Lesson 05 Demo 09

Working with Various Built-in SQL Functions

Objective: To demonstrate the use of various built-in SQL functions in MySQL for performing data manipulation and retrieval efficiently

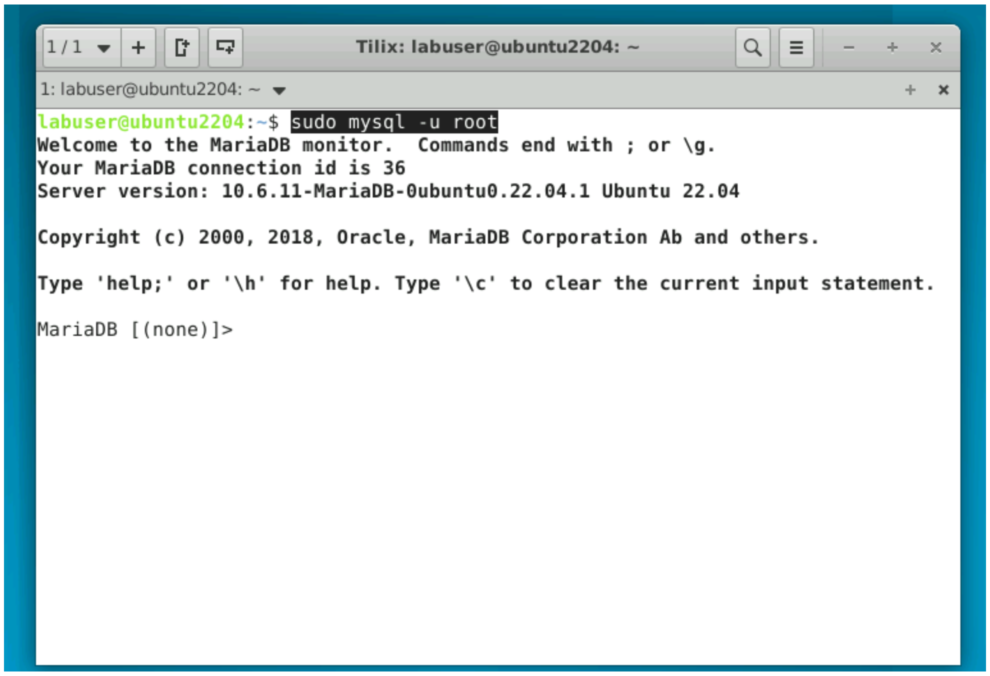
Tools required: MySQL

Prerequisites: None

Steps to be followed:
1. Set up a database and table
2. Use the built-in functions

