**Tugas Basis Data Pertemuan 12**

**Anisa Intania Putri - 2024071010**

**Karine Olivia Permana – 2024071025**

- Lab 1 (File 7)

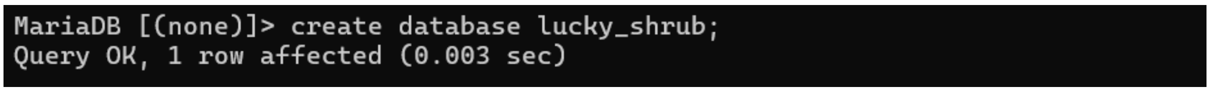
```
--  Optimizing SELECT statements in MySQL
-- 1: Create the database:
CREATE DATABASE Lucky_Shrub;

-- 2: Use the database:
USE Lucky_Shrub;

-- 3: Create the Orders table:
CREATE TABLE Orders(OrderID INT NOT NULL, ClientID VARCHAR(10)
DEFAULT NULL,
ProductID VARCHAR(10) DEFAULT NULL, Quantity INT DEFAULT NULL,
Cost DECIMAL(6,2) DEFAULT NULL,
Date DATE DEFAULT NULL, PRIMARY KEY (OrderID));

-- 4: Create the Employees table:
CREATE TABLE Employees(EmployeeID INT DEFAULT NULL, FullName
VARCHAR(100) DEFAULT NULL,
Role VARCHAR(50) DEFAULT NULL, Department VARCHAR(255) DEFAULT
NULL);

-- 5: Insert data into the Orders table:
INSERT INTO Orders (OrderID, ClientID, ProductID , Quantity,
Cost, Date)
VALUES
(1, "Cl1", "P1", 10, 500, "2020-09-01"),
(2, "Cl2", "P2", 5, 100, "2020-09-05"),
(3, "Cl3", "P3", 20, 800, "2020-09-03"),
(4, "Cl4", "P4", 15, 150, "2020-09-07"),
(5, "Cl3", "P3", 10, 450, "2020-09-08"),
(6, "Cl2", "P2", 5, 800, "2020-09-09"),
(7, "Cl1", "P4", 22, 1200, "2020-09-10"),
(8, "Cl3", "P1", 15, 150, "2020-09-10"),
(9, "Cl1", "P1", 10, 500, "2020-09-12"),
(10, "Cl2", "P2", 5, 100, "2020-09-13"),
(11, "Cl1", "P2", 15, 80, "2020-09-12"),
(12, "Cl1", "P1", 10, 500, "2022-09-01"),
(13, "Cl2", "P2", 5, 100, "2022-09-05"),
(14, "Cl3", "P3", 20, 800, "2022-09-03"),
(15, "Cl4", "P4", 15, 150, "2022-09-07"),
(16, "Cl3", "P3", 10, 450, "2022-09-08"),
(17, "Cl2", "P2", 5, 800, "2022-09-09"),
(18, "Cl1", "P4", 22, 1200, "2022-09-10"),
(19, "Cl3", "P1", 15, 150, "2022-09-10"),
(20, "Cl1", "P1", 10, 500, "2022-09-12"),
(21, "Cl2", "P2", 5, 100, "2022-09-13"),
(22, "Cl2", "P1", 10, 500, "2021-09-01"),
(23, "Cl2", "P2", 5, 100, "2021-09-05"),
(24, "Cl3", "P3", 20, 800, "2021-09-03"),
(25, "Cl4", "P4", 15, 150, "2021-09-07"),
(26, "Cl1", "P3", 10, 450, "2021-09-08"),
(27, "Cl2", "P1", 20, 1000, "2022-09-01"),
(28, "Cl2", "P2", 10, 200, "2022-09-05"),
(29, "Cl3", "P3", 20, 800, "2021-09-03");

-- 5: Insert data into the Employees table:
INSERT INTO Employees (EmployeeID, FullName, Role,
Department)
VALUES
(1, "Seamus Hogan", "Manager", "Management"),
(2, "Thomas Eriksson", "Assistant ", "Sales"),
(3, "Simon Tolo", "Executive", "Management"),
(4, "Francesca Soffia", "Assistant  ", "Human Resources"),
(5, "Emily Sierra", "Accountant", "Finance"),
(6, "Greta Galkina", "Accountant", "Finance"),
(7, "Maria Carter", "Executive", "Human Resources"),
(8, "Rick Griffin", "Manager", "Marketing");

-- Task 1
-- Lucky Shrub need data on client orders. They have written
the following SELECT query to retrieve
-- all data from the Orders table:

SELECT * FROM Orders;

-- Task 2
-- Lucky Shrub need to find the order placed by the client
Cl1.
-- They have written the following query to complete this
task:

SELECT * FROM Orders WHERE ClientID ='Cl1';

-- However, this query's execution plan shows that it does not
use an index to perform this search,
-- as indicated by the NULL values in possible_keys and keys
columns.

-- Task 3
-- Lucky Shrub have written the following SELECT query to find
the details of the employee whose
-- last name is 'Tolo':

SELECT * FROM Employees WHERE FullName LIKE '%Tolo';
```

