SQL 2.2

Learning Outcomes

At the end of this session, you will be able to:

- **Effortless Date Handling**: Retrieve current dates and perform date arithmetic with SQL functions like **DATE()** and **strftime()**.
- **Conditional Logic Mastery**: Understand **CASE** statements for creating new data based on defined conditions.
- **Database Control:** Execute essential operations like database and table **creation, alteration**, and **deletion**.
- **Data Manipulation and Querying Skills:** Acquire proficiency in **inserting, deleting**, and querying data in SQL databases.
- **Understanding Relationships:** Grasp **Primary** and **Foreign Keys**' significance in managing data relationships.
- **Exploration with Joins**: Comprehend **INNER JOIN** and **LEFT OUTER JOIN** for merging data from multiple tables.

Working with Date Data

The date() function in SQL retrieves the current date. It's used to display the present date in SQL queries.

Syntax: `SELECT date();`

Working with Date Data

```
SELECT DATE();
```

	DATE()
1	2023-12-06

```
SELECT date() AS current_date;
```

	current_date
1	2023-12-06

Working with Date Data

```
SELECT datetime() AS dateTimeNow;
```

dateTimeNow
2023-12-07 09:43:56

Working with Date Data

- Date manipulation in SQL involves commands to modify dates by adding or subtracting days, months, or years.
- It aids in temporal analysis and dataset adjustments for precise date-based computations in data science.

Subtracting Time Periods

To subtract a day, month, or year from a given date, you can use:

```
• SELECT DATE('2018-11-01','-1 day');
• SELECT DATE('2018-11-01','-1 month');
• SELECT DATE('2018-11-01','-1 year');
```

Adding Time Periods

To add a day, month, or year to a given date, use:

```
SELECT DATE('2018-11-01','+1 day');
SELECT DATE('2018-11-01','+1 month');
SELECT DATE('2018-11-01','+1 year');
```

Activity

In the table employees of employee.db database, display the hired date along with the month and date.

```
SELECT hiredate, strftime("%m", hiredate)
AS month, strftime("%d", hiredate) AS day
FROM employees;
```

Note: Before this, we need to update delimiter of dates.

```
UPDATE employees SET hiredate =
SUBSTR(hiredate, 7, 4) || '-' ||
SUBSTR(hiredate, 1, 2) || '-' ||
SUBSTR(hiredate, 4, 2);
```

hiredate	month	day
1990-12-17	12	17
1998-02-02	02	02
1996-01-02	01	02
1990-04-02	04	02
1994-06-23	06	23
1993-05-01	05	01
1997-09-22	09	22

CASE Statement

- The CASE statement in SQL allows conditional logic to create new columns based on specific criteria within existing data.
- For instance, when analyzing a sales dataset, you might want to categorize sales figures into 'High,' 'Medium,' or 'Low' based on predefined thresholds. The CASE statement facilitates this categorization.

Syntax: The basic syntax for the **CASE** statement is as follows:

```
SELECT *,
  CASE
    WHEN condition_1 THEN result_1
    WHEN condition_2 THEN result_2
    ELSE default_result
  END AS new_column_name
FROM table_name;
```

CASE Statement

```
SELECT *, CASE
WHEN sal > 60000 THEN "High"
WHEN sal > 40000 THEN "Medium"
ELSE 'Low'
END AS salary_cat
FROM employees;
```

id	name	job	mgr	hiredate	sal	comm	dept	salary_cat
8	GRANT	ENGINEER	10	03-30-1997	32000	NULL	2	Low
9	JACKSON	CEO	NULL	01-01-1990	75000	NULL	4	High
10	FILLMORE	MANAGER	9	08-09-1994	56000	NULL	2	Medium
11	ADAMS	ENGINEER	10	03-15-1996	34000	NULL	2	Low
12	WASHINGTON	ADMIN	6	04-16-1998	18000	NULL	4	Low
13	MONROE	ENGINEER	10	12-03-2000	30000	NULL	2	Low
14	ROOSEVELT	CPA	9	10-12-1995	35000	NULL	1	Low

The above code will create a new column called bins which will have values of high, medium, and low.

Creating a Database and Tables

Let’s create our own database.

