DBMS

**Q. 1. Implementation of DDL commands of SQL with suitable examples**

* **Create table [20 Marks]**
* **Alter table**
* **Drop Table**

Create table student(Roll no ,sname,date of birth).Add new column into student relation name address as text data type and column phone of data type integer

ANS:-Sure, let's go through the implementation of the DDL (Data Definition Language) commands in SQL with suitable examples.

1. \*\*Create Table\*\*: This command is used to create a new table in the database.

```sql

CREATE TABLE student (

RollNo INT PRIMARY KEY,

SName VARCHAR(50),

DateOfBirth DATE

);

```

This SQL statement creates a table named `student` with three columns: `RollNo`, `SName`, and `DateOfBirth`. The `RollNo` column is of type `INT` and serves as the primary key for the table. `SName` is a `VARCHAR` (variable character) type column, and `DateOfBirth` is a `DATE` type column.

2. \*\*Alter Table\*\*: This command is used to modify an existing table structure. In this case, we want to add a new column named `Address` to the `student` table.

```sql

ALTER TABLE student

ADD Address TEXT;

```

This SQL statement alters the `student` table to add a new column named `Address` of type `TEXT`.

3. \*\*Alter Table\*\* (Adding another column):

```sql

ALTER TABLE student

ADD Phone INT;

```

This SQL statement further alters the `student` table to add another new column named `Phone` of type `INT`.

4. \*\*Drop Table\*\*: This command is used to remove an existing table from the database.

```sql

DROP TABLE student;

```

This SQL statement drops the `student` table from the database.

So, the combined SQL script for the given requirements would be:

```sql

-- Create table

CREATE TABLE student (

RollNo INT PRIMARY KEY,

SName VARCHAR(50),

DateOfBirth DATE

);

-- Add new column 'Address'

ALTER TABLE student

ADD Address TEXT;

-- Add new column 'Phone'

ALTER TABLE student

ADD Phone INT;

-- Drop table

DROP TABLE student;

```

This script creates a table named `student`, adds two columns (`Address` and `Phone`), and finally drops the table.

**Q. 2. Write a PL/SQL Program to Find Factorial of a Number**

-- PL/SQL Program to Find Factorial of a Number

-- Declare variables

DECLARE

num INTEGER := &number; -- Input number for which factorial is to be found

factorial INTEGER := 1; -- Initialize factorial to 1

BEGIN

-- Check if input number is negative

IF num < 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Factorial is not defined for negative numbers.');

ELSE

-- Calculate factorial

FOR i IN 1..num LOOP

factorial := factorial \* i;

END LOOP;

-- Output the factorial

DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || num || ' is ' || factorial);

END IF;

END;

**Q. 1. Implementation of different types of function with suitable examples**

* By using Number function
* By using Aggregate Function
* By using Character Function
* By using Conversion Function
* By using Date Function

Ans:-Sure, let's implement functions of different types in SQL with suitable examples:

* 1. \*\*Using Number Function\*\*:
* ```sql
* -- Example using Number Function (ABS)
* SELECT ABS(-10) AS AbsoluteValue FROM dual;
* -- Output: AbsoluteValue
* -- 10
* ```
* 2. \*\*Using Aggregate Function\*\*:
* ```sql
* -- Example using Aggregate Function (SUM)
* SELECT SUM(salary) AS TotalSalary FROM employees;
* -- Output: TotalSalary
* -- 100000
* ```
* 3. \*\*Using Character Function\*\*:
* ```sql
* -- Example using Character Function (UPPER)
* SELECT UPPER('hello') AS UppercaseString FROM dual;
* -- Output: UppercaseString
* -- HELLO
* ```
* 4. \*\*Using Conversion Function\*\*:
* ```sql
* -- Example using Conversion Function (TO\_NUMBER)
* SELECT TO\_NUMBER('123') AS NumericValue FROM dual;
* -- Output: NumericValue
* -- 123
* ```
* 5. \*\*Using Date Function\*\*:
* ```sql
* -- Example using Date Function (SYSDATE)
* SELECT SYSDATE AS CurrentDate FROM dual;
* -- Output: CurrentDate
* -- 2024-06-02 14:30:45
* ```
* **Q. 2. Write a PL/SQL Program to Find Factorial of a Number**
* Q. 1. PL/SQL Program to Print Table of a Number
* Ans:--- PL/SQL Program to Print Table of a Number
* -- Declare variables
* DECLARE
* num INTEGER := &number; -- Input number for which table is to be printed
* BEGIN
* -- Check if input number is non-negative
* IF num >= 0 THEN
* -- Loop to print the table
* FOR i IN 1..10 LOOP
* DBMS\_OUTPUT.PUT\_LINE(num || ' \* ' || i || ' = ' || (num \* i));
* END LOOP;
* ELSE
* DBMS\_OUTPUT.PUT\_LINE('Table is not defined for negative numbers.');
* END IF;
* END;
* **Q. 2. Implementation of different types of Joins**
* **Inner Join**
* **Outer Join**
* **Natural Join**  [20 Marks]
* Customer(customer\_id,first\_name)
* Order(order\_id,amount)
* Cust\_order(customerid , first\_amount,amount)
* Certainly! Here's how you can implement different types of joins (Inner Join, Outer Join, and Natural Join) using the provided tables.
* Assuming the tables are defined as follows:
* ```sql
* CREATE TABLE Customer (
* customer\_id INT PRIMARY KEY,
* first\_name VARCHAR(50)
* );
* CREATE TABLE Order (
* order\_id INT PRIMARY KEY,
* amount DECIMAL(10,2)
* );
* CREATE TABLE Cust\_order (
* customer\_id INT,
* order\_id INT,
* first\_amount DECIMAL(10,2),
* amount DECIMAL(10,2),
* FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),
* FOREIGN KEY (order\_id) REFERENCES Order(order\_id)
* );
* ```
* Now, let's proceed with examples for each type of join:
* 1. \*\*Inner Join\*\*:
* Inner Join returns rows when there is at least one match in both tables.
* ```sql
* SELECT Customer.customer\_id, Customer.first\_name, Order.order\_id, Order.amount
* FROM Customer
* INNER JOIN Cust\_order ON Customer.customer\_id = Cust\_order.customer\_id
* INNER JOIN Order ON Cust\_order.order\_id = Order.order\_id;
* ```
* 2. \*\*Outer Join\*\*:
* Outer Join returns all rows from both tables, matching rows from both tables when available, and filling in NULLs for missing matches on either side.
* - Left Outer Join:
* ```sql
* SELECT Customer.customer\_id, Customer.first\_name, Cust\_order.order\_id, Cust\_order.amount
* FROM Customer
* LEFT OUTER JOIN Cust\_order ON Customer.customer\_id = Cust\_order.customer\_id;
* ```
* - Right Outer Join:
* ```sql
* SELECT Cust\_order.customer\_id, Cust\_order.order\_id, Order.amount
* FROM Cust\_order
* RIGHT OUTER JOIN Order ON Cust\_order.order\_id = Order.order\_id;
* ```
* 3. \*\*Natural Join\*\*:
* Natural Join performs a join using all columns with the same name in both tables.
* ```sql
* SELECT \*
* FROM Customer
* NATURAL JOIN Cust\_order;
* ```
* **Q. 2. Write a program to implement SQL Cursors.**

-- SQL Program to Implement Cursors

* -- Declare variables
* DECLARE
* emp\_id employees.employee\_id%TYPE;
* emp\_first\_name employees.first\_name%TYPE;
* emp\_last\_name employees.last\_name%TYPE;
* emp\_salary employees.salary%TYPE;
* -- Declare cursor
* CURSOR emp\_cursor IS
* SELECT employee\_id, first\_name, last\_name, salary
* FROM employees;
* BEGIN
* -- Open cursor
* OPEN emp\_cursor;
* -- Fetch data from cursor
* LOOP
* FETCH emp\_cursor INTO emp\_id, emp\_first\_name, emp\_last\_name, emp\_salary;
* EXIT WHEN emp\_cursor%NOTFOUND;
* -- Display employee information
* DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp\_id);
* DBMS\_OUTPUT.PUT\_LINE('First Name: ' || emp\_first\_name);
* DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || emp\_last\_name);
* DBMS\_OUTPUT.PUT\_LINE('Salary: ' || emp\_salary);
* DBMS\_OUTPUT.PUT\_LINE('-------------------------');
* END LOOP;
* -- Close cursor
* CLOSE emp\_cursor;
* END;
* **Q. 1. Write a program by using PL/SQL.**