- Lab 2 (File 8)

```
-- C4M2L2 Item 8 Lab: MySQL optimization techniques exercise

-- 1.Create the database:
CREATE DATABASE IF NOT EXISTS Lucky;

-- 2.Use database:
USE Lucky;

-- 3.Create the tables:
CREATE  TABLE   Orders  (OrderID  INT  NOT  NULL  PRIMARY  KEY,
ClientID  VARCHAR(10),  ProductID  VARCHAR(10),  Quantity  INT,
Cost DECIMAL(6,2), Date DATE);

CREATE TABLE Products (
    ProductID VARCHAR(10),
    ProductName VARCHAR(100),
    BuyPrice DECIMAL(6,2),
    SellPrice DECIMAL(6,2),
    NumberOfItems INT
);

CREATE TABLE Activity (
    ActivityID INT PRIMARY KEY,
    Properties JSON
);

-- 4.Populate the tables with data:
INSERT INTO Orders (OrderID, ClientID, ProductID , Quantity,
Cost, Date) VALUES
(1, "Cl1", "P1", 10, 500, "2020-09-01" ),
(2, "Cl2", "P2", 5, 100, "2020-09-05"),
(3, "Cl3", "P3", 20, 800, "2020-09-03"),
(4, "Cl4", "P4", 15, 150, "2020-09-07"),
(5, "Cl3", "P3", 10, 450, "2020-09-08"),
(6, "Cl2", "P2", 5, 800, "2020-09-09"),
(7, "Cl1", "P4", 22, 1200, "2020-09-10"),
(8, "Cl3", "P1", 15, 150, "2020-09-10"),
(9, "Cl1", "P1", 10, 500, "2020-09-12"),
(10, "Cl2", "P2", 5, 100, "2020-09-13"),
(11, "Cl4", "P5", 5, 100, "2020-09-15"),
(12, "Cl1", "P1", 10, 500, "2022-09-01"),
(13, "Cl2", "P2", 5, 100, "2022-09-05"),
(14, "Cl3", "P3", 20, 800, "2022-09-03"),
(15, "Cl4", "P4", 15, 150, "2022-09-07"),
(16, "Cl3", "P3", 10, 450, "2022-09-08"),
(17, "Cl2", "P2", 5, 800, "2022-09-09"),
(18, "Cl1", "P4", 22, 1200, "2022-09-10"),
(19, "Cl3", "P1", 15, 150, "2022-09-10"),
(20, "Cl1", "P1", 10, 500, "2022-09-12"),
(21, "Cl2", "P2", 5, 100, "2022-09-13"),
(22, "Cl2", "P1", 10, 500, "2021-09-01"),
(23, "Cl2", "P2", 5, 100, "2021-09-05"),
(24, "Cl3", "P3", 20, 800, "2021-09-03"),
(25, "Cl4", "P4", 15, 150, "2021-09-07"),
(26, "Cl1", "P3", 10, 450, "2021-09-08"),
(27, "Cl2", "P1", 20, 1000, "2022-09-01"),
(28, "Cl2", "P2", 10, 200, "2022-09-05"),
(29, "Cl3", "P3", 20, 800, "2021-09-03"),
(30, "Cl1", "P1", 10, 500, "2022-09-01");

INSERT INTO Products (ProductID, ProductName, BuyPrice,
SellPrice, NumberOfItems) VALUES
("P1", "Artificial grass bags ", 40, 50, 100),
("P2", "Wood panels", 15, 20, 250),
("P3", "Patio slates",  35, 40, 60),
("P4", "Sycamore trees ", 7, 10, 50),
("P5", "Trees and Shrubs", 35, 50, 75),
("P6", "Water fountain", 65, 80, 15);

INSERT INTO Activity(ActivityID, Properties) VALUES
(1, '{ "ClientID": "Cl1", "ProductID": "P1", "Order": "True"
}' ),
(2, '{ "ClientID": "Cl2", "ProductID": "P4", "Order": "False"
}' ),
(3, '{ "ClientID": "Cl5", "ProductID": "P5", "Order": "True"
}' );

-- Task 1
-- Lucky Shrub need to find out how many orders were placed by
clients with the following Client IDs in 2022; Cl1, Cl2 and
Cl3. They have created the following query to extract this
information.

SELECT
    CONCAT(ClientID, ": ", COUNT(OrderID), " orders") AS Result
FROM Orders
WHERE YEAR(Date) = 2022
  AND ClientID IN ("Cl1", "Cl2", "Cl3")
GROUP BY ClientID;

-- Task 2
PREPARE GetOrderDetail FROM
'SELECT OrderID, Quantity, OrderCost, Date
 FROM Orders
 WHERE ClientID = ?
 AND YEAR(Date) = ?';
SET @Client = 'Cl1';
SET @Yr = 2020;

EXECUTE GetOrderDetail USING @Client, @Yr;

-- Task 3
SELECT
    p.ProductID,
    p.ProductName,
    p.BuyPrice,
    p.SellPrice
FROM Activity a
JOIN Products p
    ON p.ProductID = a.Properties ->> '$.ProductID'
WHERE a.Properties ->> '$.Order' = 'True';
```

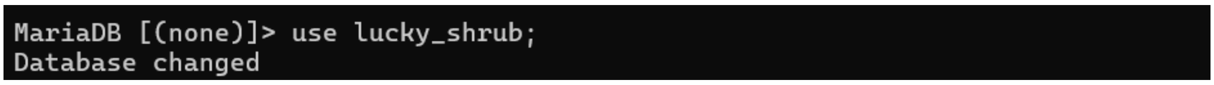
**QUIZ 5**

1. Membuat database



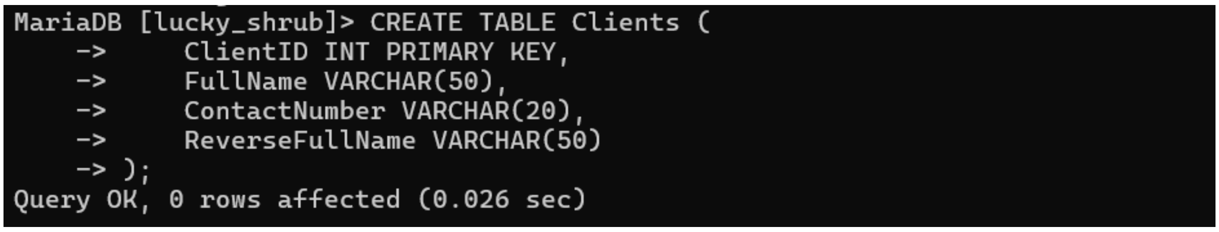
```
MariaDB [(none)]> create database lucky_shrub;
Query OK, 1 row affected (0.003 sec)
```

2. Memilih database untuk digunakan



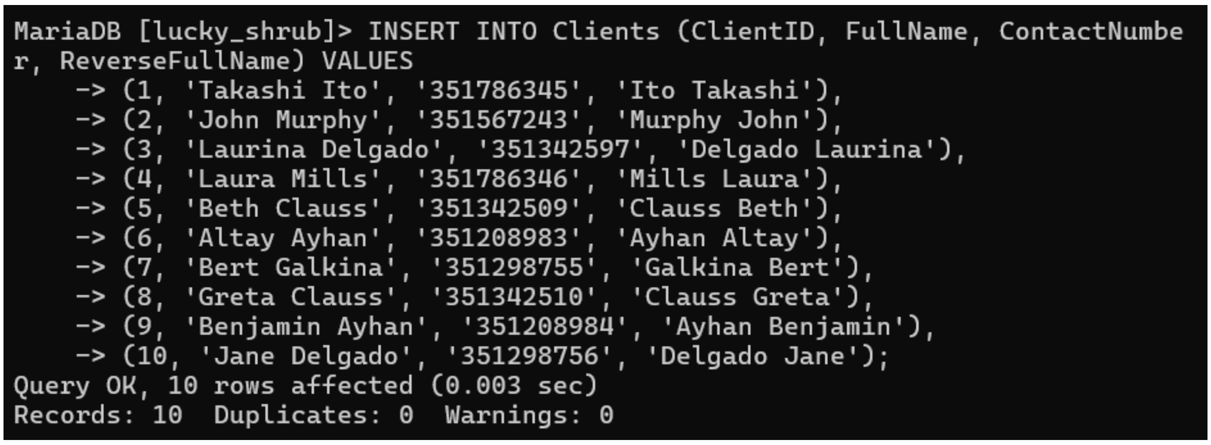
```
MariaDB [(none)]> use lucky_shrub;
Database changed
```

3. Membuat tabel Clients



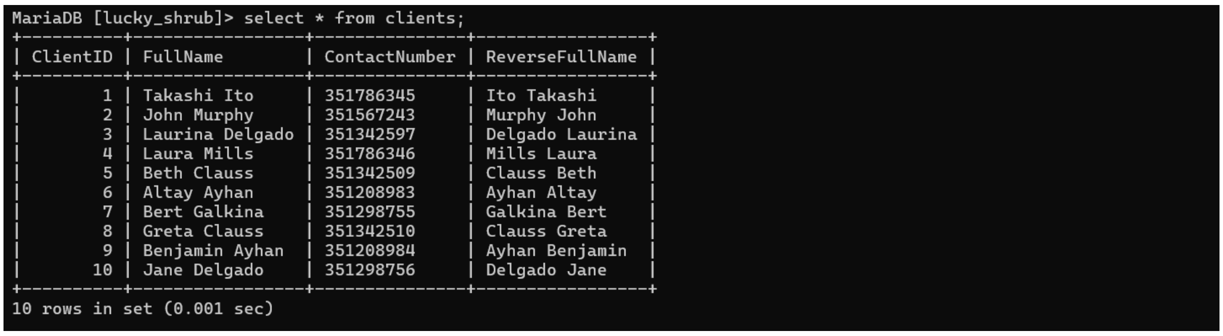
```
MariaDB [lucky_shrub]> CREATE TABLE Clients (
    ->     ClientID INT PRIMARY KEY,
    ->     FullName VARCHAR(50),
    ->     ContactNumber VARCHAR(20),
    ->     ReverseFullName VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.026 sec)
```

