Text

Description automatically generated

**Activity based**

**Project 3 Report on**

**Database Management Systems**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice (C2P2)**

**By**

**Student Name: Atharva shevate**

**Manasvi deshpande**

**SRN No : 202201727**

**202201886**

**Roll No : 02&52**

**Div : E**

**Second Year Engineering**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024**

**Water Quality Monitoring System**

**Project Statement:**

**Create a database system to manage car rentals, including vehicle inventory, rental reservations, customer bookings, and rental agreements.**

**Mention about stored procedure, function, trigger and views commands applicable to given problem statement**

1. Stored Procedures:

- Stored procedures can be used to encapsulate and execute a series of SQL statements. In this scenario, a stored procedure named `GetAverageTemperatureByWaterBody` is defined to calculate the average temperature of a specified water body.

2. Functions:

- Functions can be utilized to perform specific calculations or operations within SQL queries. Although not explicitly mentioned in the provided script, functions could be created for tasks like calculating pollution index based on pollution levels or deriving certain statistical measures from the data.

3. Triggers:

- Triggers are database objects that automatically perform actions in response to specified database events. For instance, a trigger named `UpdateReports` is defined to automatically update the content of reports table after inserting new lab analysis results into the `Lab\_Analysis\_Results` table.

4. Views:

- Views provide a way to represent data from one or more tables in the database, often simplifying complex queries. In this scenario, views could be created to present summarized or aggregated data, such as a view that combines information from multiple tables to display comprehensive reports or analytics on water body conditions.

**Five lines description about stored procedure, function, trigger and views that you are using:**

1. Stored Procedure: Utilized for encapsulating a series of SQL statements to perform specific tasks, enhancing modularity and reusability within the database system.

2. Function: Employed to execute specific calculations or operations within SQL queries, facilitating complex data manipulation and computation.

3. Trigger: Automated database object that responds to specified database events, enabling automatic execution of actions such as updating tables based on data changes.

4. Views: Database objects that represent a subset of data from one or more tables, providing simplified and structured access to complex data sets for querying and reporting purposes.

5. Cursor: Control structure used to traverse through and manipulate result sets row by row, offering granular control over data processing and manipulation within stored procedures or functions.

**SQL statements related to project**:

1. Database Creation and Selection:

- Drops the existing database `waterDB` if it exists.

- Creates a new database named `waterDB`.

- Switches to the newly created `waterDB` database.

2. Table Creation:

- Defines the following tables:

- Water\_Bodies: Stores information about water bodies.

- Pollution\_Levels: Records pollution levels in water bodies.

- Chemical\_Attributes: Stores chemical attributes of water bodies.

- Biological\_Data: Records biological data of water bodies.

- Temperature\_Readings: Stores temperature readings of water bodies.

- Samples: Records samples collected from water bodies.

- Lab\_Analysis\_Results: Stores lab analysis results of samples.

- Environmental\_Scientists: Contains information about environmental scientists.

- Local\_Authorities: Stores details of local authorities.

- Reports: Stores generated reports related to water bodies.

3. Data Population:

- Inserts sample data into each table to populate them with initial records.

4. SQL Operations:

- Performs various SQL operations like `JOIN`, `UPDATE`, `UNION`, `INTERSECT`, `EXCEPT`, and subqueries.

- Demonstrates different types of joins including `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, and `SELF JOIN`.

5. Stored Procedures:

- Defines a stored procedure named `GetAverageTemperatureByWaterBody` to calculate the average temperature of a specified water body.

- Defines a stored procedure named `UpdatePollutionType` to update pollution types based on certain conditions using a cursor.

6. Trigger:

- Defines a trigger named `UpdateReports` which automatically updates the content of reports table after inserting new lab analysis results into the `Lab\_Analysis\_Results` table.

7. Stored Procedure Invocation:

- Calls the `GetAverageTemperatureByWaterBody` stored procedure to calculate the average temperature of a specified water body.

8. Cursor Invocation: Calls the `UpdatePollutionType` stored procedure to update pollution types based on certain conditions.