Syntax: `CREATE DATABASE database_name;`

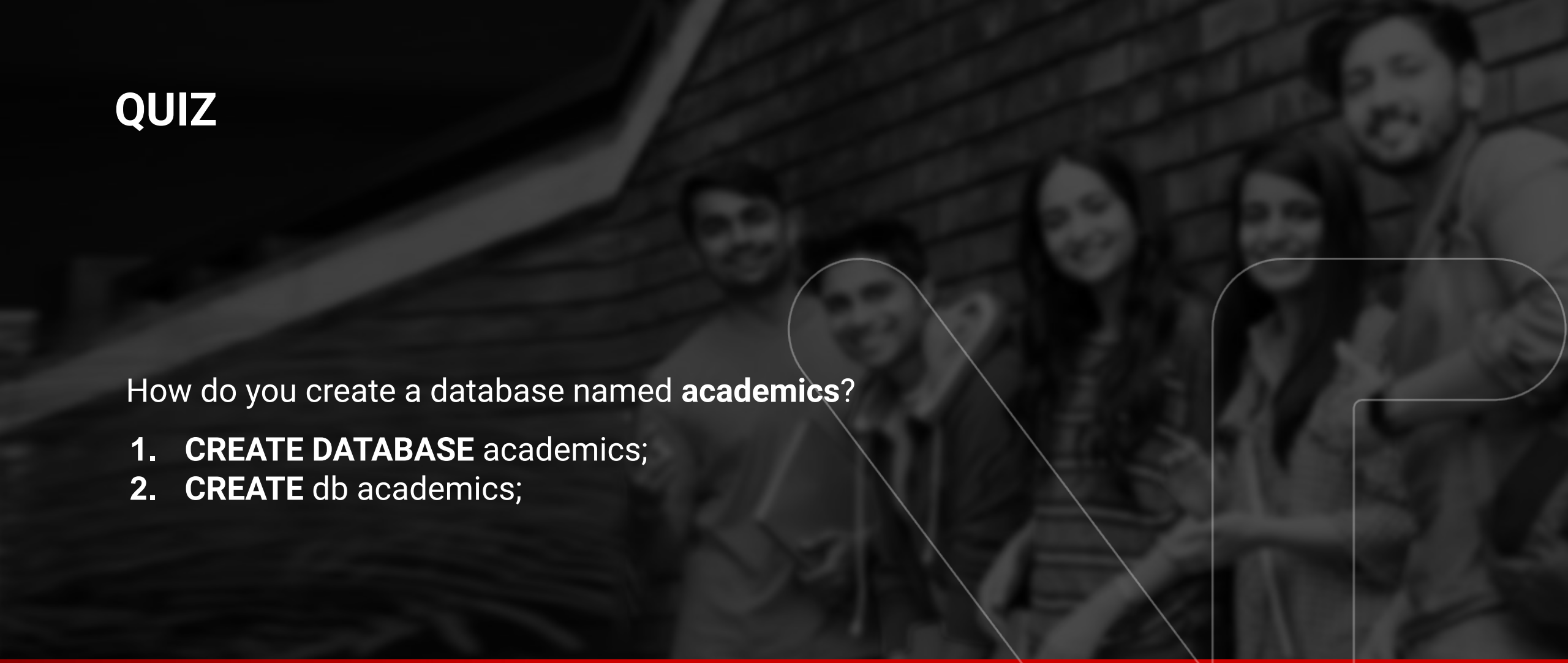
Example: An example is given below, in which a database named Analysis is created:

```
CREATE DATABASE Analysis;
```

Note: In tools like db-browser, a graphical interface allows database creation without the need for running SQL commands directly.

Hero

QUIZ



How do you create a database named **academics**?

1. **CREATE DATABASE** academics;
2. **CREATE** db academics;

Creating Tables

- Once we know how to create a database, the next step is to start creating tables in a database. In order to create tables, the first thing we need to do is to define the structure of the table.
- To do that, we will use the command called CREATE TABLE. Below is the general syntax of the command.

```
CREATE TABLE
table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype
);
```

Hero

Storage Classes and Data Types in SQLite

Each value stored in an SQLite database (or manipulated by the database engine) has one of the following storage classes:

- **NULL:** Represents a NULL value.
- **INTEGER:** Stores signed integers in varying byte sizes (0, 1, 2, 3, 4, 6, or 8) based on the value's magnitude.
- **REAL:** Represents floating-point values stored as 8-byte IEEE floating point numbers.
- **TEXT:** Stores text strings utilizing the database encoding (UTF-8, UTF-16BE, or UTF-16LE).
- **BLOB:** Represents data stored precisely as entered, functioning as a blob of data.

Activity

In your SQLite DB-Browser, create a Student table in academics with following attributes:

- **student_id:** This will contain the number data and the student ID of a student.
- **first_name:** This will contain text data and a student's first name.
- **last_name:** This will again contain text data and the last name of a student.

