SQL ASSIGNMENT -2

- Harshinie M

Assignment 1: Write a SELECT query to retrieve all columns from a 'customers' table and modify it to return only the customer name and email address for customers in a specific city.

```
mysql> SELECT * FROM customers;
+-------------+---------------+-----------------+-----------+--------+
| customer_id | customer_name | email           | city      | region |
+-------------+---------------+-----------------+-----------+--------+
|           1 | Anu           | anu@gmail.com   | Chennai   | South  |
|           2 | Ravi          | ravi@gmail.com  | Bangalore | South  |
|           3 | Meena         | meena@gmail.com | Chennai   | South  |
|           4 | Kiran         | kiran@gmail.com | Delhi     | North  |
|           5 | Priya         | priya@gmail.com | Mumbai    | West   |
+-------------+---------------+-----------------+-----------+--------+
5 rows in set (0.00 sec)

mysql> SELECT customer_name, email
    -> FROM customers
    -> WHERE city = 'Chennai';
+---------------+-----------------+
| customer_name | email           |
+---------------+-----------------+
| Anu           | anu@gmail.com   |
| Meena         | meena@gmail.com |
+---------------+-----------------+
2 rows in set (0.00 sec)
```

Assignment 2: Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.

```
mysql> CREATE TABLE orders (
    ->     order_id INT PRIMARY KEY,
```

```
    ->     customer_id INT,
    ->     order_amount INT,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO orders VALUES
    -> (101, 1, 5000),
    -> (102, 1, 3000),
    -> (103, 2, 4000),
    -> (104, 3, 6000);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> CREATE TABLE products (
    ->     product_id INT PRIMARY KEY,
    ->     product_name VARCHAR(50),
    ->     price INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO products VALUES
    -> (10, 'Laptop', 50000),
    -> (11, 'Mouse',  1000);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT c.customer_name, o.order_id, o.order_amount
    -> FROM customers c
    -> INNER JOIN orders o
    -> ON c.customer_id = o.customer_id
    -> WHERE c.region = 'South';
+---------------+----------+--------------+
| customer_name | order_id | order_amount |
+---------------+----------+--------------+
| Anu           |      101 |         5000 |
| Anu           |      102 |         3000 |
| Ravi          |      103 |         4000 |
| Meena         |      104 |         6000 |
+---------------+----------+--------------+
4 rows in set (0.00 sec)

mysql> SELECT c.customer_name, o.order_id
    -> FROM customers c
    -> LEFT JOIN orders o
    -> ON c.customer_id = o.customer_id;
+---------------+----------+
```

```
| customer_name | order_id |
+---------------+----------+
| Anu           |      101 |
| Anu           |      102 |
| Ravi          |      103 |
| Meena         |      104 |
| Kiran         |     NULL |
| Priya         |     NULL |
+---------------+----------+
6 rows in set (0.00 sec)

mysql>
```

Assignment 3: Utilize a subquery to find customers who have placed orders above the average order value and write a UNION query to combine two SELECT statements with the same number of columns.

```
mysql> SELECT *
    -> FROM orders
    -> WHERE order_amount >
    -> (
    ->    SELECT AVG(order_amount)
    ->    FROM orders
    -> );
+----------+-------------+--------------+
| order_id | customer_id | order_amount |
+----------+-------------+--------------+
|      101 |           1 |         5000 |
|      104 |           3 |         6000 |
+----------+-------------+--------------+
2 rows in set (0.01 sec)

mysql> SELECT customer_name FROM customers WHERE region = 'South'
    -> UNION
    -> SELECT customer_name FROM customers WHERE region = 'North';
+---------------+
| customer_name |
+---------------+
| Anu           |
| Ravi          |
| Meena         |
| Kiran         |
```

```
+---------------+
4 rows in set (0.00 sec)
```

Assignment 4: Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO orders VALUES (105, 4, 7000);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> UPDATE products
   -> SET price = price + 500
   -> WHERE product_id = 10;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Assignment 5: Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO orders VALUES (201, 1, 2000);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT sp1;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO orders VALUES (202, 2, 3000);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT sp2;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO orders VALUES (203, 3, 4000);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> ROLLBACK TO sp2;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Assignment 6: Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

Transaction Logs and Data Recovery

A transaction log records all database changes such as INSERT, UPDATE, COMMIT, and ROLLBACK. It helps the database maintain data consistency and reliability. In case of a

system crash, the database uses the transaction log to redo committed transactions and undo incomplete ones, ensuring no data corruption occurs.

Hypothetical Scenario

In an online banking system, a money transfer transaction debits an amount from one account but crashes before crediting the other account. When the system restarts, the database checks the transaction log, detects the incomplete transaction, and rolls it back, restoring the original balance.

Conclusion

Transaction logs are essential for crash recovery and ensure that the database remains consistent even after unexpected shutdowns.