4. Insert data ke tabel Clients



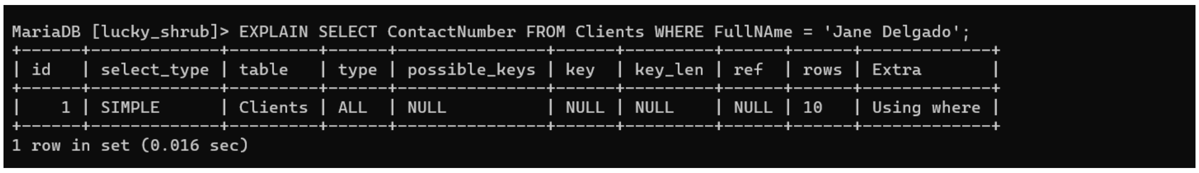
```
MariaDB [lucky_shrub]> INSERT INTO Clients (ClientID, FullName, ContactNumbe
r, ReverseFullName) VALUES
    -> (1, 'Takashi Ito', '351786345', 'Ito Takashi'),
    -> (2, 'John Murphy', '351567243', 'Murphy John'),
    -> (3, 'Laurina Delgado', '351342597', 'Delgado Laurina'),
    -> (4, 'Laura Mills', '351786346', 'Mills Laura'),
    -> (5, 'Beth Clauss', '351342509', 'Clauss Beth'),
    -> (6, 'Altay Ayhan', '351208983', 'Ayhan Altay'),
    -> (7, 'Bert Galkina', '351298755', 'Galkina Bert'),
    -> (8, 'Greta Clauss', '351342510', 'Clauss Greta'),
    -> (9, 'Benjamin Ayhan', '351208984', 'Ayhan Benjamin'),
    -> (10, 'Jane Delgado', '351298756', 'Delgado Jane');
Query OK, 10 rows affected (0.003 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

5. Cek isi tabel setelah insert



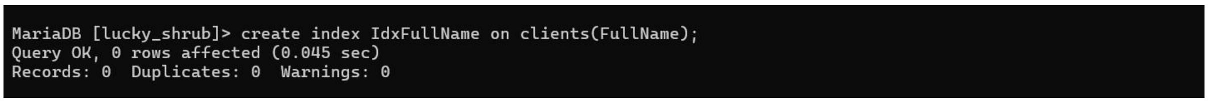
```
MariaDB [lucky_shrub]> select * from clients;
+----------+-----------------+---------------+-----------------+
| ClientID | FullName        | ContactNumber | ReverseFullName |
+----------+-----------------+---------------+-----------------+
|        1 | Takashi Ito     | 351786345     | Ito Takashi     |
|        2 | John Murphy     | 351567243     | Murphy John     |
|        3 | Laurina Delgado | 351342597     | Delgado Laurina |
|        4 | Laura Mills     | 351786346     | Mills Laura     |
|        5 | Beth Clauss     | 351342509     | Clauss Beth     |
|        6 | Altay Ayhan     | 351208983     | Ayhan Altay     |
|        7 | Bert Galkina    | 351298755     | Galkina Bert    |
|        8 | Greta Clauss    | 351342510     | Clauss Greta    |
|        9 | Benjamin Ayhan  | 351208984     | Ayhan Benjamin  |
|       10 | Jane Delgado    | 351298756     | Delgado Jane    |
+----------+-----------------+---------------+-----------------+
10 rows in set (0.001 sec)
```

6. EXPLAIN sebelum membuat index



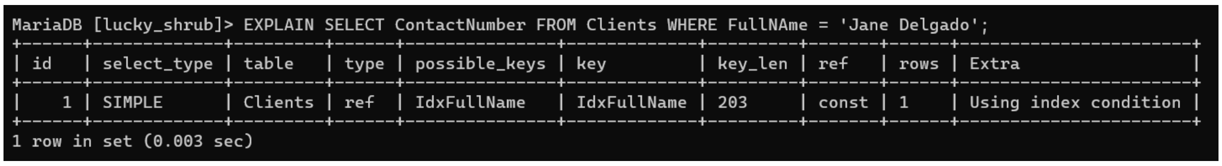
```
MariaDB [lucky_shrub]> EXPLAIN SELECT ContactNumber FROM Clients WHERE FullNAme = 'Jane Delgado';
+------+-------------+---------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table   | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+---------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Clients | ALL  | NULL          | NULL | NULL    | NULL | 10   | Using where |
+------+-------------+---------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.016 sec)
```

7. Membuat index pada kolom FullName



```
MariaDB [lucky_shrub]> create index IdxFullName on clients(FullName);
Query OK, 0 rows affected (0.045 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

8. EXPLAIN setelah membuat index



```
MariaDB [lucky_shrub]> EXPLAIN SELECT ContactNumber FROM Clients WHERE FullNAme = 'Jane Delgado';
+------+-------------+---------+------+---------------+-------------+---------+-------+------+-----------------------+
| id   | select_type | table   | type | possible_keys | key         | key_len | ref   | rows | Extra                 |
+------+-------------+---------+------+---------------+-------------+---------+-------+------+-----------------------+
|    1 | SIMPLE      | Clients | ref  | IdxFullName   | IdxFullName | 203     | const | 1    | Using index condition |
+------+-------------+---------+------+---------------+-------------+---------+-------+------+-----------------------+
1 row in set (0.003 sec)
```

Notes: Index dibuat agar pencarian berdasarkan FullName menjadi lebih cepat. Index bekerja seperti daftar isi pada buku: memudahkan pencarian.

Perubahan:

- Type = ref : Artinya query sudah memakai INDEX untuk mencari data.

- key = IdxFullName : Index yang baru kamu buat digunakan sebagai jalur pencarian.
- rows = 1 : Hanya perlu mengecek 1 baris untuk menemukan hasil, jadi jauh lebih cepat.
- Using index condition : Menandakan MariaDB memanfaatkan index secara optimal untuk melakukan filter.

