Experiment No. 8

Aim :	To understand and apply the concept of index creation in SQL in order to improve the performance of queries on large databases.
Objectives :	i> Learn what an index is & why is it used. ii> Understand how indexing affects database performance. iii> Practice creating & removing indexes using SQL. iv> Compare query performance with & without indexes.

Experiment No. 8

Aim: To understand and apply the concept of index creation in SQL in order to improve the performance of queries on large databases.

Objectives:

i) Learn what an index is & why it's used.
ii) Understand how indexing affects DB's performance
iii) Practice creating & removing indexes using SQL
iv) Compare query and it's performance with & without indexes.

About indexes: An index in SQL is a data structure associated with a table that improves the speed of data retrieval operations. It acts like a reference or a pointer, allowing the database engine to quickly locate & access rows without scanning the entire table - much like how an index in a book helps to locate a topic faster.

Purpose of indexing:

i) Speeds up SELECT queries, especially those using WHERE, JOIN, ORDER BY and GROUP BY.
ii) It reduces the amount of data scanned during a query.

iii) Enhances the performance of large datasets or frequent searches.

Working of indexes : Indexes are typically implemented using data structure like B-trees or hash tables, depending on the type of index & the database system. When a query is executed, the database engine uses the index to narrow to the required rows efficiently instead of doing a full table scan.

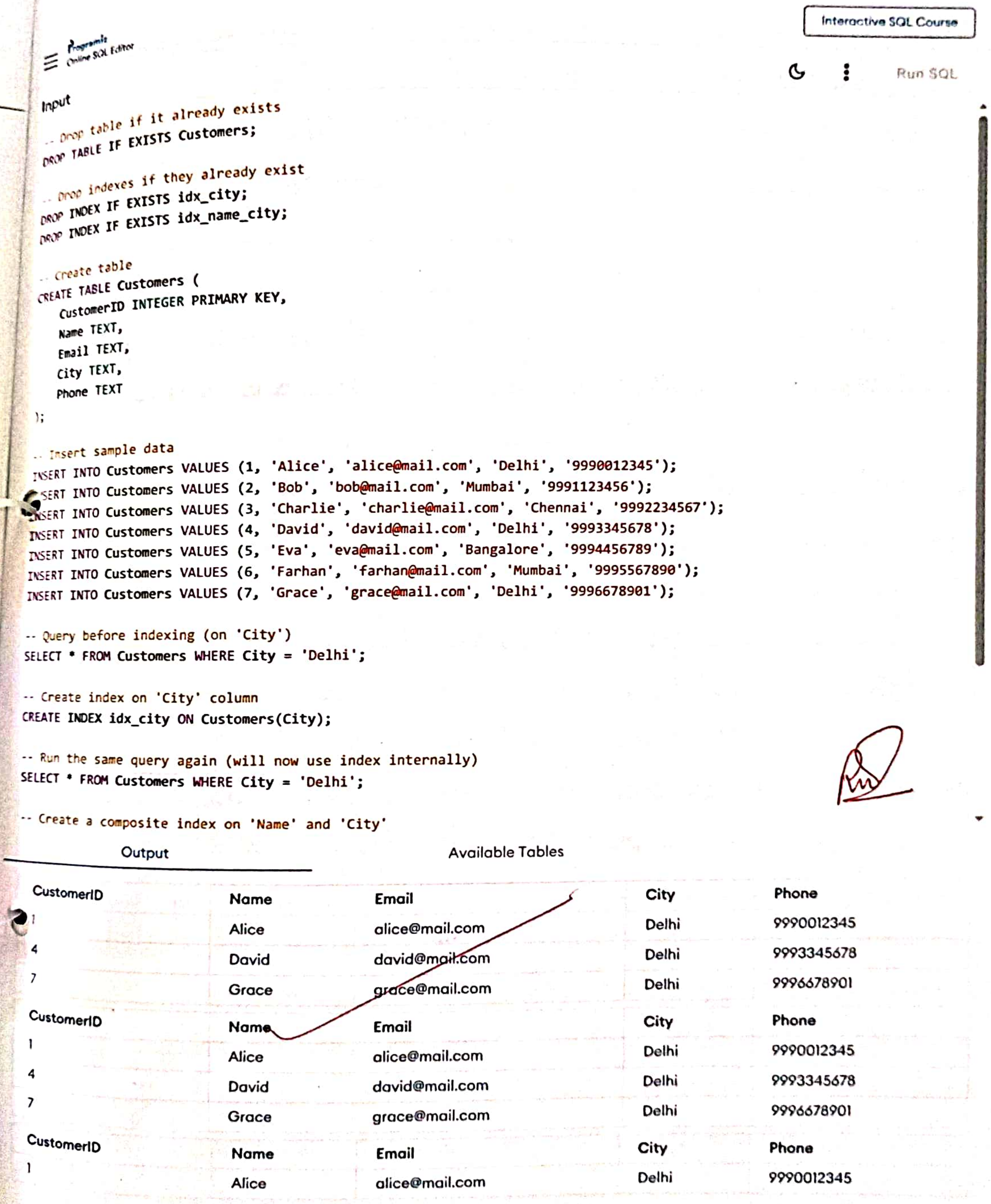
Types of ~~SQL~~ indexing in SQL :

