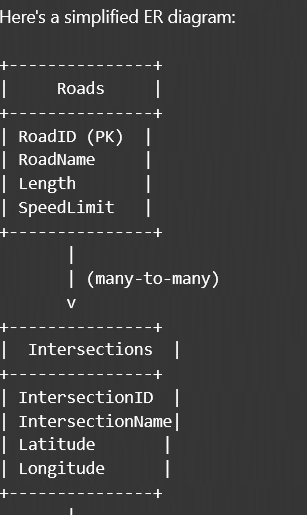
- A road can have multiple traffic data entries, but a traffic data entry is associated with only one road.

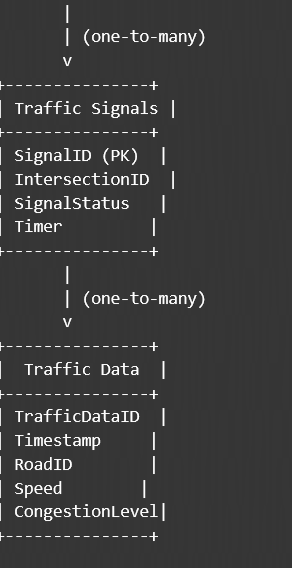
**Task 3: ER Diagram Design**

Draw the ER diagram for the TFMS, incorporating all identified entities, attributes, and relationships.

Label primary keys (PK) and foreign keys (FK) where applicable to establish relationships between entities.

Op:’





**Task 4: Justification and Normalization**

Justify your design choices, including considerations for scalability, real-time data processing, and efficient traffic management.

Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

Op:

The ER diagram is designed to support the TFMS functionalities, including route optimization, traffic signal control, and historical analysis. The relationships between entities ensure data integrity and minimize redundancy.

To ensure normalization, the ER diagram adheres to the following principles:

1. 1NF (First Normal Form): Each table cell contains a single value.

2. 2NF (Second Normal Form): Each non-key attribute depends on the entire primary key.

3. 3NF (Third Normal Form): If a table is in 2NF, and a non-key attribute depends on another non-key attribute, then it should be moved to a separate table.

**Deliverables**

1. **ER Diagram**: A well-drawn ER diagram that accurately reflects the structure and relationships of the TFMS database.
2. **Entity Definitions**: Clear definitions of entities and their attributes, supporting the ER diagram.
3. **Relationship Descriptions**: Detailed descriptions of relationships with cardinality and optionality constraints.
4. **Justification Document**: A document explaining design choices, normalization considerations, and how the ER diagram supports TFMS functionalities.

**Question 2:**

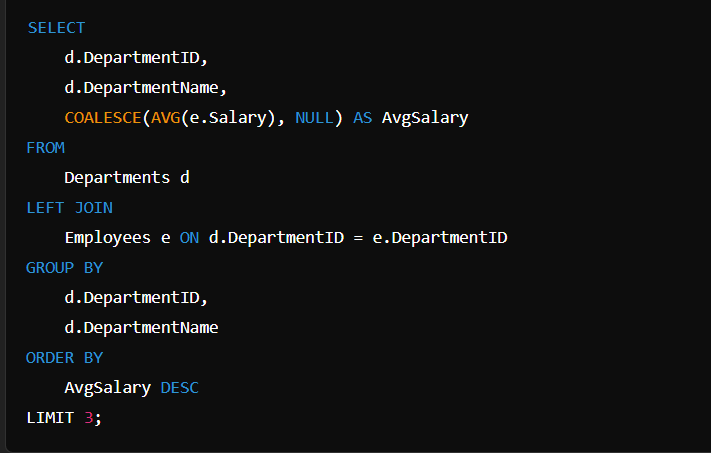
**Question 1: Top 3 Departments with Highest Average Salary**

**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.



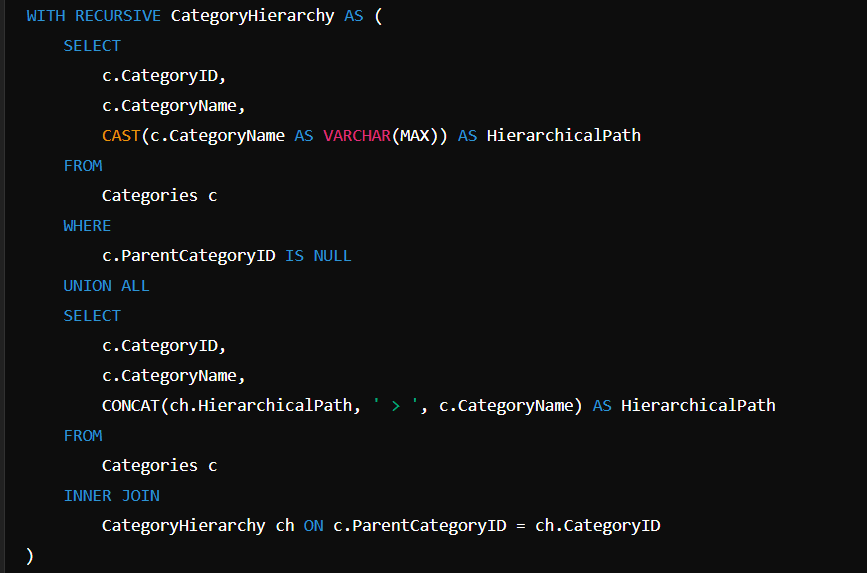
**Question 2: Retrieving Hierarchical Category Paths**