Ans:--- PL/SQL Program to Calculate Factorial

* -- Declare variables
* DECLARE
* num INTEGER := &number; -- Input number for which factorial is to be found
* factorial INTEGER := 1; -- Initialize factorial to 1
* BEGIN
* -- Check if input number is negative
* IF num < 0 THEN
* DBMS\_OUTPUT.PUT\_LINE('Factorial is not defined for negative numbers.');
* ELSE
* -- Calculate factorial
* FOR i IN 1..num LOOP
* factorial := factorial \* i;
* END LOOP;
* -- Output the factorial
* DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || num || ' is ' || factorial);
* END IF;
* END;
* Certainly! Here's an explanation and example of each type of operator in SQL:
* **Q. 1 Implementation of different types of operators in SQL [20 Marks]**
* **∙ Arithmetic Operators**
* **∙ Logical Operators**
* **∙ Comparison Operator**
* **∙ Special Operator**
* **∙ Set Operation**
* 1. \*\*Arithmetic Operators\*\*:
* - Arithmetic operators perform mathematical operations on numeric operands.
* Example:
* ```sql
* SELECT 10 + 5 AS Addition, 10 - 5 AS Subtraction, 10 \* 5 AS Multiplication, 10 / 5 AS Division
* FROM dual;
* ```
* 2. \*\*Logical Operators\*\*:
* - Logical operators are used to combine multiple conditions together.
* Example:
* ```sql
* SELECT \* FROM employees
* WHERE salary > 50000 AND department\_id = 10;
* ```
* 3. \*\*Comparison Operators\*\*:
* - Comparison operators are used to compare values.
* Example:
* ```sql
* SELECT \* FROM products
* WHERE price >= 100;
* ```
* 4. \*\*Special Operator\*\*:
* - Special operators perform special functions in SQL.
* Example:
* ```sql
* SELECT \* FROM customers
* WHERE customer\_name LIKE 'A%';
* ```
* - `LIKE` operator is a special operator used for pattern matching.
* 5. \*\*Set Operators\*\*:
* - Set operators are used to combine the results of two or more SELECT queries.
* Example:
* ```sql
* SELECT employee\_id, first\_name FROM employees
* UNION
* SELECT employee\_id, first\_name FROM temp\_employees;
* ```
* - `UNION` operator combines the results of two or more SELECT statements and removes duplicate rows.
* These are some of the most commonly used operators in SQL, each serving a different purpose in querying and manipulating data in a database.
* Q**. 2 . Implementation of DDL commands of SQL with suitable examples**
* Sure, let's go through the implementation of the DDL (Data Definition Language) commands in SQL with suitable examples:
* 1. \*\*CREATE TABLE\*\*:
* - This command is used to create a new table in the database.
* Example:
* ```sql
* CREATE TABLE employees (
* employee\_id INT PRIMARY KEY,
* first\_name VARCHAR(50),
* last\_name VARCHAR(50),
* department\_id INT,
* salary DECIMAL(10,2)
* );
* ```
* 2. \*\*ALTER TABLE\*\*:
* - This command is used to modify an existing table structure.
* Example:
* ```sql
* ALTER TABLE employees
* ADD email VARCHAR(100);
* ```
* 3. \*\*DROP TABLE\*\*:
* - This command is used to remove an existing table from the database.
* Example:
* ```sql
* DROP TABLE employees;
* ```
* 4. \*\*CREATE INDEX\*\*:
* - This command is used to create an index on a table.
* Example:
* ```sql
* CREATE INDEX idx\_department\_id ON employees (department\_id);
* ```
* 5. \*\*DROP INDEX\*\*:
* - This command is used to remove an index from the database.
* Example:
* ```sql
* DROP INDEX idx\_department\_id;
* ```
* **These are some of the commonly used DDL commands in SQL, each serving a different purpose in defining and modifying the structure of the database schema.**
* **Create table [20 Marks]**
* **Alter table**
* **Drop Table**
* Create table student(Roll no ,sname,date of birth).Add new column into student relation name address as text data type and column phone of data type integer
* Ans:-Certainly! Below is the implementation of the DDL commands `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` in SQL with suitable examples:
* 1. \*\*CREATE TABLE\*\*:
* - This command is used to create a new table in the database.
* ```sql
* CREATE TABLE student (
* RollNo INT PRIMARY KEY,
* SName VARCHAR(50),
* DateOfBirth DATE
* );
* ```
* This SQL statement creates a table named `student` with three columns: `RollNo`, `SName`, and `DateOfBirth`. The `RollNo` column is of type `INT` and serves as the primary key for the table. `SName` is a `VARCHAR` (variable character) type column, and `DateOfBirth` is a `DATE` type column.
* 2. \*\*ALTER TABLE\*\*:
* - This command is used to modify an existing table structure.
* ```sql
* ALTER TABLE student
* ADD Address TEXT,
* ADD Phone INT;
* ```
* This SQL statement alters the `student` table to add two new columns: `Address` of type `TEXT` and `Phone` of type `INT`.
* 3. \*\*DROP TABLE\*\*:
* - This command is used to remove an existing table from the database.
* ```sql
* DROP TABLE student;
* ```
* This SQL statement drops the `student` table from the database.
* So, the combined SQL script for the given requirements would be:
* ```sql
* -- Create table
* CREATE TABLE student (
* RollNo INT PRIMARY KEY,
* SName VARCHAR(50),
* DateOfBirth DATE
* );
* -- Add new column 'Address' and 'Phone'
* ALTER TABLE student
* ADD Address TEXT,
* ADD Phone INT;
* -- Drop table
* DROP TABLE student;
* ```
* This script creates a table named `student`, adds two columns (`Address` and `Phone`), and finally drops the table.
* **Q. 1. Study & Implementation of SQL Triggers.**
* Ans:--- Create audit\_log table
* CREATE TABLE audit\_log (
* log\_id INT PRIMARY KEY,
* action VARCHAR(50),
* table\_name VARCHAR(50),
* timestamp TIMESTAMP
* );
* -- Create trigger for inserting into employees table
* CREATE OR REPLACE TRIGGER employee\_insert\_trigger
* AFTER INSERT ON employees
* FOR EACH ROW
* BEGIN
* INSERT INTO audit\_log (log\_id, action, table\_name, timestamp)
* VALUES (audit\_log\_seq.NEXTVAL, 'INSERT', 'employees', SYSTIMESTAMP);
* END;
* /
* **Q. 1. Study & Implementation of PL/SQL**
* Ans:--- PL/SQL Block to Calculate Total Salary
* DECLARE
* total\_salary NUMBER := 0;
* BEGIN
* FOR emp\_rec IN (SELECT salary FROM employees WHERE department\_id = 10) LOOP
* total\_salary := total\_salary + emp\_rec.salary;
* END LOOP;
* DBMS\_OUTPUT.PUT\_LINE('Total Salary for Department 10: ' || total\_salary);
* END;
* /
* **Q. 2. . Implementation of different types of function with suitable examples**
* **Number function**
* **Aggregate Function**
* **Character Function**
* **Conversion Function**
* **Date Function**
* Ans:-Certainly! Below are implementations of different types of functions in SQL with suitable examples:
* 1. \*\*Number Function\*\* (ABS):
* - This function returns the absolute value of a number.
* ```sql
* SELECT ABS(-10) AS AbsoluteValue;
* -- Output: AbsoluteValue
* -- 10
* ```
* 2. \*\*Aggregate Function\*\* (SUM):
* - This function calculates the sum of values in a column.
* ```sql
* SELECT SUM(salary) AS TotalSalary FROM employees;
* -- Output: TotalSalary
* -- 100000
* ```
* 3. \*\*Character Function\*\* (UPPER):
* - This function converts a string to uppercase.
* ```sql
* SELECT UPPER('hello') AS UppercaseString;
* -- Output: UppercaseString
* -- HELLO
* ```
* 4. \*\*Conversion Function\*\* (TO\_NUMBER):
* - This function converts a string to a number.
* ```sql
* SELECT TO\_NUMBER('123') AS NumericValue;
* -- Output: NumericValue
* -- 123
* ```
* 5. \*\*Date Function\*\* (SYSDATE):
* - This function returns the current system date and time.
* ```sql
* SELECT SYSDATE AS CurrentDate FROM dual;
* -- Output: CurrentDate
* -- 2024-06-02 14:30:45
* ```
* These examples demonstrate the usage of different types of functions in SQL, including number functions, aggregate functions, character functions, conversion functions, and date functions. Each function serves a specific purpose in SQL queries and can be used to manipulate and transform data as needed.
