Experiment 3 Output

In this experiment, the **Snowflake Schema** was implemented to achieve a more normalized data structure than the **Star Schema**.

Creating Database and Using in MySQL

```
mysql> CREATE DATABASE RetailDataWarehouse;
Query OK, 1 row affected (0.03 sec)

mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| classdb              |
| company              |
| employee             |
| hospital             |
| information_schema   |
| movie                |
| mydatabase           |
| mysql                |
| performance_schema   |
| retaildatawarehouse  |
| root                 |
| shopping             |
| student              |
| sys                  |
| utd                  |
+----------------------+
15 rows in set (0.00 sec)
```

Fig 1: Database in MySQL

Creating Tables

```
mysql> CREATE TABLE product_dim_normalized (
    ->     product_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     product_name VARCHAR(100) NOT NULL,
    ->     category_id INT,
    ->     brand VARCHAR(50),
    ->     FOREIGN KEY (category_id) REFERENCES product_category(category_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE region (
    ->     region_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     region_name VARCHAR(100) NOT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE city (
    ->     city_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     city_name VARCHAR(100) NOT NULL,
    ->     region_id INT,
    ->     FOREIGN KEY (region_id) REFERENCES region(region_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE customer_dim_normalized (
    ->     customer_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     customer_name VARCHAR(100) NOT NULL,
    ->     city_id INT,
    ->     FOREIGN KEY (city_id) REFERENCES city(city_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE time_dim (
    ->     time_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     date DATE NOT NULL,
    ->     day_of_week VARCHAR(10),
    ->     month VARCHAR(10),
    ->     quarter VARCHAR(10),
```

Fig 2: Creating Tables with datatypes

Inserting Data

```
mysql> CREATE TABLE sales_fact_normalized (
    ->     sale_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     product_id INT,
    ->     customer_id INT,
    ->     store_id INT,
    ->     time_id INT,
    ->     amount DECIMAL(10, 2),
    ->     quantity INT,
    ->     FOREIGN KEY (product_id) REFERENCES product_dim_nor
    ->     FOREIGN KEY (customer_id) REFERENCES customer_dim_n
    ->     FOREIGN KEY (time_id) REFERENCES time_dim_normalize
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO product_category (category_name) VALUES ('E
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO product_dim_normalized (product_name, categ
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO region (region_name) VALUES ('North');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO city (city_name, region_id) VALUES ('Delhi'
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO customer_dim_normalized (customer_name, cit
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO time_dim_normalized (date, day_of_week, mon
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO sales_fact_normalized (product_id, customer
Query OK, 1 row affected (0.00 sec)
```

Fig 3: Inserting data into tables

Aggregate Queries

```
mysql> INSERT INTO product_dim_normalized (product_name, ca
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO region (region_name) VALUES ('North');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO city (city_name, region_id) VALUES ('Del
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO customer_dim_normalized (customer_name,
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO time_dim_normalized (date, day_of_week,
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO sales_fact_normalized (product_id, custo
Query OK, 1 row affected (0.00 sec)

mysql> SELECT c.customer_name, p.product_name, s.amount
    -> FROM sales_fact_normalized s
    -> JOIN customer_dim_normalized c ON s.customer_id = c.
    -> JOIN product_dim_normalized p ON s.product_id = p.pr
+---------------+--------------+---------+
| customer_name | product_name | amount  |
+---------------+--------------+---------+
| John Doe      | Laptop       | 1000.00 |
+---------------+--------------+---------+
1 row in set (0.00 sec)
```

Fig 4: Aggregate Queries in tables