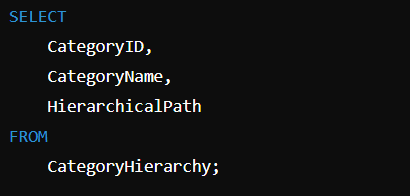
**Task:**

* + - 1. 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).

**Deliverables:**

* + - 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.





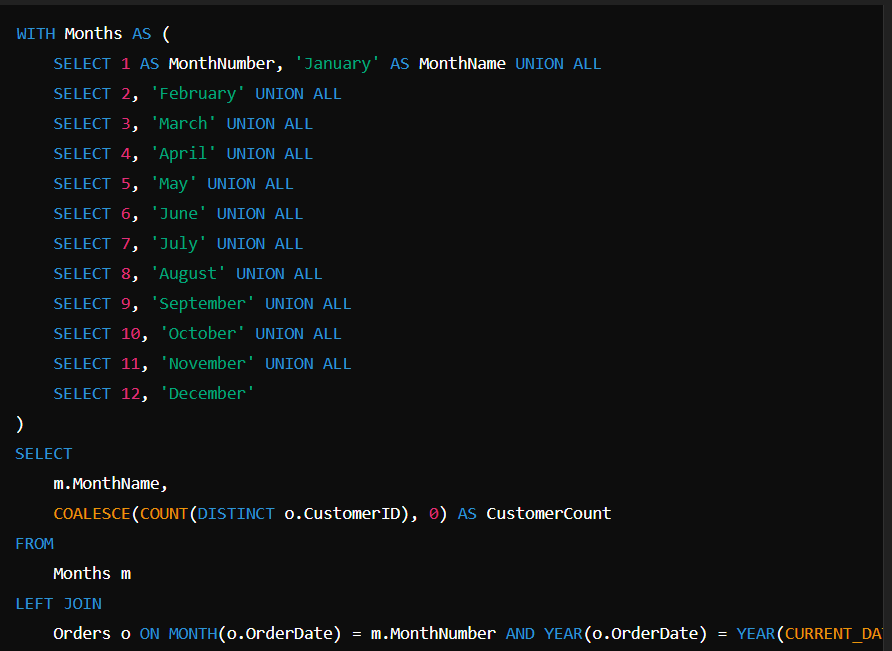
**Question 3: Total Distinct Customers by Month**

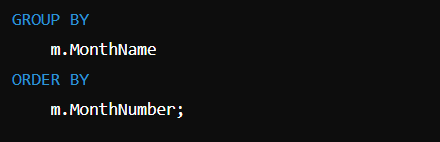
**Task:**

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.





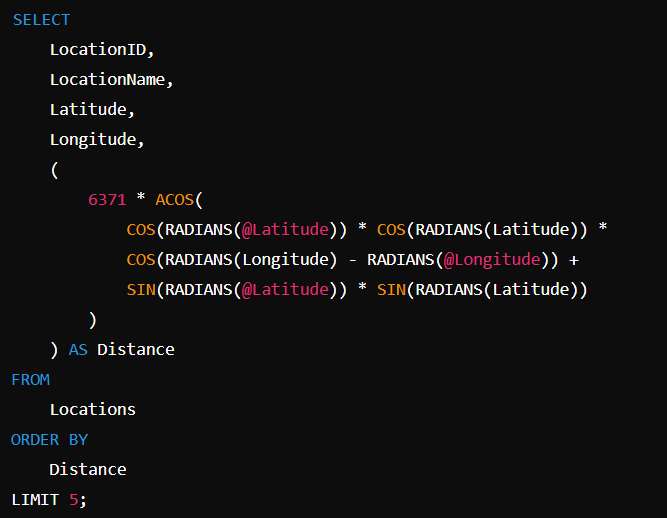
**Question 4: Finding Closest Locations**

**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.



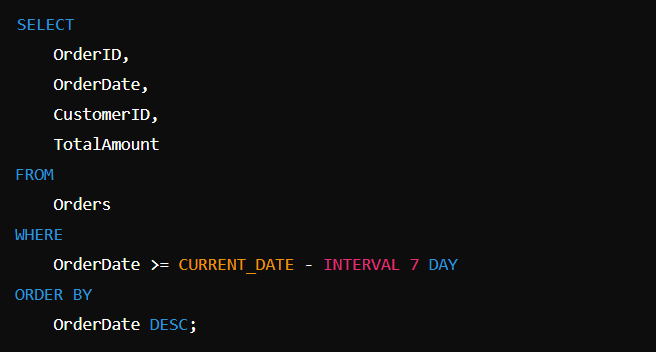
**Question 5: Optimizing Query for Orders Table**

**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.

**Deliverables:**

* 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.