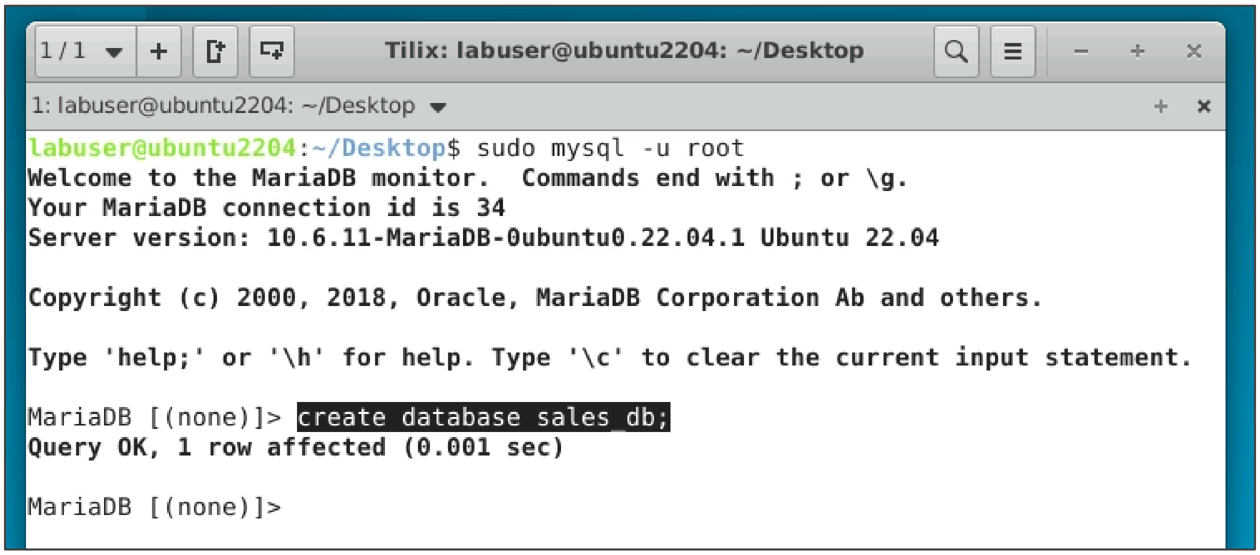
Step 1: Set up a database and table

1.1 Open a terminal window and access MySQL as a root user:
sudo mysql -u root



```
labuser@ubuntu2204:~$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.6.11-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

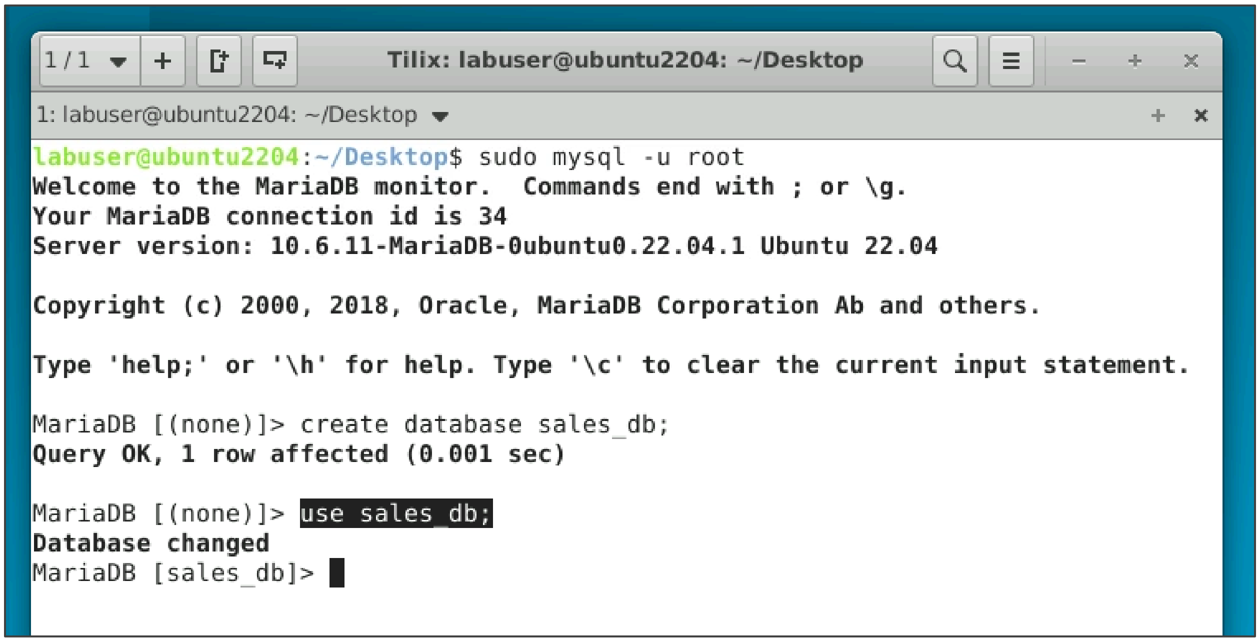
1.2 Create a new database named **sales_db**:
create database sales_db;



1 / 1
Tilix: labuser@ubuntu2204: ~/Desktop
1: labuser@ubuntu2204: ~/Desktop

