Experiment - 9

Aim:
Write the queries to create the views and queries based on views.

Tool Used: Maria DB

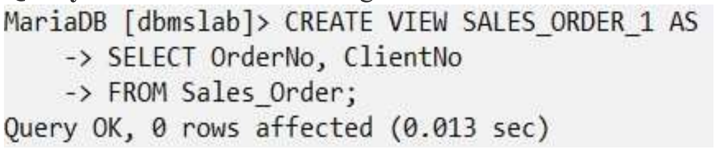
Theory and Procedure:

A VIEW is a virtual table, through which a selective portion of the data from one or more tables can be seen. Views do not contain data of their own. They are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database. DML operations on a view like INSERT, UPDATE, DELETE affects the data in the original table upon which the view is based.

Syntax:
CREATE VIEW view_name AS SELECT
column1, column2, ...
FROM table_name;

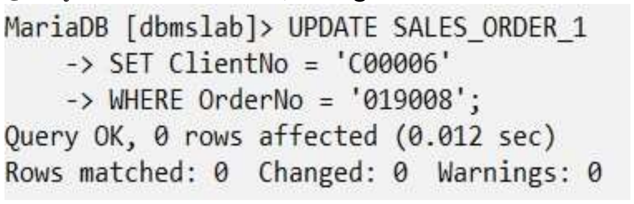
Queries:

Query-1: Create a view having OrderNo and ClientNo.



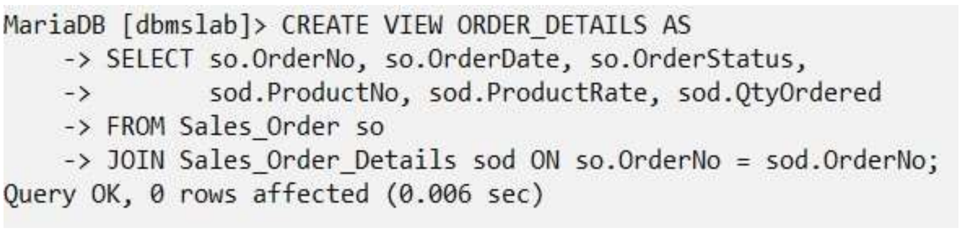
```
MariaDB [dbmslab]> CREATE VIEW SALES_ORDER_1 AS
    -> SELECT OrderNo, ClientNo
    -> FROM Sales_Order;
Query OK, 0 rows affected (0.013 sec)
```

Query-2: In above view, change client No to ‘C00006’ where order No is ‘019008’.



```
MariaDB [dbmslab]> UPDATE SALES_ORDER_1
    -> SET ClientNo = 'C00006'
    -> WHERE OrderNo = '019008';
Query OK, 0 rows affected (0.012 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

Query-3: Create a view on OrderNo, OrderDate, OrderStatus of the sales_order table and ProductNo, ProductRate and QtyOrdered of Sales_Order_Details.



```
MariaDB [dbmslab]> CREATE VIEW ORDER_DETAILS AS
    -> SELECT so.OrderNo, so.OrderDate, so.OrderStatus,
    ->        sod.ProductNo, sod.ProductRate, sod.QtyOrdered
    -> FROM Sales_Order so
    -> JOIN Sales_Order_Details sod ON so.OrderNo = sod.OrderNo;
Query OK, 0 rows affected (0.006 sec)
```

Experiment – 10

Aim:

Demonstrate the concept of Control Structures.

Theory:

The selection structure tests a condition, then executes one sequence of statements instead of another, depending on whether the condition is true or false. A condition is any variable or expression that returns a BOOLEAN value (TRUE or FALSE).

The iteration structure executes a sequence of statements repeatedly as long as a condition holds true.

The sequence-structure simply executes a sequence of statements in the order in which they occur.

Syntax of IF-THEN Statement:

```
IF condition THEN
   sequence_of_statements
END IF;
```

Query:

```
IF sales > quota THEN
   compute_bonus(empid);
   UPDATE payroll SET pay = pay + bonus WHERE empno = emp_id;
END IF;
```

Syntax of IF-THEN-ELSE Statement:

```
IF condition THEN
   sequence_of_statements1
ELSE
   sequence_of_statements2
END IF;
```

Query:

```
IF trans_type = 'CR' THEN
   UPDATE accounts SET balance = balance + credit WHERE ...
ELSE
   UPDATE accounts SET balance = balance - debit WHERE ...
END IF;
```

Syntax of IF-THEN-ELSEIF Statement:

```
IF condition1 THEN
   sequence_of_statements1
ELSIF condition2 THEN
   sequence_of_statements2
```

```
ELSE
   sequence_of_statements3
END IF;
```

Query:

```
BEGIN
   ...
   IF sales > 50000 THEN
      bonus := 1500;
   ELSIF sales > 35000 THEN
      bonus := 500;
   ELSE
      bonus := 100;
   END IF;
   INSERT INTO payroll VALUES (emp_id, bonus, ...);
END;
```

Syntax of CASE Statement:

```
CASE grade
   WHEN 'case1' THEN sequence_of_statements1;
   WHEN 'case2' THEN sequence_of_statements1;
   WHEN 'case3' THEN sequence_of_statements1;
   ...
   ...
   ELSE sequence_of_default_statement;
END CASE;
```

Query:

```
CASE grade
   WHEN 'A' THEN dbms_output.put_line('Excellent');
   WHEN 'B' THEN dbms_output.put_line('Very Good');
   WHEN 'C' THEN dbms_output.put_line('Good');
   WHEN 'D' THEN dbms_output.put_line('Fair');
   WHEN 'F' THEN dbms_output.put_line('Poor');
   ELSE dbms_output.put_line('No such grade');
END CASE;
```

Experiment – 11

Aim:

Demonstrate the concept of Exception Handling.

Theory:

An exception is an error which disrupts the normal flow of program instructions. PL/SQL provides us the exception block which raises the exception thus helping the programmer to find out the fault and resolve it.

There are two types of exceptions defined in PL/SQL

1. System defined exceptions.
2. User defined exception.

Syntax to write an exception:

WHEN exception **THEN**
  statement;
DECLARE
declarations section;
BEGIN
executable command(s);
EXCEPTION
WHEN exception1 THEN
statement1;
WHEN exception2 THEN
statement2;
[WHEN others THEN]
/ default exception handling code */*
END;

Query for TOO_MANY_ROWS Exception:

```
DECLARE
      temp varchar(20);
BEGIN
      SELECT g_name into temp from geeks;
      dbms_output.put_line(temp);
EXCEPTION
   WHEN too_many_rows THEN
      dbms_output.put_line('error trying to SELECT too many rows');
end;
```

Output:

```
error trying to SELECT too many rows
```