**Question 3:**

**PL/SQL Questions**

**Question 1: Handling Division Operation**

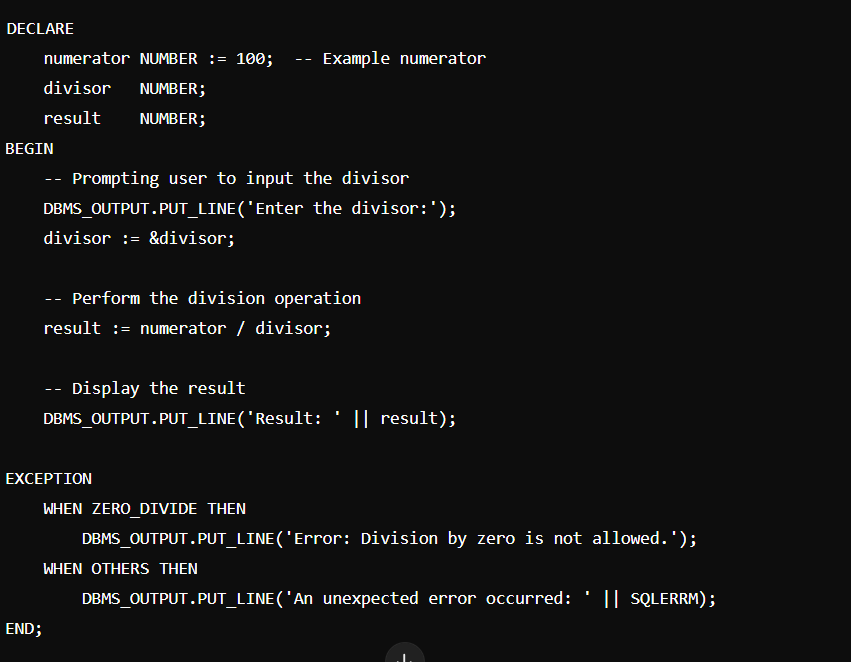
**Task:**

* + - 1. 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.

Op:



**Question 2: Updating Rows with FORALL**

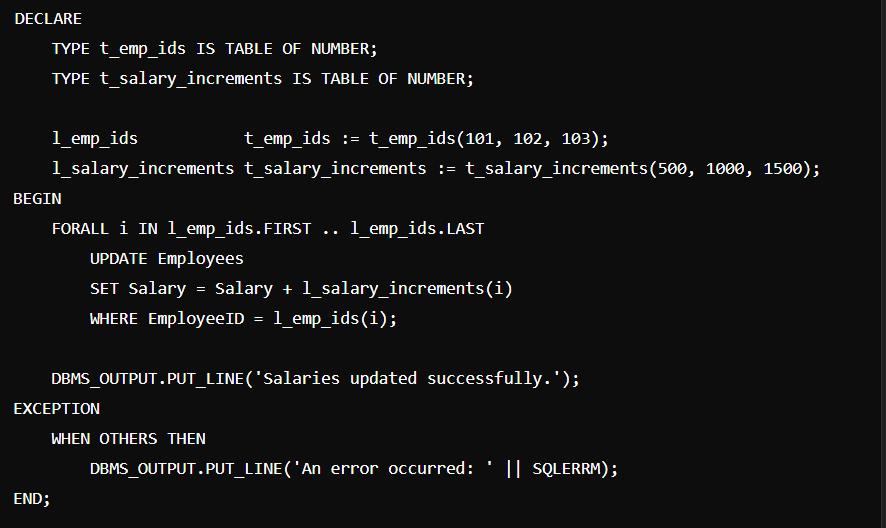
**Task:**

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

**Deliverables:**

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

Op:



**Question 3: Implementing Nested Table Procedure**

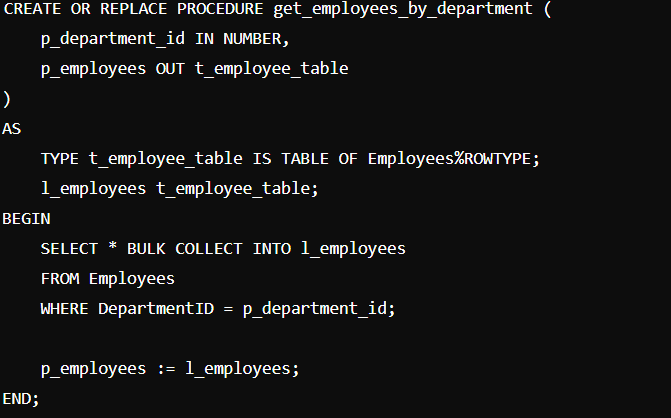
**Task:**

* + - 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.