```
labuser@ubuntu2204:~/Desktop$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.6.11-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database sales_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
```

1.3 Select the **sales_db** database:
use sales_db;



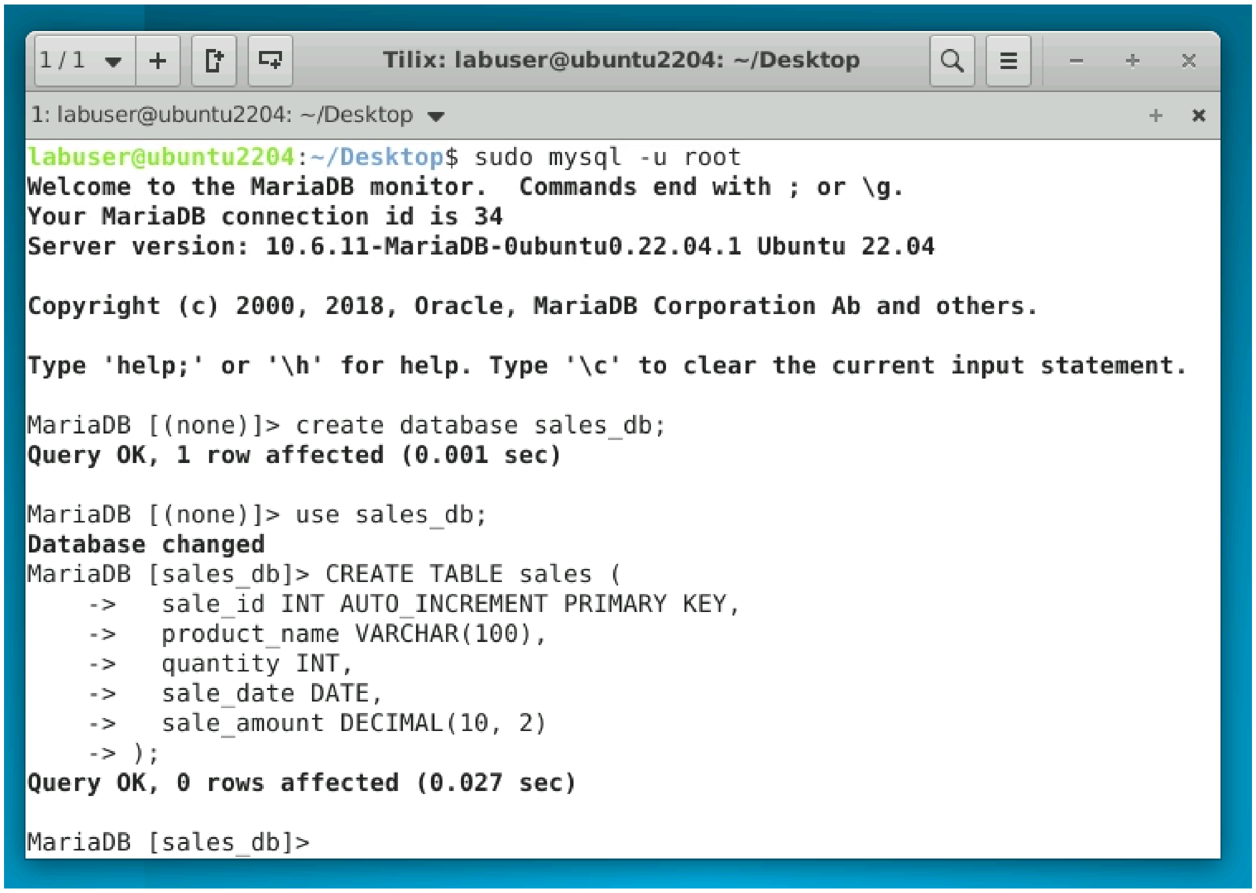
1 / 1
Tilix: labuser@ubuntu2204: ~/Desktop
1: labuser@ubuntu2204: ~/Desktop