- Lab 1 (File 7)

```
--  Optimizing SELECT statements in MySQL
-- 1: Create the database:
CREATE DATABASE Lucky_Shrub;

-- 2: Use the database:
USE Lucky_Shrub;

-- 3: Create the Orders table:
CREATE TABLE Orders(OrderID INT NOT NULL, ClientID VARCHAR(10)
DEFAULT NULL,
ProductID VARCHAR(10) DEFAULT NULL, Quantity INT DEFAULT NULL, Cost
DECIMAL(6,2) DEFAULT NULL,
Date DATE DEFAULT NULL, PRIMARY KEY (OrderID));

-- 4: Create the Employees table:
CREATE TABLE Employees(EmployeeID INT DEFAULT NULL, FullName
VARCHAR(100) DEFAULT NULL,
Role VARCHAR(50) DEFAULT NULL, Department VARCHAR(255) DEFAULT
NULL);

-- 5: Insert data into the Orders table:
INSERT INTO Orders (OrderID, ClientID, ProductID , Quantity, Cost, Date)
VALUES
(1, "Cl1", "P1", 10, 500, "2020-09-01"),
(2, "Cl2", "P2", 5, 100, "2020-09-05"),
(3, "Cl3", "P3", 20, 800, "2020-09-03"),
(4, "Cl4", "P4", 15, 150, "2020-09-07"),
(5, "Cl3", "P3", 10, 450, "2020-09-08"),
(6, "Cl2", "P2", 5, 800, "2020-09-09"),
(7, "Cl1", "P4", 22, 1200, "2020-09-10"),
(8, "Cl3", "P1", 15, 150, "2020-09-10"),
(9, "Cl1", "P1", 10, 500, "2020-09-12"),
(10, "Cl2", "P2", 5, 100, "2020-09-13"),
(11, "Cl1", "P2", 15, 80, "2020-09-12"),
(12, "Cl1", "P1", 10, 500, "2022-09-01"),
(13, "Cl2", "P2", 5, 100, "2022-09-05"),
(14, "Cl3", "P3", 20, 800, "2022-09-03"),
(15, "Cl4", "P4", 15, 150, "2022-09-07"),
(16, "Cl3", "P3", 10, 450, "2022-09-08"),
(17, "Cl2", "P2", 5, 800, "2022-09-09"),
(18, "Cl1", "P4", 22, 1200, "2022-09-10"),
(19, "Cl3", "P1", 15, 150, "2022-09-10"),
(20, "Cl1", "P1", 10, 500, "2022-09-12"),
(21, "Cl2", "P2", 5, 100, "2022-09-13"),
(22, "Cl2", "P1", 10, 500, "2021-09-01"),
(23, "Cl2", "P2", 5, 100, "2021-09-05"),
(24, "Cl3", "P3", 20, 800, "2021-09-03"),
(25, "Cl4", "P4", 15, 150, "2021-09-07"),
(26, "Cl1", "P3", 10, 450, "2021-09-08"),
(27, "Cl2", "P1", 20, 1000, "2022-09-01"),
(28, "Cl2", "P2", 10, 200, "2022-09-05"),
(29, "Cl3", "P3", 20, 800, "2021-09-03");

-- 5: Insert data into the Employees table:
INSERT INTO Employees (EmployeeID, FullName,  Role, Department)
VALUES
(1, "Seamus Hogan", "Manager", "Management"),
(2, "Thomas Eriksson", "Assistant ", "Sales"),
(3, "Simon Tolo", "Executive", "Management"),
(4, "Francesca Soffia", "Assistant  ", "Human Resources"),
(5, "Emily Sierra", "Accountant", "Finance"),
(6, "Greta Galkina", "Accountant", "Finance"),
(7, "Maria Carter", "Executive", "Human Resources"),
(8, "Rick Griffin", "Manager", "Marketing");

-- Task 1
-- Lucky Shrub need data on client orders. They have written the following SELECT query
to retrieve
-- all data from the Orders table:

SELECT * FROM Orders;

-- Task 2
-- Lucky Shrub need to find the order placed by the client Cl1.
-- They have written the following query to complete this task:

SELECT * FROM Orders WHERE ClientID ='Cl1';

-- However, this query's execution plan shows that it does not use an index to perform this
search,
-- as indicated by the NULL values in possible_keys and keys columns.

-- Task 3
-- Lucky Shrub have written the following SELECT query to find the details of the
employee whose
-- last name is 'Tolo':

SELECT * FROM Employees WHERE FullName LIKE '%Tolo';
```

- Lab 2 (File 8)

```
-- C4M2L2 Item 8 Lab: MySQL optimization techniques exercise

-- 1.Create the database:
CREATE DATABASE IF NOT EXISTS Lucky;

-- 2.Use database:
USE Lucky;

-- 3.Create the tables:
CREATE TABLE  Orders (OrderID INT NOT NULL PRIMARY KEY, ClientID
VARCHAR(10), ProductID VARCHAR(10), Quantity INT, Cost DECIMAL(6,2), Date
DATE);

CREATE TABLE Products (
   ProductID VARCHAR(10),
   ProductName VARCHAR(100),
   BuyPrice DECIMAL(6,2),
   SellPrice DECIMAL(6,2),
   NumberOfItems INT
);

CREATE TABLE Activity (
   ActivityID INT PRIMARY KEY,
   Properties JSON
);

-- 4.Populate the tables with data:
INSERT INTO Orders (OrderID, ClientID, ProductID , Quantity, Cost, Date) VALUES
(1, "Cl1", "P1", 10, 500, "2020-09-01" ),
(2, "Cl2", "P2", 5, 100, "2020-09-05"),
(3, "Cl3", "P3", 20, 800, "2020-09-03"),
(4, "Cl4", "P4", 15, 150, "2020-09-07"),
(5, "Cl3", "P3", 10, 450, "2020-09-08"),
(6, "Cl2", "P2", 5, 800, "2020-09-09"),
(7, "Cl1", "P4", 22, 1200, "2020-09-10"),
(8, "Cl3", "P1", 15, 150, "2020-09-10"),
(9, "Cl1", "P1", 10, 500, "2020-09-12"),
(10, "Cl2", "P2", 5, 100, "2020-09-13"),
(11, "Cl4", "P5", 5, 100, "2020-09-15"),
(12, "Cl1", "P1", 10, 500, "2022-09-01"),
(13, "Cl2", "P2", 5, 100, "2022-09-05"),
(14, "Cl3", "P3", 20, 800, "2022-09-03"),
(15, "Cl4", "P4", 15, 150, "2022-09-07"),
(16, "Cl3", "P3", 10, 450, "2022-09-08"),
(17, "Cl2", "P2", 5, 800, "2022-09-09"),
(18, "Cl1", "P4", 22, 1200, "2022-09-10"),
(19, "Cl3", "P1", 15, 150, "2022-09-10"),
(20, "Cl1", "P1", 10, 500, "2022-09-12"),
(21, "Cl2", "P2", 5, 100, "2022-09-13"),
(22, "Cl2", "P1", 10, 500, "2021-09-01"),
(23, "Cl2", "P2", 5, 100, "2021-09-05"),
(24, "Cl3", "P3", 20, 800, "2021-09-03"),
(25, "Cl4", "P4", 15, 150, "2021-09-07"),
(26, "Cl1", "P3", 10, 450, "2021-09-08"),
(27, "Cl2", "P1", 20, 1000, "2022-09-01"),
(28, "Cl2", "P2", 10, 200, "2022-09-05"),
(29, "Cl3", "P3", 20, 800, "2021-09-03"),
(30, "Cl1", "P1", 10, 500, "2022-09-01");

INSERT INTO Products (ProductID, ProductName, BuyPrice, SellPrice, NumberOfItems)
VALUES
("P1", "Artificial grass bags ", 40, 50, 100),
("P2", "Wood panels", 15, 20, 250),
("P3", "Patio slates",  35, 40, 60),
("P4", "Sycamore trees ", 7, 10, 50),
("P5", "Trees and Shrubs", 35, 50, 75),
("P6", "Water fountain", 65, 80, 15);

INSERT INTO Activity(ActivityID, Properties) VALUES
(1, '{ "ClientID": "Cl1", "ProductID": "P1", "Order": "True" }' ),
(2, '{ "ClientID": "Cl2", "ProductID": "P4", "Order": "False" }' ),
(3, '{ "ClientID": "Cl5", "ProductID": "P5", "Order": "True" }' );

-- Task 1
-- Lucky Shrub need to find out how many orders were placed by clients with the following
Client IDs in 2022; Cl1, Cl2 and Cl3. They have created the following query to extract this
information.

SELECT
    CONCAT(ClientID, ": ", COUNT(OrderID), " orders") AS Result
FROM Orders
WHERE YEAR(Date) = 2022
  AND ClientID IN ("Cl1", "Cl2", "Cl3")
GROUP BY ClientID;



-- Task 2
PREPARE GetOrderDetail FROM
'SELECT OrderID, Quantity, OrderCost, Date
 FROM Orders
 WHERE ClientID = ?
 AND YEAR(Date) = ?';
SET @Client = 'Cl1';
SET @Yr = 2020;

EXECUTE GetOrderDetail USING @Client, @Yr;

-- Task 3
SELECT
```

```
   p.ProductID,
   p.ProductName,
   p.BuyPrice,
   p.SellPrice
FROM Activity a
JOIN Products p
   ON p.ProductID = a.Properties ->> '$.ProductID'
WHERE a.Properties ->> '$.Order' = 'True';
```

**QUIZ 6**

1. Database optimization involves optimizing only the data retrieval statements that are executed against a database.

   a. True

   b. False

   **Answer: b. False**

2. Which of the following guidelines should you follow when optimizing SQL select statements? Select all that apply.

   Use OUTER JOIN instead of INNER JOIN wherever possible.