* **Q. 2. Study & Implementation of SQL Triggers.**
* Ans:--- Create audit\_log table
* CREATE TABLE audit\_log (
* log\_id INT PRIMARY KEY,
* action VARCHAR(50),
* table\_name VARCHAR(50),
* timestamp TIMESTAMP
* );
* -- Create trigger for inserting into employees table
* CREATE OR REPLACE TRIGGER employee\_insert\_trigger
* AFTER INSERT ON employees
* FOR EACH ROW
* BEGIN
* INSERT INTO audit\_log (log\_id, action, table\_name, timestamp)
* VALUES (audit\_log\_seq.NEXTVAL, 'INSERT', 'employees', SYSTIMESTAMP);
* END;
* /
* **Q. 1. Implementation of DDL commands of SQL with suitable examples**
* **Create table**
* **Alter table**
* **Drop** TaCertainly! Let's go through the implementation of the DDL (Data Definition Language) commands in SQL with suitable examples:
* 1. \*\*CREATE TABLE\*\*:
* - This command is used to create a new table in the database.
* ```sql
* CREATE TABLE employees (
* employee\_id INT PRIMARY KEY,
* first\_name VARCHAR(50),
* last\_name VARCHAR(50),
* department\_id INT,
* salary DECIMAL(10,2)
* );
* ```
* This SQL statement creates a table named `employees` with five columns: `employee\_id`, `first\_name`, `last\_name`, `department\_id`, and `salary`. The `employee\_id` column is of type `INT` and serves as the primary key for the table. `first\_name` and `last\_name` are `VARCHAR` (variable character) type columns, `department\_id` is an `INT`, and `salary` is a `DECIMAL` type column.
* 2. \*\*ALTER TABLE\*\*:
* - This command is used to modify an existing table structure.
* ```sql
* ALTER TABLE employees
* ADD email VARCHAR(100);
* ```
* This SQL statement alters the `employees` table to add a new column named `email` of type `VARCHAR(100)`.
* 3. \*\*DROP TABLE\*\*:
* - This command is used to remove an existing table from the database.
* ```sql
* DROP TABLE employees;
* ```
* This SQL statement drops the `employees` table from the database.
* These examples demonstrate the usage of DDL commands in SQL to create, alter, and drop tables. Each command serves a different purpose in defining and modifying the structure of the database schema.ble
* **Q. 2 Implementation of DML commands of SQL with suitable examples**
* **Insert**
* Update
* Delete
* Ans:-Certainly! Let's go through the implementation of the DML (Data Manipulation Language) commands in SQL with suitable examples:
* 1. \*\*INSERT\*\*:
* - This command is used to insert new rows of data into a table.
* ```sql
* -- Inserting a single row into the employees table
* INSERT INTO employees (employee\_id, first\_name, last\_name, department\_id, salary)
* VALUES (1, 'John', 'Doe', 101, 50000.00);
* -- Inserting multiple rows into the employees table
* INSERT INTO employees (employee\_id, first\_name, last\_name, department\_id, salary)
* VALUES (2, 'Jane', 'Smith', 102, 60000.00),
* (3, 'David', 'Johnson', 101, 55000.00);
* ```
* 2. \*\*UPDATE\*\*:
* - This command is used to modify existing records in a table.
* ```sql
* -- Update the salary of employee with employee\_id 1
* UPDATE employees
* SET salary = 52000.00
* WHERE employee\_id = 1;
* -- Update the department\_id for employees in department 101
* UPDATE employees
* SET department\_id = 102
* WHERE department\_id = 101;
* ```
* 3. \*\*DELETE\*\*:
* - This command is used to remove rows from a table.
* ```sql
* -- Delete employee with employee\_id 2
* DELETE FROM employees
* WHERE employee\_id = 2;
* -- Delete all employees in department 101
* DELETE FROM employees
* WHERE department\_id = 101;
* ```
* These examples demonstrate the usage of DML commands in SQL to insert, update, and delete data in a table. Each command serves a different purpose in manipulating the data stored in the database.