```
labuser@ubuntu2204:~/Desktop$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.6.11-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database sales_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use sales_db;
Database changed
MariaDB [sales_db]>
```

.

1.4 Create a **sales** table with relevant fields:

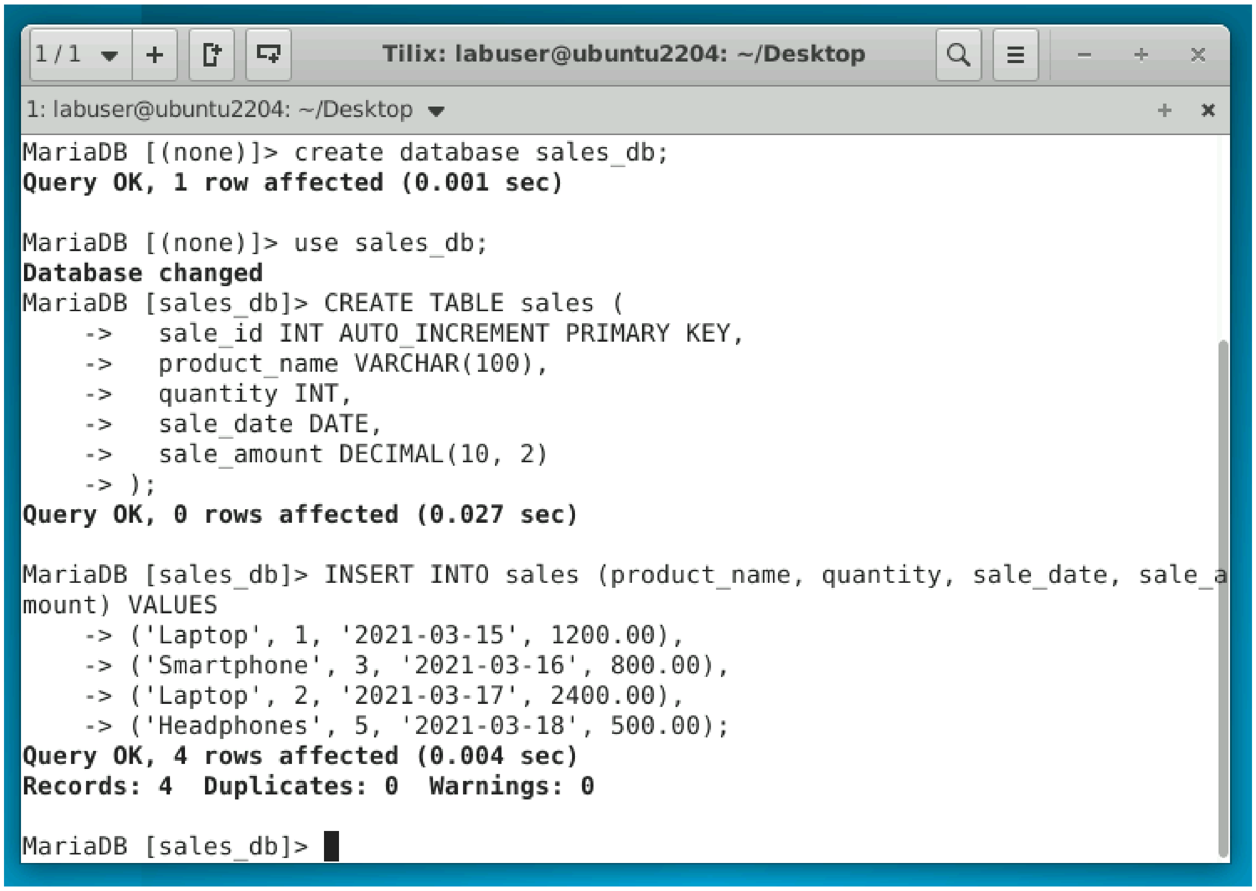
```
CREATE TABLE sales (
  sale_id INT AUTO_INCREMENT PRIMARY KEY,
  product_name VARCHAR(100),
  quantity INT,
  sale_date DATE,
  sale_amount DECIMAL(10, 2)
);
```


1 / 1
Tilix: labuser@ubuntu2204: ~/Desktop
1: labuser@ubuntu2204: ~/Desktop