***CODE***

***Based on the provided entity-relationship (ER) diagram, here are the SQL commands to create tables, establish relationships, and insert sample data:***

***-- Create tables***

***CREATE TABLE Customer (***

***CustomerID INT PRIMARY KEY,***

***Name VARCHAR(50),***

***DOB DATE,***

***Occupation VARCHAR(50),***

***Address VARCHAR(100),***

***email VARCHAR(50),***

***cred\_no VARCHAR(20)***

***);***

***CREATE TABLE Classifier (***

***ClassifierID INT PRIMARY KEY,***

***Name VARCHAR(50),***

***DOB DATE,***

***Occupation VARCHAR(50),***

***Status VARCHAR(20),***

***email VARCHAR(50),***

***Address VARCHAR(100)***

***);***

***CREATE TABLE Relationship (***

***RelationshipID INT PRIMARY KEY,***

***CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID),***

***ClassifierID INT FOREIGN KEY REFERENCES Classifier(ClassifierID)***

***);***

***CREATE TABLE Discount\_details (***

***Discount\_ID INT PRIMARY KEY,***

***Discount\_type VARCHAR(20),***

***One VARCHAR(10)***

***);***

***CREATE TABLE Vehicle (***

***Vehicle\_ID INT PRIMARY KEY,***

***Ord\_ID INT,***

***vehicle\_model VARCHAR(50),***

***vehicle\_name VARCHAR(50),***

***deposit DECIMAL(10,2)***

***);***

***CREATE TABLE Bill\_ID (***

***Bill\_ID INT PRIMARY KEY,***

***Book\_ID INT,***

***Ad\_word VARCHAR(10),***

***Suit\_no VARCHAR(20),***

***Amount DECIMAL(10,2)***

***);***

***CREATE TABLE Billing (***

***Ord\_ID INT PRIMARY KEY,***

***Bill INT FOREIGN KEY REFERENCES Bill\_ID(Bill\_ID),***

***Bill\_type VARCHAR(20),***

***Bill\_terms VARCHAR(50),***

***Bill\_amount DECIMAL(10,2),***

***Ad\_account VARCHAR(50),***

***advance DECIMAL(10,2),***

***discount DECIMAL(10,2)***

***);***

***CREATE TABLE Booking\_details (***

***Book\_ID INT PRIMARY KEY,***

***Fees\_ID INT,***

***Drive\_option VARCHAR(20),***

***Funding VARCHAR(20),***

***date DATE,***

***Outcome\_ID INT***

***);***

***-- Insert sample data***

***INSERT INTO Customer (CustomerID, Name, DOB, Occupation, Address, email, cred\_no)***

***VALUES (1, 'John Doe', '1985-07-15', 'Engineer', '123 Main St, Cityville', 'john.doe@email.com', '1234567890');***

***INSERT INTO Classifier (ClassifierID, Name, DOB, Occupation, Status, email, Address)***

***VALUES (1, 'Jane Smith', '1990-03-22', 'Manager', 'Active', 'jane.smith@email.com', '456 Oak Rd, Townville');***

***INSERT INTO Relationship (RelationshipID, CustomerID, ClassifierID)***

***VALUES (1, 1, 1);***

***INSERT INTO Discount\_details (Discount\_ID, Discount\_type, One)***

***VALUES (1, 'Seasonal', 'Yes');***

***INSERT INTO Vehicle (Vehicle\_ID, Ord\_ID, vehicle\_model, vehicle\_name, deposit)***

***VALUES (1, 1001, 'Sedan', 'City Car', 1000.00);***

***INSERT INTO Bill\_ID (Bill\_ID, Book\_ID, Ad\_word, Suit\_no, Amount)***

***VALUES (1, 101, 'Yes', 'ABC123', 5000.00);***

***INSERT INTO Billing (Ord\_ID, Bill, Bill\_type, Bill\_terms, Bill\_amount, Ad\_account, advance, discount)***

***VALUES (1001, 1, 'Monthly', 'Net 30', 2500.00, 'marketing@company.com', 500.00, 100.00);***

***INSERT INTO Booking\_details (Book\_ID, Fees\_ID, Drive\_option, Funding, date, Outcome\_ID)***

***VALUES (101, 501, 'Self-drive', 'Cash', '2024-04-25', 1);***

***Note: This creates the tables based on the ER diagram and inserts sample data into each table. You can modify or add more data as per your requirements.***

**Conclusion :**

This SQL script creates a database schema for managing environmental data related to water bodies, including tables for water bodies, pollution levels, chemical attributes, biological data, temperature readings, samples, scientists, local authorities, and reports. It populates these tables with sample data and demonstrates various SQL operations like joins, updates, unions, intersects, and subqueries. Additionally, it defines stored procedures for retrieving average temperatures and updating pollution types, as well as a trigger for updating reports upon new lab analysis results. Overall, it provides a comprehensive framework for environmental data management and analysis in a relational database system.