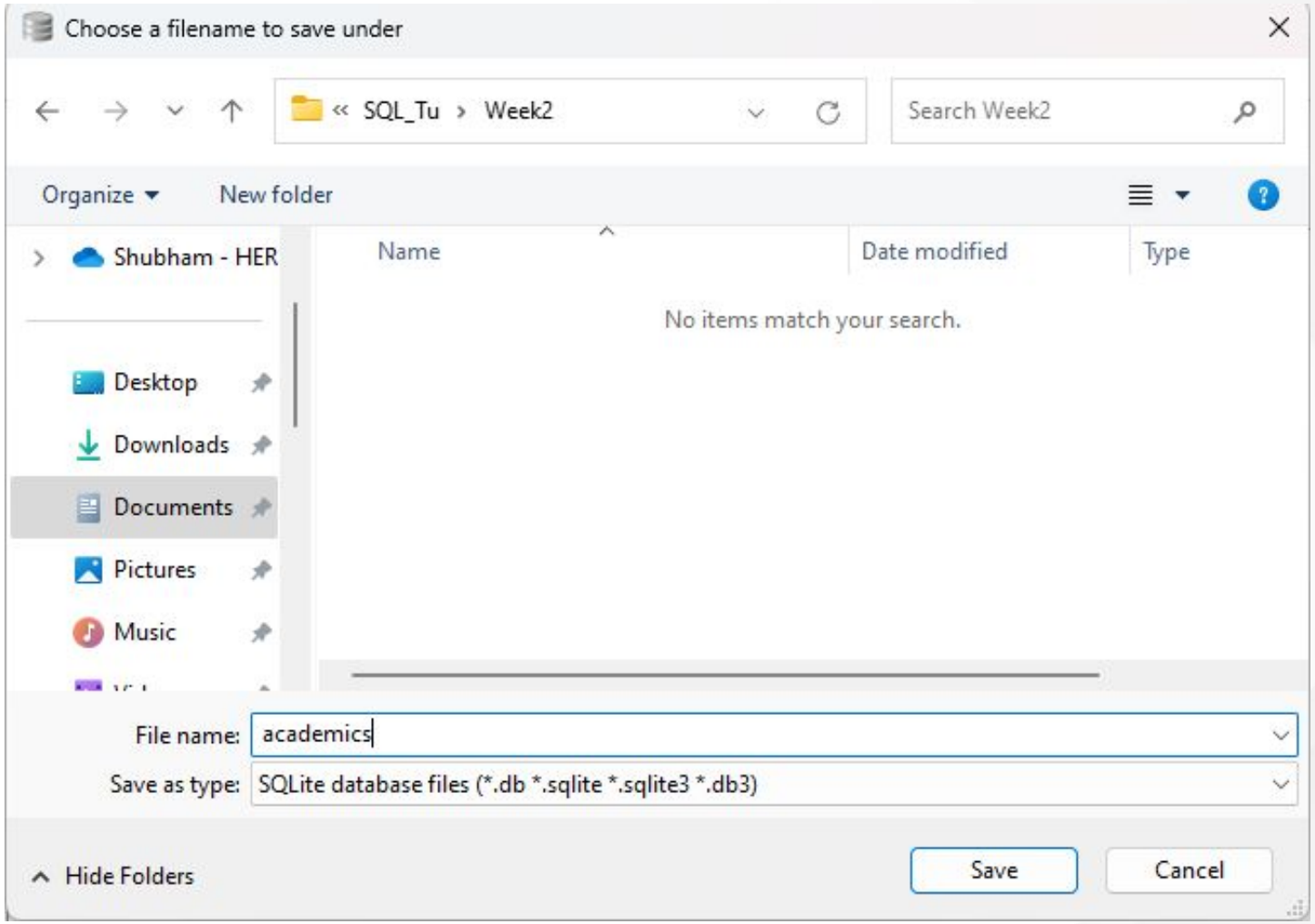
Activity

In your SQLite DB-Browser, create a Student table in academics with following attributes:

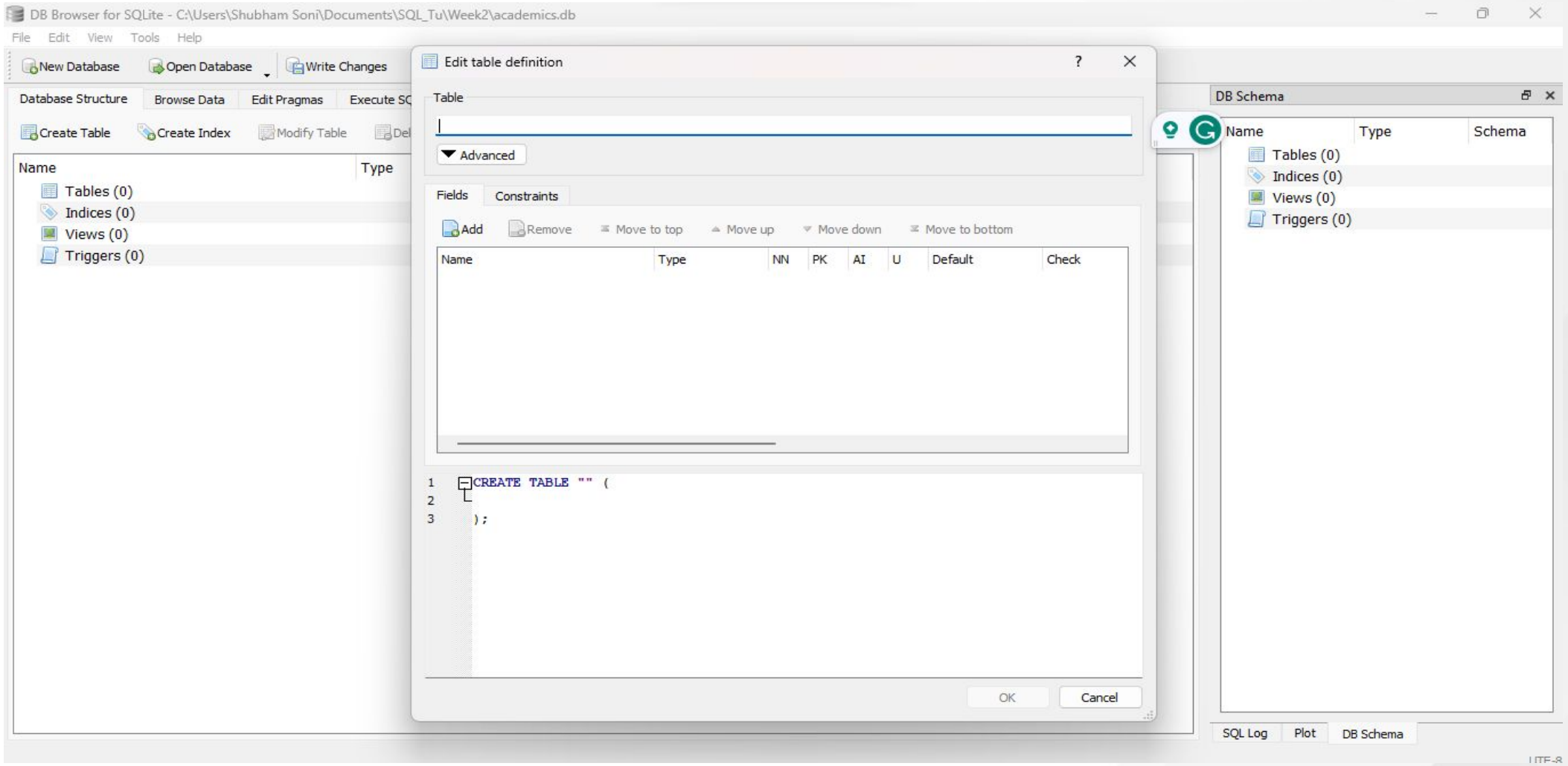
- **student_id:** This will contain the number data and the student ID of a student.
- **first_name:** This will contain text data and a student's first name.
- **last_name:** This will again contain text data and the last name of a student.

```
CREATE TABLE student (
    Student_Id INTEGER,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50)
);
```