```
labuser@ubuntu2204:~/Desktop$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.6.11-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database sales_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use sales_db;
Database changed
MariaDB [sales_db]> CREATE TABLE sales (
    ->     sale_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     product_name VARCHAR(100),
    ->     quantity INT,
    ->     sale_date DATE,
    ->     sale_amount DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.027 sec)

MariaDB [sales_db]>
```

1.5 Insert data into the **sales** table:

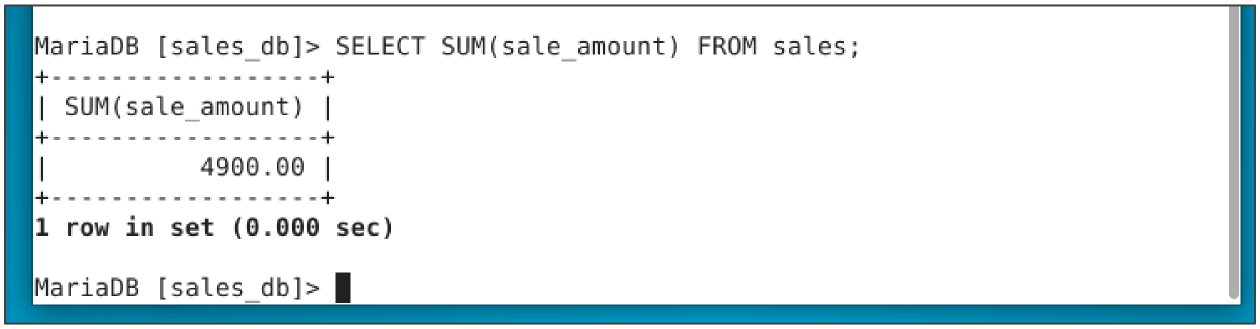
INSERT INTO sales (product_name, quantity, sale_date, sale_amount) VALUES
('Laptop', 1, '2021-03-15', 1200.00),
('Smartphone', 3, '2021-03-16', 800.00),
('Laptop', 2, '2021-03-17', 2400.00),
('Headphones', 5, '2021-03-18', 500.00);


1 / 1
Tilix: labuser@ubuntu2204: ~/Desktop
1: labuser@ubuntu2204: ~/Desktop

```
MariaDB [(none)]> create database sales_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use sales_db;
Database changed
MariaDB [sales_db]> CREATE TABLE sales (
    ->    sale_id INT AUTO_INCREMENT PRIMARY KEY,
    ->    product_name VARCHAR(100),
    ->    quantity INT,
    ->    sale_date DATE,
    ->    sale_amount DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.027 sec)