   Avoid using leading wildcards in predicates.

   Avoid using trailing wildcards in predicates.

   Avoid using unnecessary columns in the SELECT clause.

   Use INNER JOIN instead of OUTER JOIN wherever possible.

   **Answer: Avoid using leading wildcards in predicates.**

   **Avoid using trailing wildcards in predicates.**

   **Avoid using unnecessary columns in the SELECT clause.**

**Use INNER JOIN instead of OUTER JOIN wherever possible.**

3. Identify the key characteristics of a Primary Index. Select all that apply.

   A Primary Index is created using the CREATE INDEX syntax.

   A Primary Key is also known as a Non-Clustered index.

   A Primary Key is also known as a Clustered index.

   A Primary Index is created automatically when a table is created.

   **Answer: A Primary Key is also known as a Clustered index.**

   **A Primary Index is created automatically when a table is created.**

4. The following query is executed on a table named Clients that contains the fields ClientID, FullName and ContactNumber.

   ```
   SELECT ContactNumber
   FROM Clients
   WHERE FullName='Client Name here';
   ```

   On which field within the table should a Secondary Index be created to optimize this query?

   a. FullName

   b. ContactNumber

   c. ClientID

   **Answer: a. FullName**

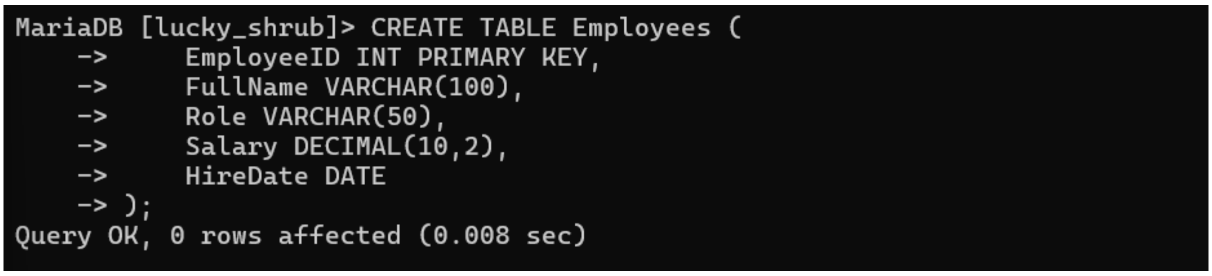
5. What keyword must be added before the following query to view its MySQL query execution plan?

   ```
   _____________ SELECT *
   FROM Orders
   WHERE ClientID='Cl1';
   ```

   **Answer: <u>EXPLAIN</u> SELECT * FROM Orders WHERE ClientID='Cl1';**

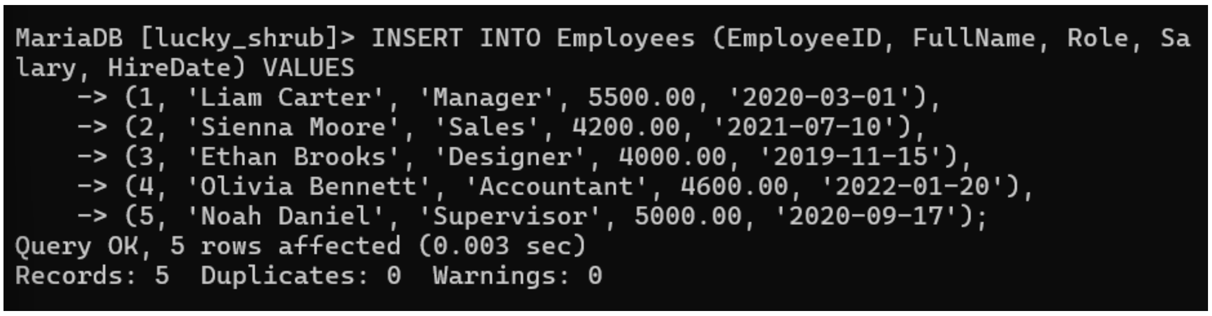
**QUIZ 7**

1. Membuat Tabel Employees



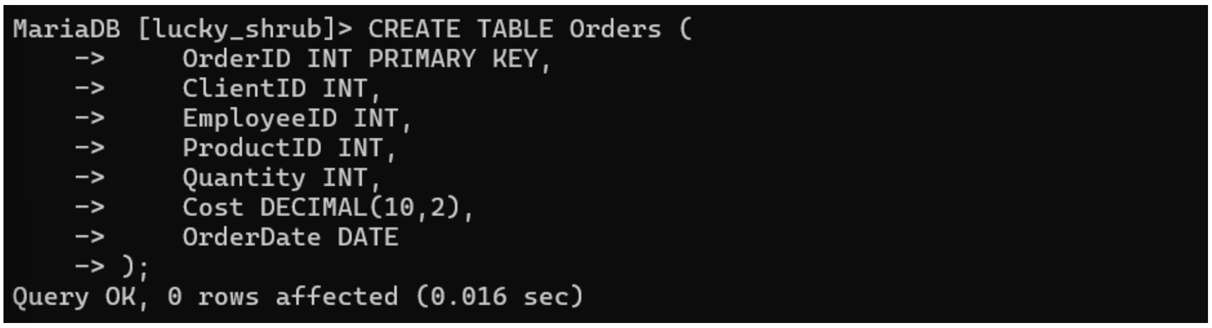
```
MariaDB [lucky_shrub]> CREATE TABLE Employees (
    ->     EmployeeID INT PRIMARY KEY,
    ->     FullName VARCHAR(100),
    ->     Role VARCHAR(50),
    ->     Salary DECIMAL(10,2),
    ->     HireDate DATE
    -> );
Query OK, 0 rows affected (0.008 sec)
```

2. Insert Data Employees



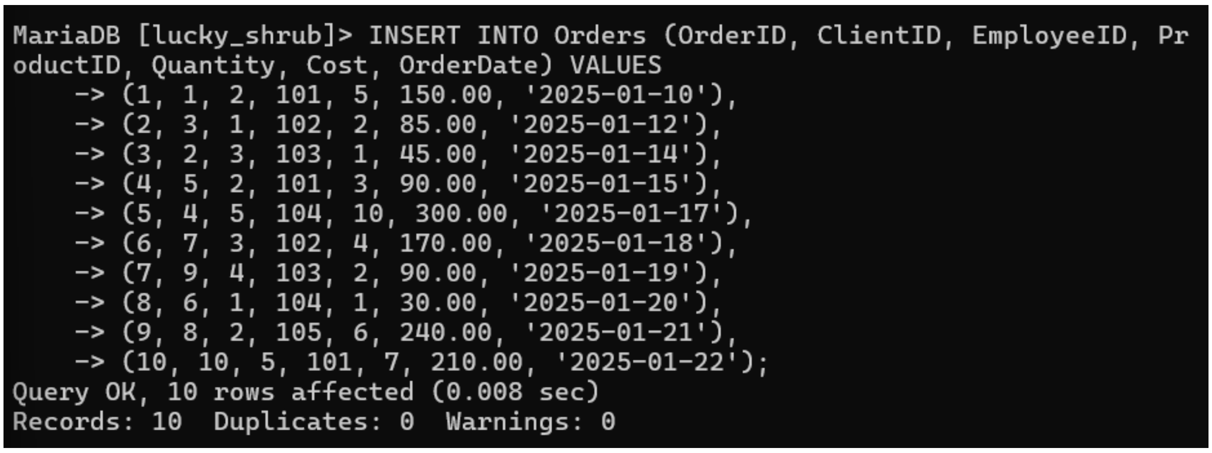
```
MariaDB [lucky_shrub]> INSERT INTO Employees (EmployeeID, FullName, Role, Sa
lary, HireDate) VALUES
    -> (1, 'Liam Carter', 'Manager', 5500.00, '2020-03-01'),
    -> (2, 'Sienna Moore', 'Sales', 4200.00, '2021-07-10'),
    -> (3, 'Ethan Brooks', 'Designer', 4000.00, '2019-11-15'),
    -> (4, 'Olivia Bennett', 'Accountant', 4600.00, '2022-01-20'),
    -> (5, 'Noah Daniel', 'Supervisor', 5000.00, '2020-09-17');
Query OK, 5 rows affected (0.003 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

3. Membuat Tabel Orders