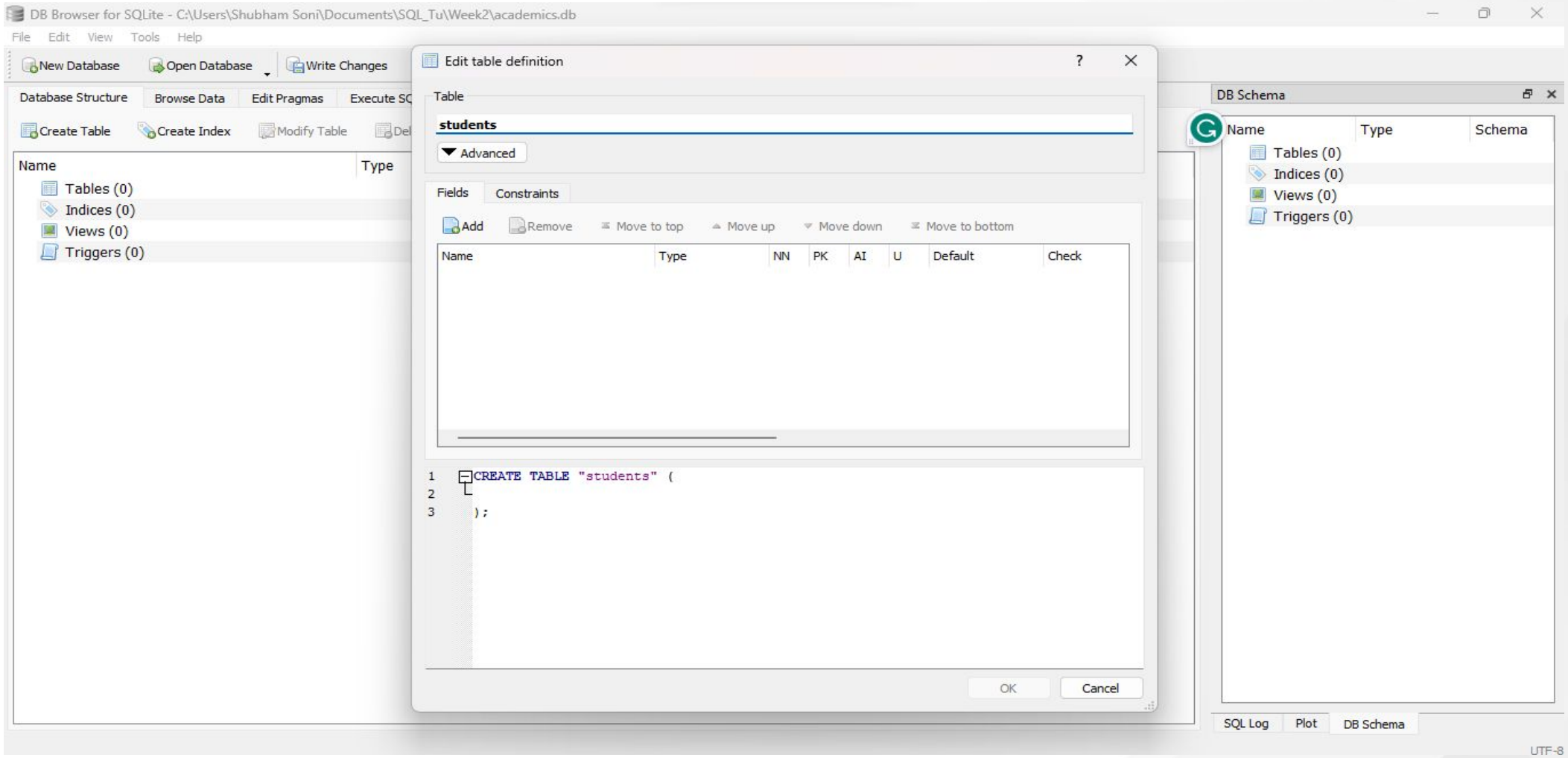
Activity


Choose a filename to save under
« SQL_Tu › Week2
Search Week2
Organize
New folder
Shubham - HER
Name
Date modified
Type
No items match your search.
Desktop
Downloads
Documents
Pictures
Music
File name:
academics
Save as type:
SQLite database files (*.db *.sqlite *.sqlite3 *.db3)
Hide Folders
Save
Cancel

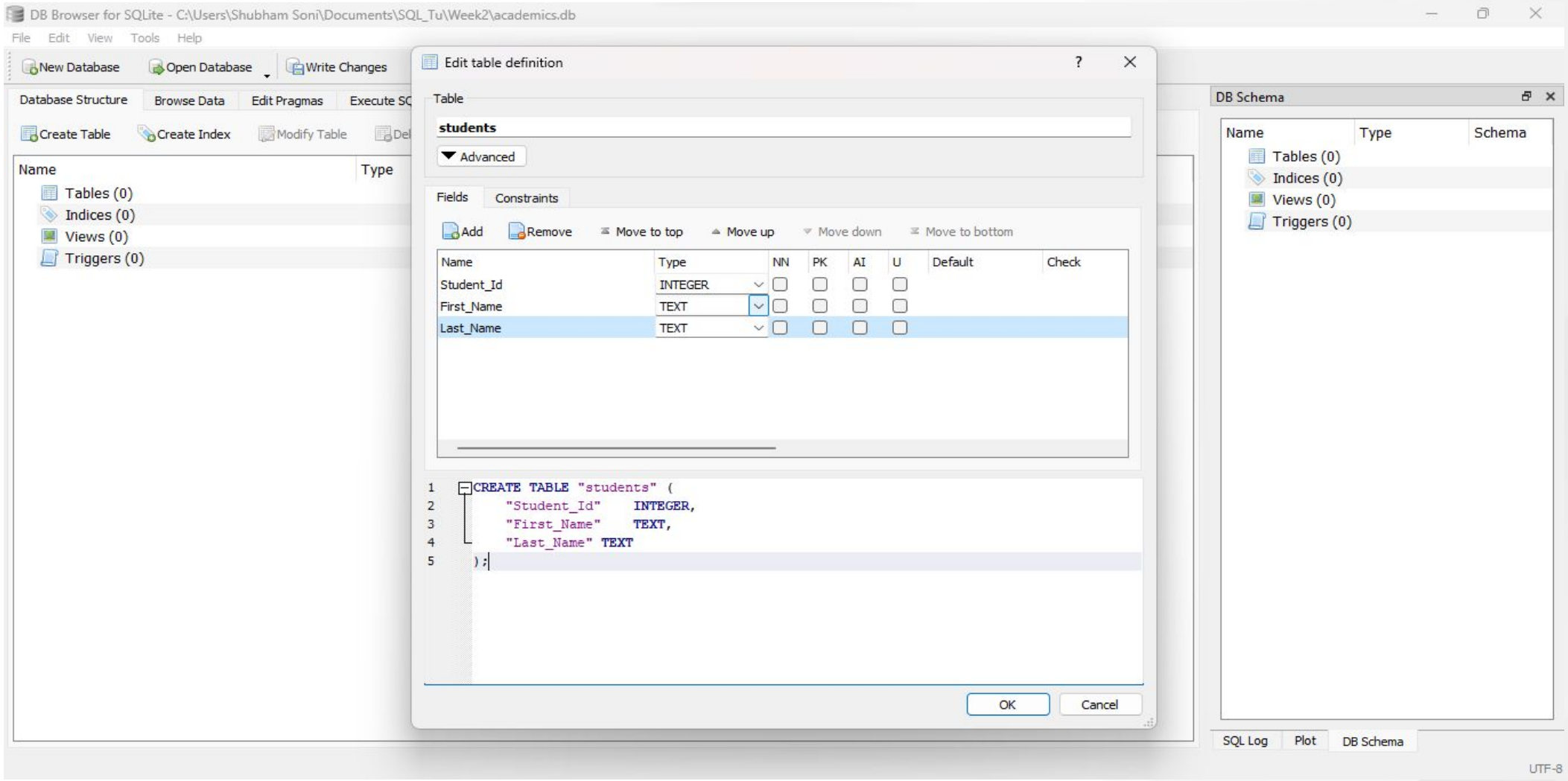
Activity


DB Browser for SQLite - C:\Users\Shubham Soni\Documents\SQL_Tu\Week2\academics.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Database Structure
Browse Data
Edit Pragmas
Create Table
Create Index
Modify Table
Name
Type
Tables (0)
Indices (0)
Views (0)
Triggers (0)
Edit table definition
Table
Advanced
Fields
Constraints
Add
Remove
Move to top
Move up
Move down
Move to bottom
Name
Type
NN
PK
AI
U
Default
Check
1 CREATE TABLE "" (
2
3);
OK
Cancel
DB Schema
Name
Type
Schema
Tables (0)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema

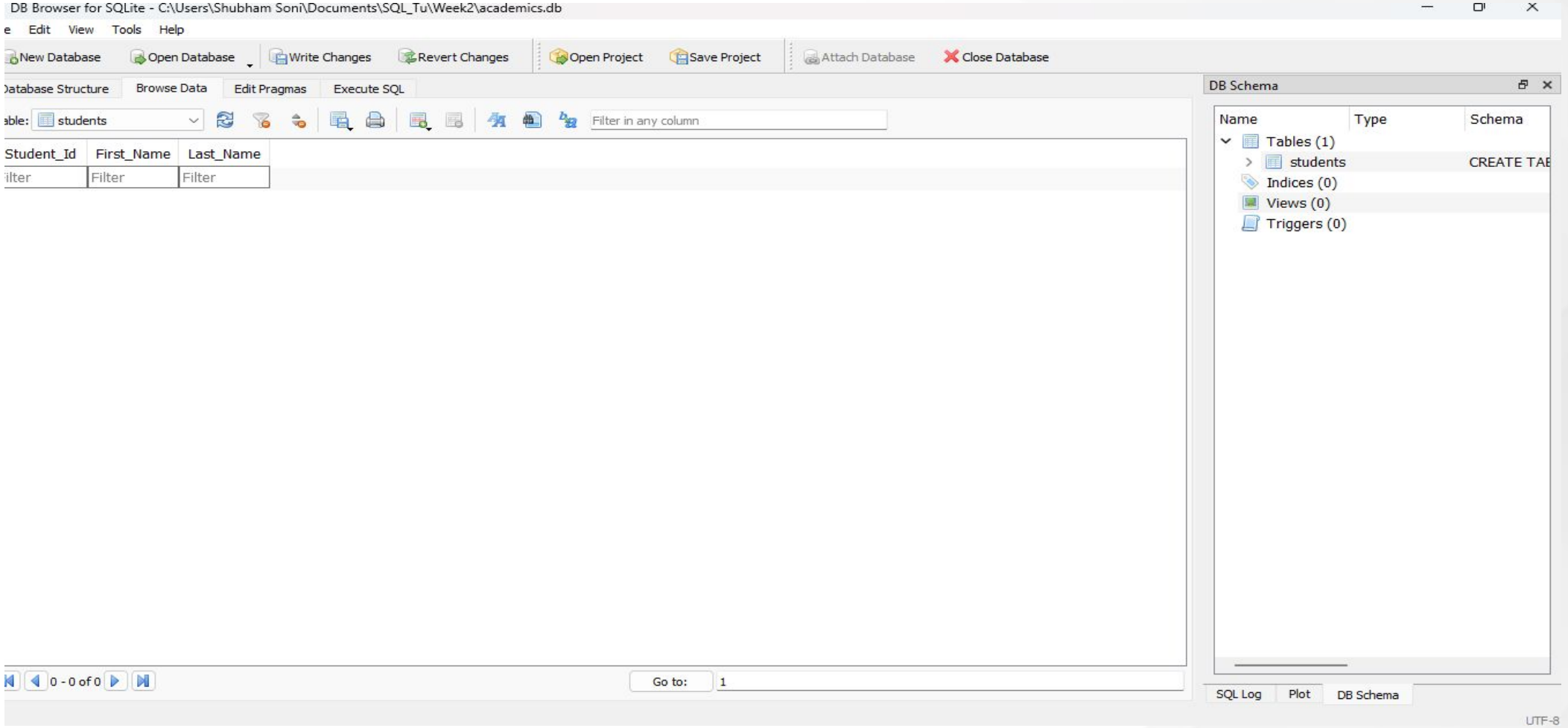
Activity


DB Browser for SQLite - C:\Users\Shubham Soni\Documents\SQL_Tu\Week2\academics.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Database Structure
Browse Data
Edit Pragmas
Create Table
Create Index
Modify Table
Name
Type
Tables (0)
Indices (0)
Views (0)
Triggers (0)
Edit table definition
Table
students
Advanced
Fields
Constraints
Add
Remove
Move to top
Move up
Move down
Move to bottom
Name
Type
NN
PK
AI
U
Default
Check
1 CREATE TABLE "students" (
2
3);
OK
Cancel
DB Schema
Name
Type
Schema
Tables (0)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