MariaDB [sales_db]> INSERT INTO sales (product_name, quantity, sale_date, sale_a
mount) VALUES
    -> ('Laptop', 1, '2021-03-15', 1200.00),
    -> ('Smartphone', 3, '2021-03-16', 800.00),
    -> ('Laptop', 2, '2021-03-17', 2400.00),
    -> ('Headphones', 5, '2021-03-18', 500.00);
Query OK, 4 rows affected (0.004 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [sales_db]> 
```

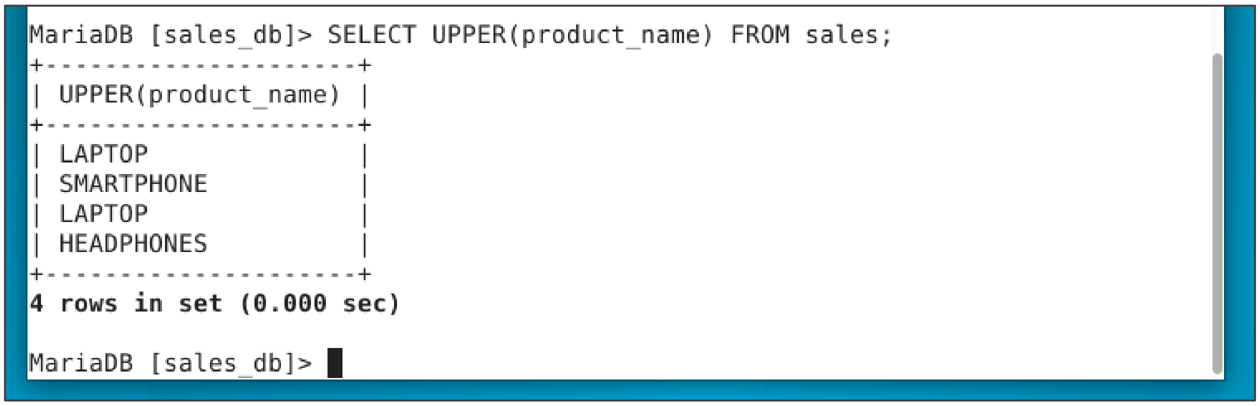
Step 2: Use the built-in functions

2.1 Calculate the total sales amount:
SELECT SUM(sale_amount) FROM sales;



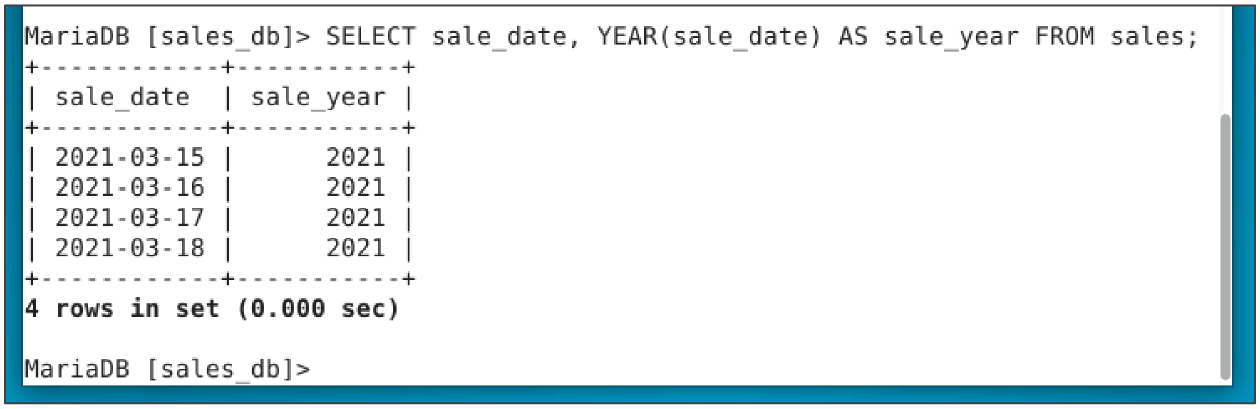
```
MariaDB [sales_db]> SELECT SUM(sale_amount) FROM sales;
+------------------+
| SUM(sale_amount) |
+------------------+
|          4900.00 |
+------------------+
1 row in set (0.000 sec)

MariaDB [sales_db]> 
```

2.2 Convert the product names to uppercase:
SELECT UPPER(product_name) FROM sales;



```
MariaDB [sales_db]> SELECT UPPER(product_name) FROM sales;
+---------------------+
| UPPER(product_name) |
+---------------------+
| LAPTOP              |
| SMARTPHONE          |
| LAPTOP              |
| HEADPHONES          |
+---------------------+
4 rows in set (0.000 sec)