```
MariaDB [lucky_shrub]> CREATE TABLE Orders (
    ->     OrderID INT PRIMARY KEY,
    ->     ClientID INT,
    ->     EmployeeID INT,
    ->     ProductID INT,
    ->     Quantity INT,
    ->     Cost DECIMAL(10,2),
    ->     OrderDate DATE
    -> );
Query OK, 0 rows affected (0.016 sec)
```

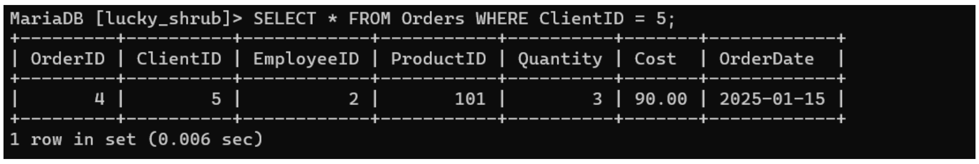
4. Insert Data Orders



```
MariaDB [lucky_shrub]> INSERT INTO Orders (OrderID, ClientID, EmployeeID, Pr
oductID, Quantity, Cost, OrderDate) VALUES
    -> (1, 1, 2, 101, 5, 150.00, '2025-01-10'),
    -> (2, 3, 1, 102, 2, 85.00, '2025-01-12'),
    -> (3, 2, 3, 103, 1, 45.00, '2025-01-14'),
    -> (4, 5, 2, 101, 3, 90.00, '2025-01-15'),
    -> (5, 4, 5, 104, 10, 300.00, '2025-01-17'),
    -> (6, 7, 3, 102, 4, 170.00, '2025-01-18'),
    -> (7, 9, 4, 103, 2, 90.00, '2025-01-19'),
    -> (8, 6, 1, 104, 1, 30.00, '2025-01-20'),
    -> (9, 8, 2, 105, 6, 240.00, '2025-01-21'),
    -> (10, 10, 5, 101, 7, 210.00, '2025-01-22');
Query OK, 10 rows affected (0.008 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

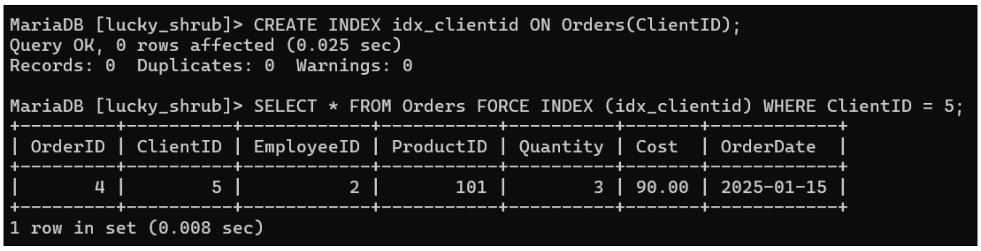
**Bagian Optimasi SELECT**

1. **Mencari semua pesanan oleh ClientID = 5**
   a. Sebelum Optimasi



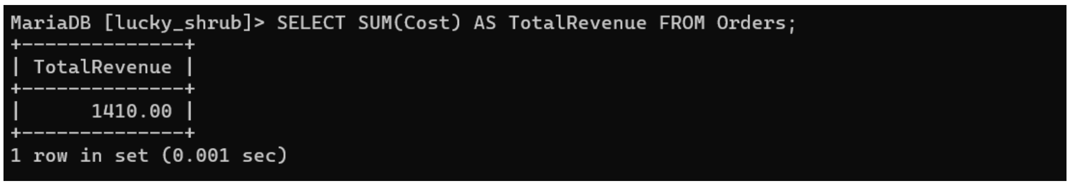
```
MariaDB [lucky_shrub]> SELECT * FROM Orders WHERE ClientID = 5;
+---------+----------+------------+-----------+----------+-------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost  | OrderDate  |
+---------+----------+------------+-----------+----------+-------+------------+
|       4 |        5 |          2 |       101 |        3 | 90.00 | 2025-01-15 |
+---------+----------+------------+-----------+----------+-------+------------+
1 row in set (0.006 sec)
```

   b. Sesudah Optimasi



```
MariaDB [lucky_shrub]> CREATE INDEX idx_clientid ON Orders(ClientID);
Query OK, 0 rows affected (0.025 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [lucky_shrub]> SELECT * FROM Orders FORCE INDEX (idx_clientid) WHERE ClientID = 5;
+---------+----------+------------+-----------+----------+-------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost  | OrderDate  |
+---------+----------+------------+-----------+----------+-------+------------+
|       4 |        5 |          2 |       101 |        3 | 90.00 | 2025-01-15 |
+---------+----------+------------+-----------+----------+-------+------------+
1 row in set (0.008 sec)
```

**2. Hitung total pendapatan dari semua pesanan**

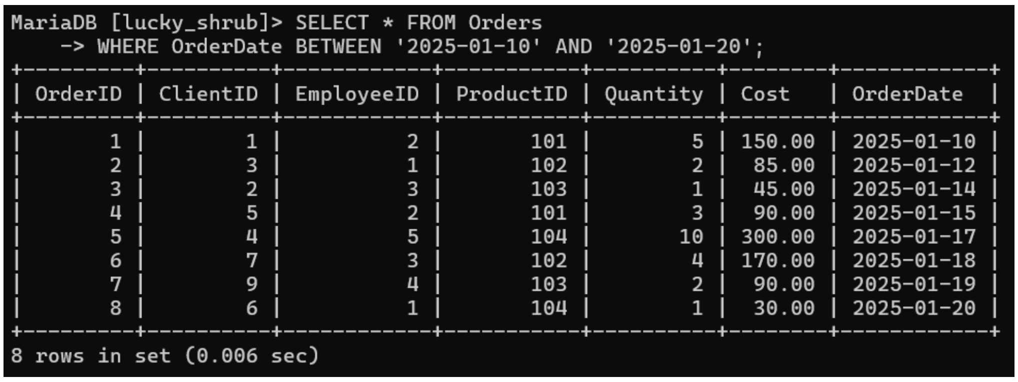


```
MariaDB [lucky_shrub]> SELECT SUM(Cost) AS TotalRevenue FROM Orders;
+--------------+
| TotalRevenue |
+--------------+
|      1410.00 |
+--------------+
1 row in set (0.001 sec)
```

Notes: Query ini sudah optimal (MySQL dapat melakukan full table scan cepat).

