

Assignment 05:

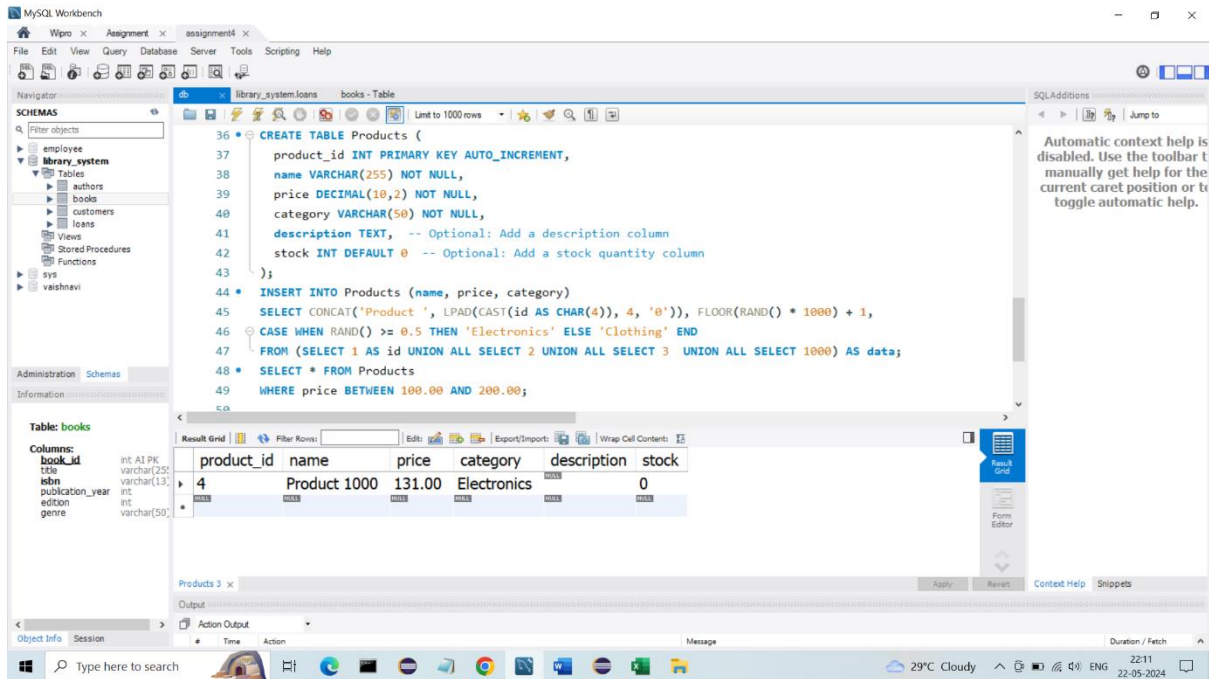
Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

In relational databases, indexes act like an organized filing system for tables. They significantly improve the performance of queries that involve filtering or sorting data based on specific columns. Here's an example:

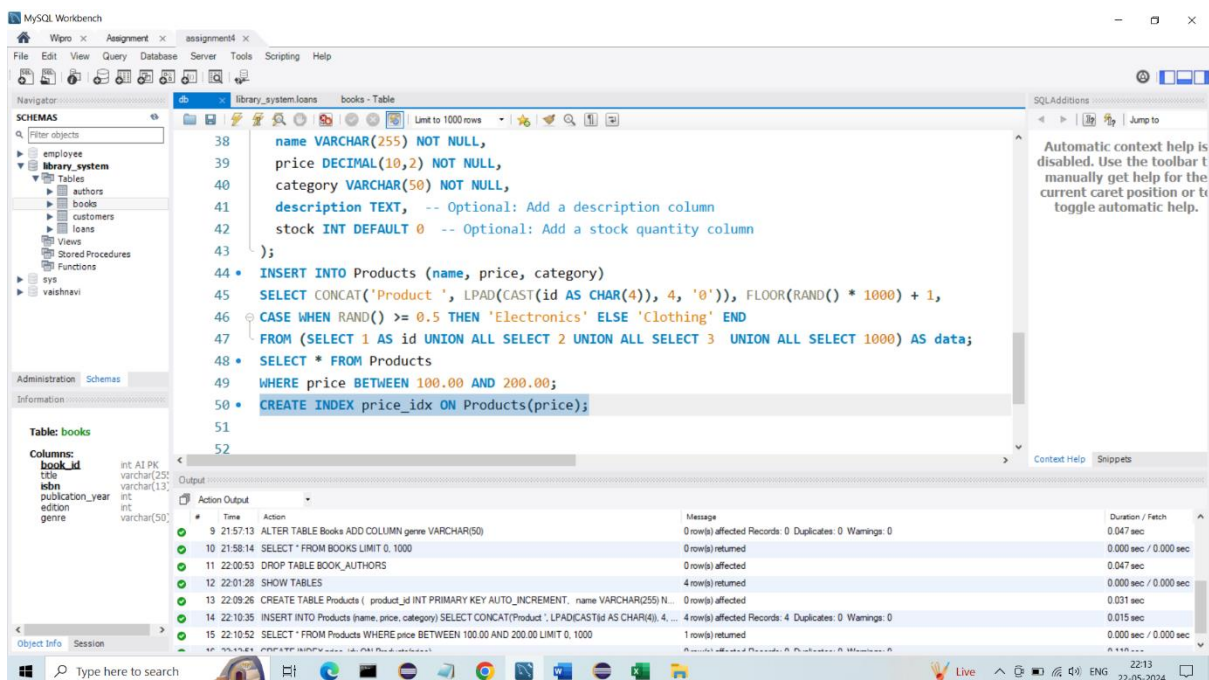
Scenario:

Consider a large table named Products with many columns, including product_id (primary key), name, price, and category. Imagine a query that retrieves all products within a specific price range (price BETWEEN 100 AND 200).

1.Create a table: -



2. Creating an Index:



Impact on Query Performance:

Now, running the same query that filters by price range will utilize the index. The database engine can

efficiently navigate the index to locate products within the specified range, significantly reducing the time it takes to execute the query.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
44 • INSERT INTO Products (name, price, category)
45 • SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, '0')), FLOOR(RAND() * 1000) + 1,
46 • CASE WHEN RAND() >= 0.5 THEN 'Electronics' ELSE 'Clothing' END
47 • FROM (SELECT 1 AS id UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 1000) AS data;
48 • SELECT * FROM Products
49 • WHERE price BETWEEN 100.00 AND 200.00;
50 • CREATE INDEX price_idx ON Products(price);
51
52
```

The Results tab shows the output of the queries:

product_id	name	price	category	description	stock
4	Product 1000	131.00	Electronics		0

The Action Output tab shows the execution of the queries:

#	Time	Action	Message	Duration / Fetch
11	22:00:53	DROP TABLE BOOK_AUTHORS	0 row(s) affected	0.047 sec
12	22:01:28	SHOW TABLES	4 row(s) returned	0.000 sec / 0.000 sec
13	22:09:26	CREATE TABLE Products (product_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) N...	0 row(s) affected	0.031 sec
14	22:10:35	INSERT INTO Products (name, price, category) SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, ...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.015 sec
15	22:10:52	SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	22:12:51	CREATE INDEX price_idx ON Products(price)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.110 sec
17	22:14:39	SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

3. Dropping the Index: -

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
44 • INSERT INTO Products (name, price, category)
45 • SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, '0')), FLOOR(RAND() * 1000) + 1,
46 • CASE WHEN RAND() >= 0.5 THEN 'Electronics' ELSE 'Clothing' END
47 • FROM (SELECT 1 AS id UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 1000) AS data;
48 • SELECT * FROM Products
49 • WHERE price BETWEEN 100.00 AND 200.00;
50 • CREATE INDEX price_idx ON Products(price);
51 • DROP INDEX price_idx ON Products;
52
```

The Results tab shows the output of the queries:

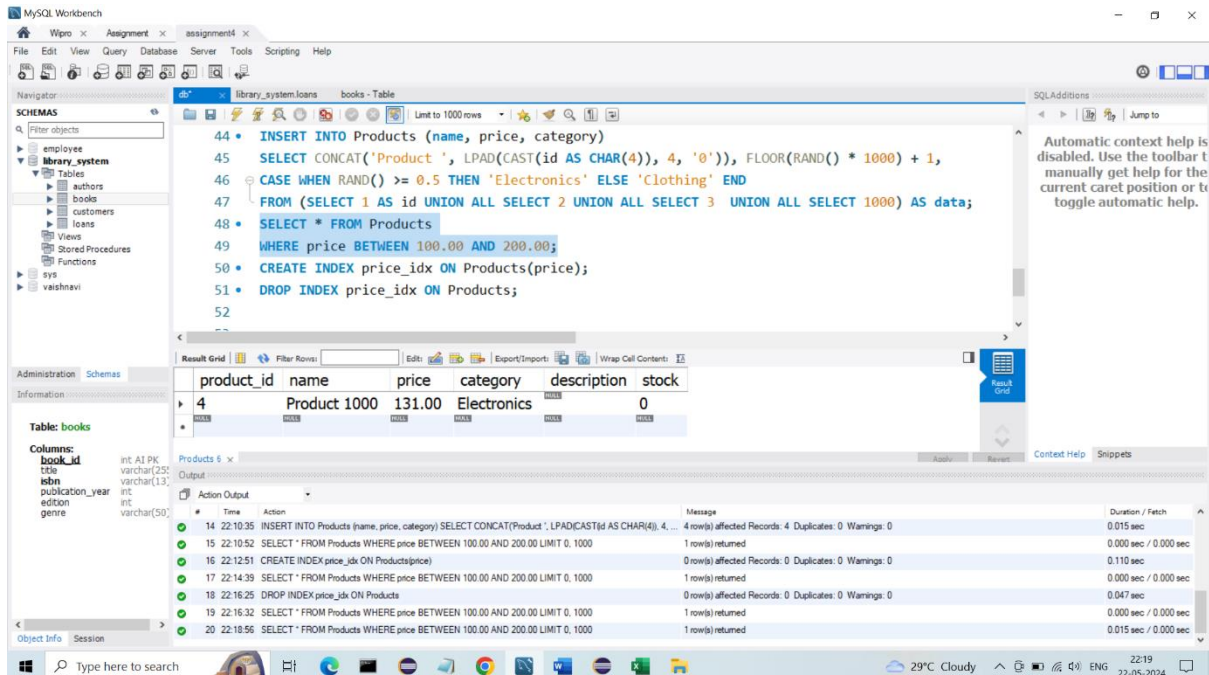
product_id	name	price	category	description	stock
4	Product 1000	131.00	Electronics		0

The Action Output tab shows the execution of the queries:

#	Time	Action	Message	Duration / Fetch
13	22:09:26	CREATE TABLE Products (product_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) N...	0 row(s) affected	0.031 sec
14	22:10:35	INSERT INTO Products (name, price, category) SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, ...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.015 sec
15	22:10:52	SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	22:12:51	CREATE INDEX price_idx ON Products(price)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.110 sec
17	22:14:39	SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
18	22:16:25	DROP INDEX price_idx ON Products	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
19	22:16:32	SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Re-running the Query:

Without the index, the database engine reverts to scanning the entire Products table, potentially leading to slower query execution, especially for large datasets.



The screenshot displays the MySQL Workbench interface. The main window shows a SQL query being executed in the 'Query' tab. The query is as follows:

```
44 INSERT INTO Products (name, price, category)
45 SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, '0')), FLOOR(RAND() * 1000) + 1,
46 CASE WHEN RAND() >= 0.5 THEN 'Electronics' ELSE 'Clothing' END
47 FROM (SELECT 1 AS id UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 1000) AS data;
48 SELECT * FROM Products
49 WHERE price BETWEEN 100.00 AND 200.00;
50 CREATE INDEX price_idx ON Products(price);
51 DROP INDEX price_idx ON Products;
52
```

The 'Result Grid' tab shows the results of the query. The first result is a table with 6 columns: product_id, name, price, category, description, and stock. The data is as follows:

product_id	name	price	category	description	stock
4	Product 1000	131.00	Electronics		0

The 'Action Output' tab shows the execution log, including the following actions and messages:

- 14 22:10:35 INSERT INTO Products (name, price, category) SELECT CONCAT('Product ', LPAD(CAST(id AS CHAR(4)), 4, '0')), FLOOR(RAND() * 1000) + 1, CASE WHEN RAND() >= 0.5 THEN 'Electronics' ELSE 'Clothing' END FROM (SELECT 1 AS id UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 1000) AS data; 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
- 15 22:10:52 SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000 1 row(s) returned
- 16 22:12:51 CREATE INDEX price_idx ON Products(price) 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
- 17 22:14:39 SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000 1 row(s) returned
- 18 22:16:25 DROP INDEX price_idx ON Products 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
- 19 22:16:32 SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000 1 row(s) returned
- 20 22:18:56 SELECT * FROM Products WHERE price BETWEEN 100.00 AND 200.00 LIMIT 0, 1000 1 row(s) returned