MariaDB [sales_db]> 
```

2.3 Extract the year from the sale date:
SELECT sale_date, YEAR(sale_date) AS sale_year FROM sales;

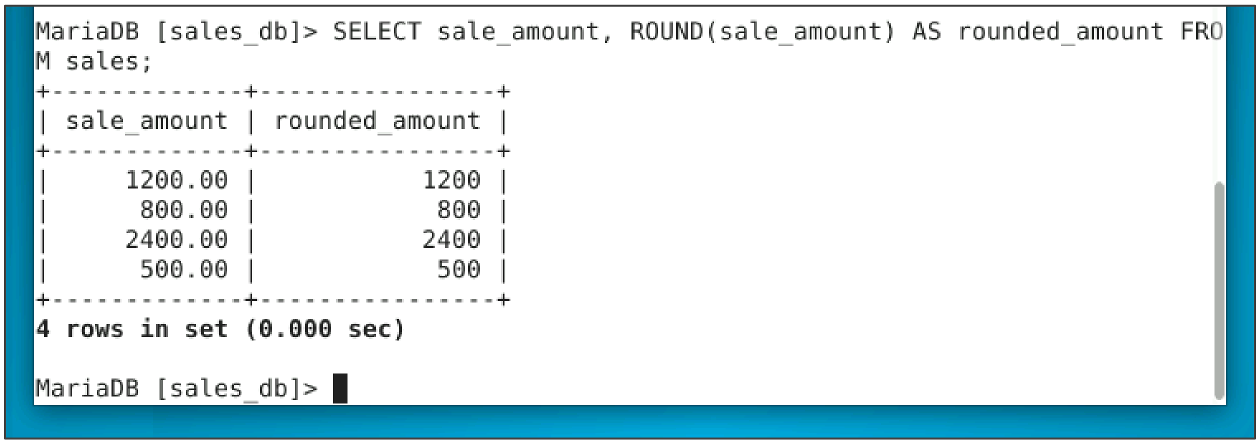


```
MariaDB [sales_db]> SELECT sale_date, YEAR(sale_date) AS sale_year FROM sales;
+------------+-----------+
| sale_date  | sale_year |
+------------+-----------+
| 2021-03-15 |      2021 |
| 2021-03-16 |      2021 |
| 2021-03-17 |      2021 |
| 2021-03-18 |      2021 |
+------------+-----------+
4 rows in set (0.000 sec)

MariaDB [sales_db]>
```

.

2.4 Round the sale amounts to the nearest whole number:

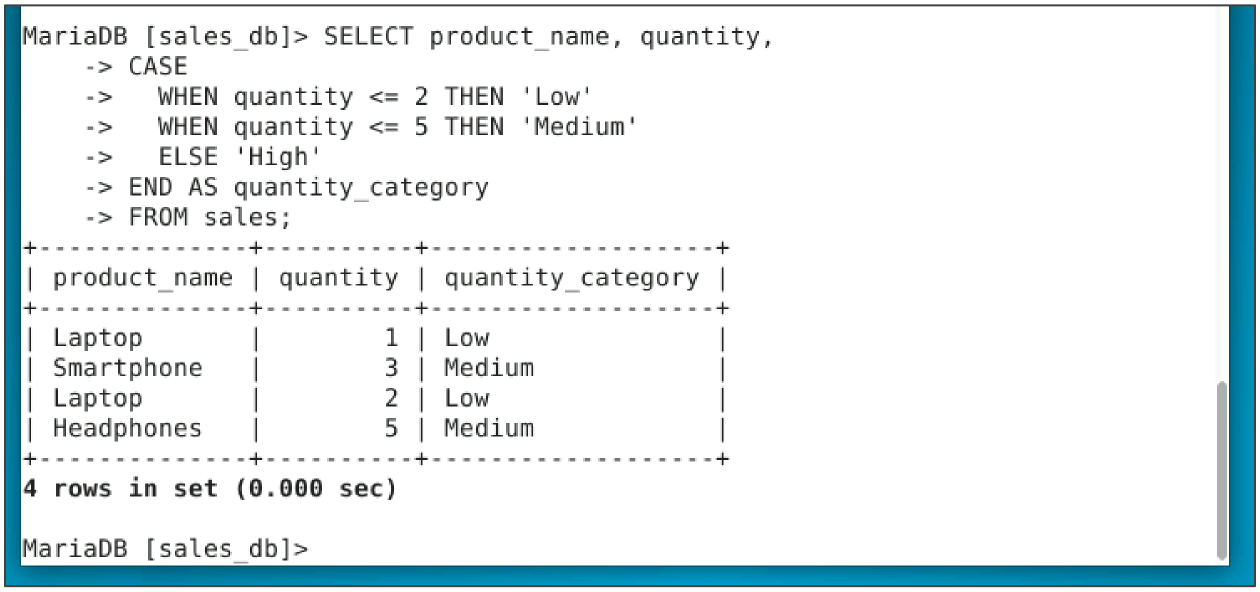
SELECT sale_amount, ROUND(sale_amount) AS rounded_amount FROM sales;