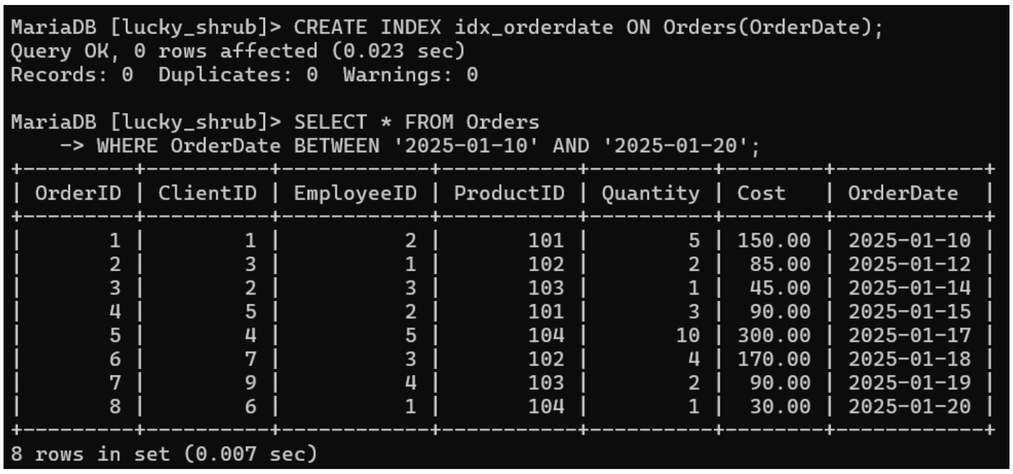
**3. Daftar pesanan dalam rentang tanggal**

a. Sebelum Optimasi



```
MariaDB [lucky_shrub]> SELECT * FROM Orders
    -> WHERE OrderDate BETWEEN '2025-01-10' AND '2025-01-20';
+---------+----------+------------+-----------+----------+--------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost   | OrderDate  |
+---------+----------+------------+-----------+----------+--------+------------+
|       1 |        1 |          2 |       101 |        5 | 150.00 | 2025-01-10 |
|       2 |        3 |          1 |       102 |        2 |  85.00 | 2025-01-12 |
|       3 |        2 |          3 |       103 |        1 |  45.00 | 2025-01-14 |
|       4 |        5 |          2 |       101 |        3 |  90.00 | 2025-01-15 |
|       5 |        4 |          5 |       104 |       10 | 300.00 | 2025-01-17 |
|       6 |        7 |          3 |       102 |        4 | 170.00 | 2025-01-18 |
|       7 |        9 |          4 |       103 |        2 |  90.00 | 2025-01-19 |
|       8 |        6 |          1 |       104 |        1 |  30.00 | 2025-01-20 |
+---------+----------+------------+-----------+----------+--------+------------+
8 rows in set (0.006 sec)
```

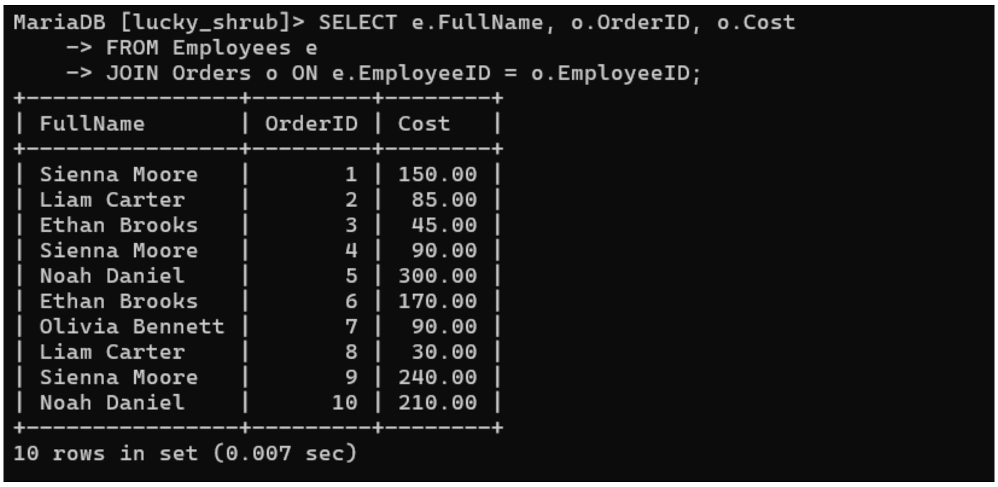
b. Sesudah Optimasi



```
MariaDB [lucky_shrub]> CREATE INDEX idx_orderdate ON Orders(OrderDate);
Query OK, 0 rows affected (0.023 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [lucky_shrub]> SELECT * FROM Orders
    -> WHERE OrderDate BETWEEN '2025-01-10' AND '2025-01-20';
+---------+----------+------------+-----------+----------+--------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost   | OrderDate  |
+---------+----------+------------+-----------+----------+--------+------------+
|       1 |        1 |          2 |       101 |        5 | 150.00 | 2025-01-10 |
|       2 |        3 |          1 |       102 |        2 |  85.00 | 2025-01-12 |
|       3 |        2 |          3 |       103 |        1 |  45.00 | 2025-01-14 |
|       4 |        5 |          2 |       101 |        3 |  90.00 | 2025-01-15 |
|       5 |        4 |          5 |       104 |       10 | 300.00 | 2025-01-17 |
|       6 |        7 |          3 |       102 |        4 | 170.00 | 2025-01-18 |
|       7 |        9 |          4 |       103 |        2 |  90.00 | 2025-01-19 |
|       8 |        6 |          1 |       104 |        1 |  30.00 | 2025-01-20 |
+---------+----------+------------+-----------+----------+--------+------------+
8 rows in set (0.007 sec)
```

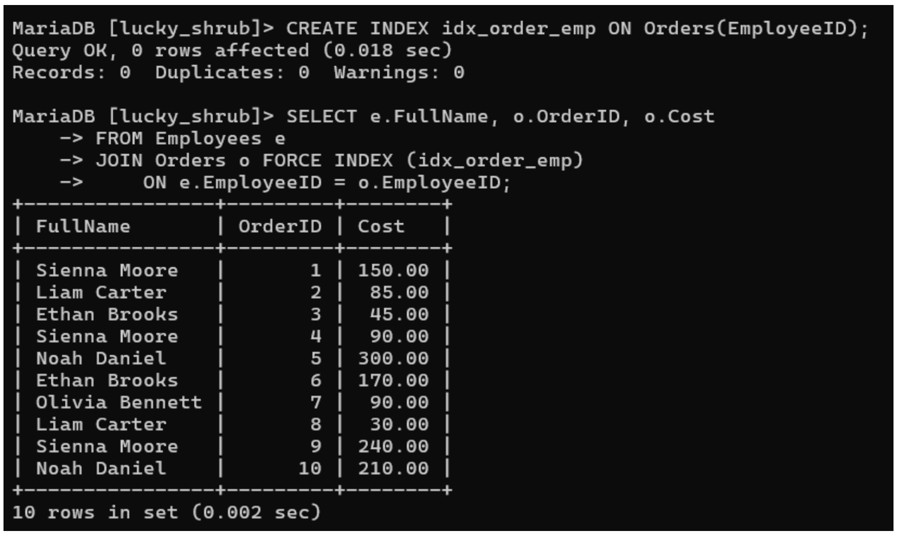
**4. Join Employees & Orders menunjukkan siapa yang melayani pesanan**

a. Sebelum Optimasi



```
MariaDB [lucky_shrub]> SELECT e.FullName, o.OrderID, o.Cost
    -> FROM Employees e
    -> JOIN Orders o ON e.EmployeeID = o.EmployeeID;
+----------------+---------+--------+
| FullName       | OrderID | Cost   |
+----------------+---------+--------+
| Sienna Moore   |       1 | 150.00 |
| Liam Carter    |       2 |  85.00 |
| Ethan Brooks   |       3 |  45.00 |
| Sienna Moore   |       4 |  90.00 |
| Noah Daniel    |       5 | 300.00 |
| Ethan Brooks   |       6 | 170.00 |
| Olivia Bennett |       7 |  90.00 |
| Liam Carter    |       8 |  30.00 |
| Sienna Moore   |       9 | 240.00 |
| Noah Daniel    |      10 | 210.00 |
+----------------+---------+--------+
10 rows in set (0.007 sec)
```