Activity


DB Browser for SQLite - C:\Users\Shubham Soni\Documents\SQL_Tu\Week2\academics.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Database Structure
Browse Data
Edit Pragmas
Create Table
Create Index
Modify Table
Name
Type
Tables (0)
Indices (0)
Views (0)
Triggers (0)
Edit table definition
Table
students
Advanced
Fields
Constraints
Add
Remove
Move to top
Move up
Move down
Move to bottom
Name
Type
NN
PK
AI
U
Default
Check
Student_Id
INTEGER
First_Name
TEXT
Last_Name
TEXT
CREATE TABLE "students" (
"Student_Id" INTEGER,
"First_Name" TEXT,
"Last_Name" TEXT
);
OK
Cancel
DB Schema
Name
Type
Schema
Tables (0)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

Activity


DB Browser for SQLite - C:\Users\Shubham Soni\Documents\SQL_Tu\Week2\academics.db
Edit
View
Tools
Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
students
Filter in any column
Student_Id
First_Name
Last_Name
Filter
Filter
Filter
0 - 0 of 0
Go to:
1
DB Schema
Name
Type
Schema
Tables (1)
students
CREATE TAB
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

Inserting Data into Tables

The INSERT INTO statement is pivotal for adding new records to a table, offering two approaches to insert data:

1. **Specifying Column Names and Values:**

```
INSERT INTO table_name (column1, column2,...) VALUES
(value1, value2,...);
```

2. **Direct Insert Without Specifying Column Names:**

```
INSERT INTO table_name VALUES (value1, value2,...);
```

Note: In the second method, you can add data directly into all columns without mentioning their names.

Hero

How to Insert Data

Inserting Data into Tables

1. **First Approach:**

```
INSERT INTO students (Student_Id, First_Name, Last_Name
) VALUES (1, 'Leo', 'Saut');
```

2. **Second Approach:**

```
INSERT INTO students VALUES (2, 'Sam', 'Stuart');
```

Inserting Data into Tables

Filling in multiple rows within a single query.

```
INSERT INTO student VALUES (3, 'Sam', 'Stuart'), (4, 'Tom', 'Little');
```

QUIZ

How do you add a row to the table ‘students’ with 2 text columns of First_name and Last_name?

1. INSERT INTO students Student_Id, First_Name) VALUES (‘Ram’,’Sharma’);
2. INSERT INTO students VALUES (‘Ram’,’Sharma’);
3. INSERT students VALUES (‘Ram’,’Sharma’);

Inserting Data into Tables

SQL 1

```
1   INSERT INTO students (Student_Id, First_Name, Last_Name )
2   VALUES (1, 'Leo', 'Saut');
3
4   INSERT INTO students
5   VALUES (2, 'Sam', 'Stuart');
6
7   INSERT INTO students (Student_Id, First_Name, Last_Name )
8   VALUES (3, 'Sam', 'Stuart'), (4, 'Tom', 'Little');
9
10  INSERT INTO students
11  VALUES (5, 'Anderw', 'Ng'), (6, 'Geoffery', 'Hinton');
```

Student_Id	First_Name	Last_Name
Filter	Filter	Filter
1	Leo	Saut
2	Sam	Stuart
3	Sam	Stuart
4	Tom	Little
5	Anderw	Ng
6	Geoffery	Hinton

Hero

SQL Command Groups

- Data Query Language (**DQL**) statement
- Data Definition Language (**DDL**) statements
- Data Manipulation Language (**DML**) statements

Data Query Language (DQL) statement

- The SELECT statement is used to access data contained in a table.
- The returned data is saved in a result table referred to as the result-set.

Syntax:

```
SELECT col1, col2, … FROM table_name;
```

Note: In the previous session, we exclusively covered DQL (Data Query Language) statements, focusing on data querying techniques.

Data Definition Language (DDL) statements

- These statements are responsible for defining and managing the structure of the database.
- DDL statements like CREATE, ALTER, DROP and DELETE FROM students;
- are used to create, modify, or delete database objects like tables.

CREATE Statement

- It generates new database objects.
- **Syntax:**

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
);
```

CREATE Statement

student_id	first_name	last_name	age
Filter	Filter	Filter	Filter

Example:

```
CREATE TABLE students (
    student_id INTEGER PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    age INTEGER
);
```

ALTER Statement

Modifies existing database objects.

Syntax:

```
ALTER TABLE table_name ADD
column_name datatype;
```

ALTER Statement

student_id	first_name	last_name	age	email
Filter	Filter	Filter	Filter	Filter

Example:

```
ALTER TABLE students ADD email TEXT;
```

DROP Statement

Syntax:

```
DROP TABLE table_name;
```

Example:

```
DROP TABLE students;
```

DELETE Statement

- **Removes all data from a table.**
- **Syntax**

```
DELETE TABLE
table_name;
```

Example:

```
DELETE FROM
students;
```

Note:

- This content provides an overview of **DDL**, its key statements (**CREATE, ALTER, DROP ,DELETE**), their syntax, and illustrative code examples.
- The **SQL** statement "**TRUNCATE TABLE students;**" is a valid command in most database management systems (DBMS). It is used to remove all rows from the "**students**" table, effectively deleting all the data while keeping the table structure intact.

Data Manipulation Language (DML) Statements

- **DML** statements are used for manipulating data within the database.
- **DML** commands (**INSERT, UPDATE, DELETE**) manage the data stored in tables, allowing for the i**nsertion, modification, or deletion** of records, thereby manipulating the database content.

Data Manipulation Language (DML) Statements

Key DML Statements:

- INSERT
- UPDATE
- DELETE

INSERT Statement

- **Adds new records into a table.**
- **Syntax:**