**Deliverables:**

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

Op:



**Question 4: Using Cursor Variables and Dynamic SQL**

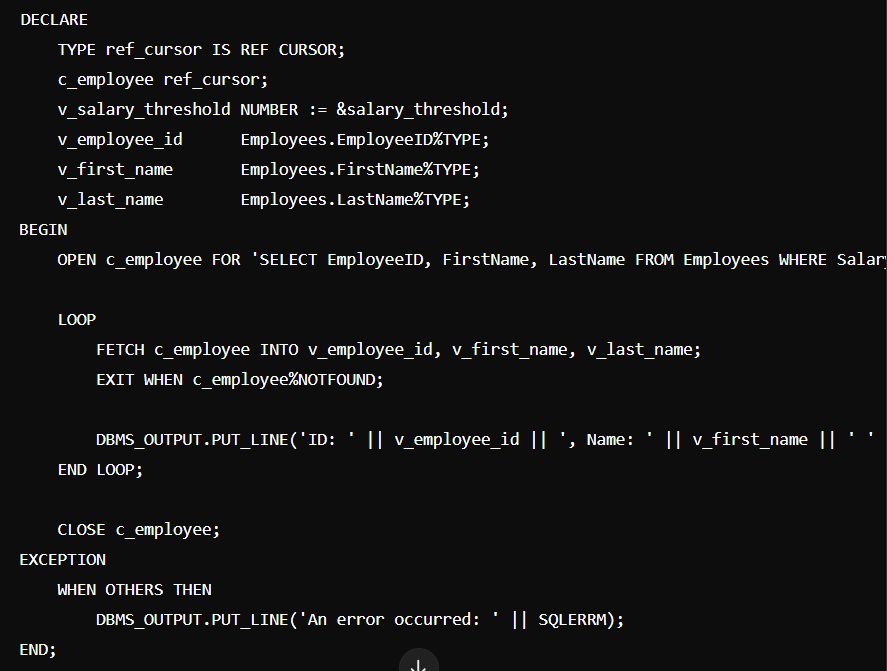
**Task:**

* + - 1. 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.

Op:



**Question 5: Designing Pipelined Function for Sales Data**

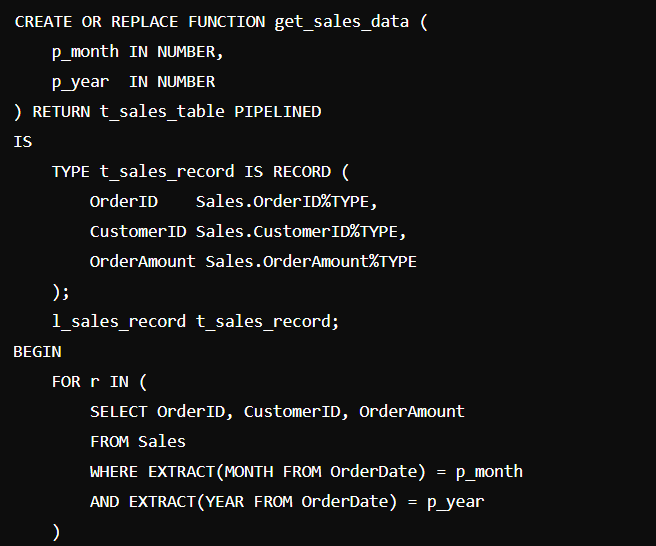
**Task:**

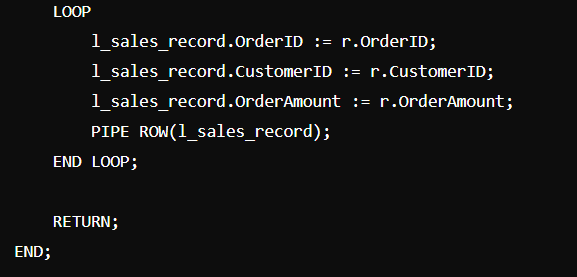
* + - 1. Design a pipelined PL/SQL function get\_sales\_data that retrieves sales data for a given month and year. The function should return a table of records containing OrderID, CustomerID, and OrderAmount for orders placed in the specified month and year.

**Deliverables:**

1. PL/SQL code for the pipelined function get\_sales\_data.
2. Explanation of how pipelined table functions improve data retrieval efficiency.

Op”;





**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. | 25% |
| **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% |