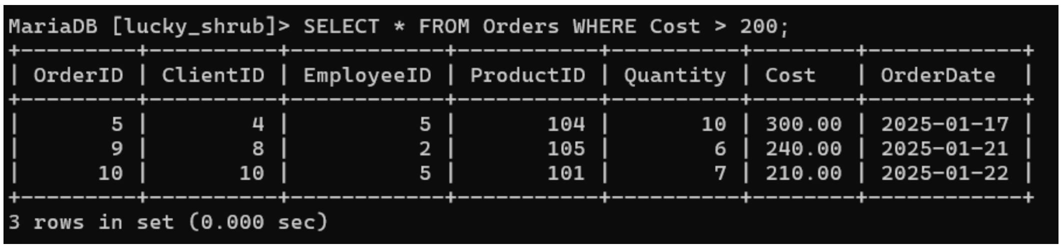
b. Sesudah Optimasi



```
MariaDB [lucky_shrub]> CREATE INDEX idx_order_emp ON Orders(EmployeeID);
Query OK, 0 rows affected (0.018 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [lucky_shrub]> SELECT e.FullName, o.OrderID, o.Cost
    -> FROM Employees e
    -> JOIN Orders o FORCE INDEX (idx_order_emp)
    ->     ON e.EmployeeID = o.EmployeeID;
+----------------+---------+--------+
| FullName       | OrderID | Cost   |
+----------------+---------+--------+
| Sienna Moore   |       1 | 150.00 |
| Liam Carter    |       2 |  85.00 |
| Ethan Brooks   |       3 |  45.00 |
| Sienna Moore   |       4 |  90.00 |
| Noah Daniel    |       5 | 300.00 |
| Ethan Brooks   |       6 | 170.00 |
| Olivia Bennett |       7 |  90.00 |
| Liam Carter    |       8 |  30.00 |
| Sienna Moore   |       9 | 240.00 |
| Noah Daniel    |      10 | 210.00 |
+----------------+---------+--------+
10 rows in set (0.002 sec)
```

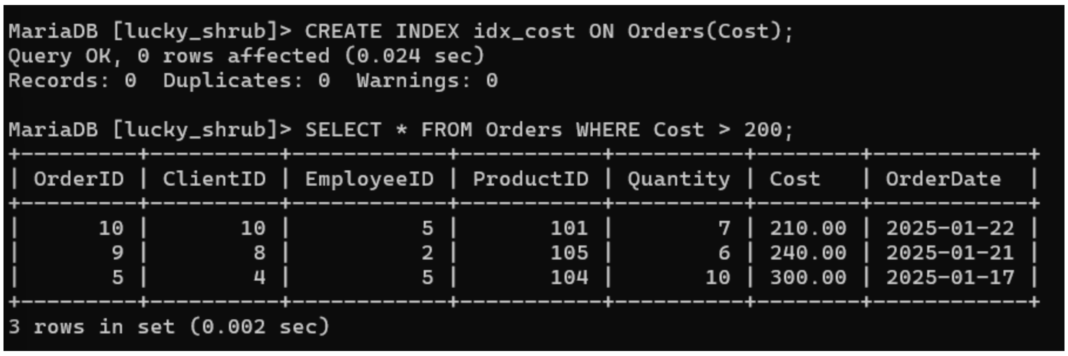
**5. Cari pesanan dengan biaya di atas 200**

a. Sebelum Optimasi



```
MariaDB [lucky_shrub]> SELECT * FROM Orders WHERE Cost > 200;
+---------+----------+------------+-----------+----------+--------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost   | OrderDate  |
+---------+----------+------------+-----------+----------+--------+------------+
|       5 |        4 |          5 |       104 |       10 | 300.00 | 2025-01-17 |
|       9 |        8 |          2 |       105 |        6 | 240.00 | 2025-01-21 |
|      10 |       10 |          5 |       101 |        7 | 210.00 | 2025-01-22 |
+---------+----------+------------+-----------+----------+--------+------------+
3 rows in set (0.000 sec)
```

b. Sesudah  Optimasi



```
MariaDB [lucky_shrub]> CREATE INDEX idx_cost ON Orders(Cost);
Query OK, 0 rows affected (0.024 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [lucky_shrub]> SELECT * FROM Orders WHERE Cost > 200;
+---------+----------+------------+-----------+----------+--------+------------+
| OrderID | ClientID | EmployeeID | ProductID | Quantity | Cost   | OrderDate  |
+---------+----------+------------+-----------+----------+--------+------------+
|      10 |       10 |          5 |       101 |        7 | 210.00 | 2025-01-22 |
|       9 |        8 |          2 |       105 |        6 | 240.00 | 2025-01-21 |
|       5 |        4 |          5 |       104 |       10 | 300.00 | 2025-01-17 |
+---------+----------+------------+-----------+----------+--------+------------+
3 rows in set (0.002 sec)
```

**QUIZ 8**

1. In the first task, you optimized the following SELECT query to extract data from the Orders table:

   `SELECT * FROM Orders;`

   What action did you take to optimize this query?

   a. You reduced the number of rows by adding filter criteria.

   b. You added an index to the OrderID column.

   c. You added an index to the ProductID column.

   d. You rewrote the query to avoid the use of *.

   **Answer: d. You rewrote the query to avoid the use of**

2. In the second task, you helped Lucky Shrub retrieve the order placed by the client with the ID of Cl1. You performed this task by creating an index on the Orders table to help optimize the following query:

   ```
   SELECT *
   FROM Orders
   WHERE ClientID = 'Cl1';
   ```

   On which column in the Orders table did you create the index?

   a. ClientID column

   b. OrderID column

   c. Quantity column

   d. ProductID column

   **Answer: a. ClientID column**

3. In the second task, you reviewed a query EXPLAIN plan before optimizing the query. The plan indicated NULL values in the `possible_keys` and `keys` columns. This suggests that there is no index in the targeted table which can be used to perform the search.

   a. True

   b. False

   **Answer: a. True**

4. In the third task you performed three steps on the Orders table and then rewrote the SELECT query to use a trailing wildcard. In the third task, Lucky Shrub used the following SELECT query, which contains a leading wildcard, to find the details of the employee whose last name is ‘Tolo’;

```
SELECT *
FROM Employees
WHERE FullName LIKE '%Tolo';
```

You helped to optimize this query by replacing the leading wildcard with a trailing wildcard. Why is the use of a trailing wildcard a more optimal approach?

a. A trailing wild card provides more accurate results when the SQL query is executed.

b. With a trailing wild card, MySQL can make use of an index created on the column to which the wildcard is assigned.

c. A trailing wildcard needs to match a smaller number of rows when compared to a leading
wildcard.

d. A trailing wild card takes less time to identify matches when compared to a leading wildcard.

**Answer: b. With a trailing wild card, MySQL can make use of an index created on the column to which the wildcard is assigned.**