i) Single-column index : Created on a single column.
Eg : CREATE INDEX idx_city ON customers (City);

ii) Composite index : Multi column index : Created on two or more columns.
Eg : CREATE INDEX idx_name_city ON customers (Name, City);

iii) Unique index : Ensures all values in the columns are unique. Automatically created with constraints like UNIQUE or PRIMARY KEY.

Conclusion: The concept of indexing was successfully implemented & understood. By applying indexes on specific columns query performance was enhanced. This experiment demonstrates the importance of indexing in DB optimization.



Input

```
-- Drop table if it already exists
DROP TABLE IF EXISTS Customers;

-- Drop indexes if they already exist
DROP INDEX IF EXISTS idx_city;
DROP INDEX IF EXISTS idx_name_city;

-- Create table
CREATE TABLE Customers (
    CustomerID INTEGER PRIMARY KEY,
    Name TEXT,
    Email TEXT,
    City TEXT,
    Phone TEXT
);

-- Insert sample data
INSERT INTO Customers VALUES (1, 'Alice', 'alice@mail.com', 'Delhi', '9990012345');
INSERT INTO Customers VALUES (2, 'Bob', 'bob@mail.com', 'Mumbai', '9991123456');
INSERT INTO Customers VALUES (3, 'Charlie', 'charlie@mail.com', 'Chennai', '9992234567');
INSERT INTO Customers VALUES (4, 'David', 'david@mail.com', 'Delhi', '9993345678');
INSERT INTO Customers VALUES (5, 'Eva', 'eva@mail.com', 'Bangalore', '9994456789');
INSERT INTO Customers VALUES (6, 'Farhan', 'farhan@mail.com', 'Mumbai', '9995567890');
INSERT INTO Customers VALUES (7, 'Grace', 'grace@mail.com', 'Delhi', '9996678901');

-- Query before indexing (on 'City')
SELECT * FROM Customers WHERE City = 'Delhi';

-- Create index on 'City' column
CREATE INDEX idx_city ON Customers(City);

-- Run the same query again (will now use index internally)
SELECT * FROM Customers WHERE City = 'Delhi';

-- Create a composite index on 'Name' and 'City'
```

[illegible]

Output   Available Tables

CustomerID	Name	Email	City	Phone
1	Alice	alice@mail.com	Delhi	9990012345
4	David	david@mail.com	Delhi	9993345678
7	Grace	grace@mail.com	Delhi	9996678901

CustomerID	Name	Email	City	Phone
1	Alice	alice@mail.com	Delhi	9990012345
4	David	david@mail.com	Delhi	9993345678
7	Grace	grace@mail.com	Delhi	9996678901

CustomerID	Name	Email	City	Phone
1	Alice	alice@mail.com	Delhi	9990012345

iv> Implicit Index : Automatically created when constr-aints like UNIQUE KEY or PRIMARY KEY are defined.

Advantages of indexing: i> Increases the speed of data retreival operations.
ii> Reduces the load of database for frequent queries.
iii> Helps in optimizing complex joins & filters.

Disadvantages of indexing: i> Takes up additional disk space.
ii> Slows down write operations (INSERT, UPDATE, DELETE) due to index maintenance.
iii> Too many indexes can degrade overall perform-ance instead of improving it.

Conclusion: The concept of indexing was successfully implemented & understood. By applying indexes on specific columns, query performance was enhanced. This experiment demonstrates the impor-tance of indexing in DB optimization.

A+ [illegible]
24/4/25