```
MariaDB [sales_db]> SELECT sale_amount, ROUND(sale_amount) AS rounded_amount FRO
M sales;
+-------------+----------------+
| sale_amount | rounded_amount |
+-------------+----------------+
|     1200.00 |           1200 |
|      800.00 |            800 |
|     2400.00 |           2400 |
|      500.00 |            500 |
+-------------+----------------+
4 rows in set (0.000 sec)

MariaDB [sales_db]> 
```

2.5 Use **CASE** to categorize sales based on quantity:

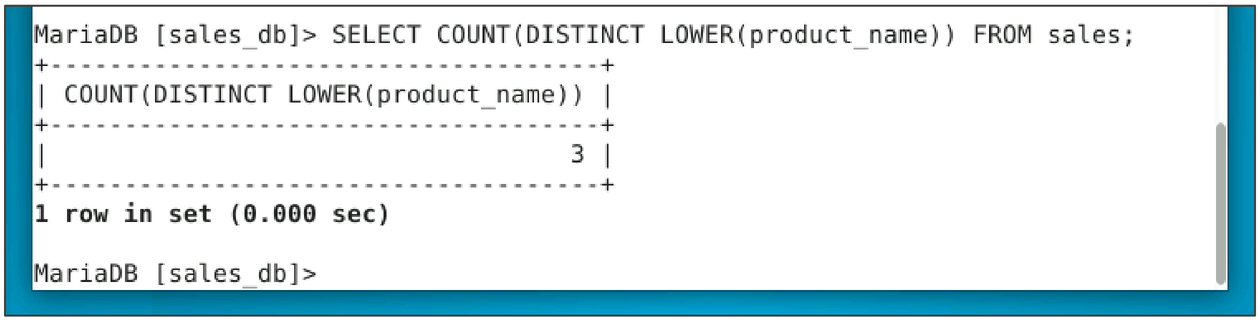
```
SELECT product_name, quantity,
CASE
  WHEN quantity <= 2 THEN 'Low'
  WHEN quantity <= 5 THEN 'Medium'
  ELSE 'High'
END AS quantity_category
FROM sales;
```



```
MariaDB [sales_db]> SELECT product_name, quantity,
    -> CASE
    ->   WHEN quantity <= 2 THEN 'Low'
    ->   WHEN quantity <= 5 THEN 'Medium'
    ->   ELSE 'High'
    -> END AS quantity_category
    -> FROM sales;
+--------------+----------+-------------------+
| product_name | quantity | quantity_category |
+--------------+----------+-------------------+
| Laptop       |        1 | Low               |
| Smartphone   |        3 | Medium            |
| Laptop       |        2 | Low               |
| Headphones   |        5 | Medium            |
+--------------+----------+-------------------+
4 rows in set (0.000 sec)

MariaDB [sales_db]>
```

.

2.6 Combine the string and aggregate functions to get a count of unique products sold:
SELECT COUNT(DISTINCT LOWER(product_name)) FROM sales;



```
MariaDB [sales_db]> SELECT COUNT(DISTINCT LOWER(product_name)) FROM sales;
+-------------------------------------+
| COUNT(DISTINCT LOWER(product_name)) |
+-------------------------------------+
|                                   3 |
+-------------------------------------+
1 row in set (0.000 sec)

MariaDB [sales_db]>
```

By following these steps, you have effectively utilized various built-in SQL functions in MySQL, enhancing your capability to manipulate and analyze the data.