```
INSERT INTO table_name (col1,
col2, ...) VALUES (value1,
value2, ...);
```

Example:

```
INSERT INTO students
(first_name, last_name, age)
VALUES ('John', 'Doe', 25);
```

UPDATE Statement

- **Modifies existing records in a table.**
- **Syntax:**

```
UPDATE table_name SET column1
= value1, column2 = value2,
... WHERE condition;
```

Example:

```
UPDATE students SET age = 26
WHERE first_name = 'John';
```

student_id	first_name	last_name	age
Filter	Filter	Filter	Filter
1	John	Doe	26

Hero

DELETE Statement

- **Removes records from a table based on a specific condition.**
- **Syntax:**

```
DELETE FROM table_name WHERE
condition;
```

Example:

```
DELETE FROM students WHERE
age > 30;
```

Understanding Relationships in Data

One-to-One Relationship in Databases

- A "one-to-one relationship" occurs when the **primary key** of one table matches the **foreign key** of another table.
- This connection establishes a direct and singular correspondence between the two tables.

Understanding Relationships in Data

Example

Imagine you have a table that contains employee information and another table that contains the salary account details. Now, each employee will have only one salary account. In this case, there is a one-to-one relationship between employees and their bank account.

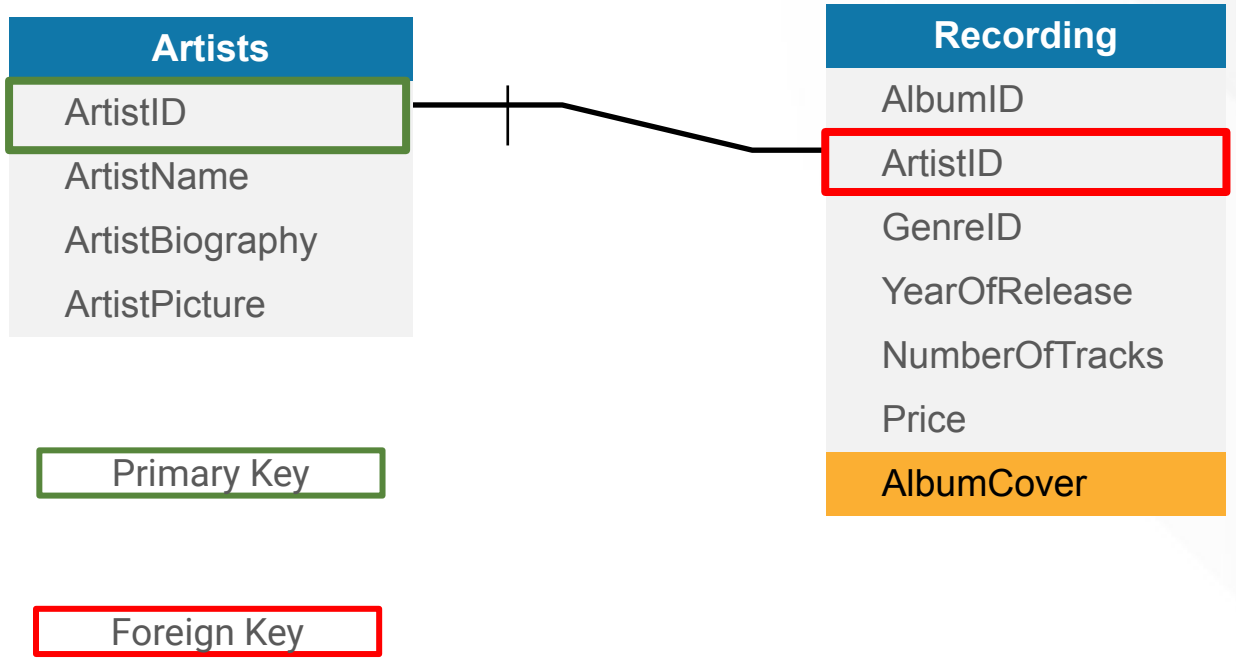
Employee
• Emp_id
• Name
• Designation
• Department

Salary_Account
• Id
• Emp_id
• Date
• Transaction

Describe Types of Relationships in Data

- **What is a Primary Key?**
  - Ensures the data in a specific column is unique.
- **What is a Foreign Key?**
  - Column or a group of columns in a relational database table that provides a link between data in two tables

Describe Types of Relationships in Data



What Are Constraints?

- Constraints are rules imposed on a table's data columns to ensure stored information remains meaningful and pertinent.
- Constraints:
  - Are the rules enforced on a data column in a table to store meaningful and relevant data?
  - Can be applied at the column or table level.
  - At the column level are applied to a single column, whereas constraints at the table level are applied to the entire table.

Types of Constraints

Following are commonly used constraints available in SQLite:

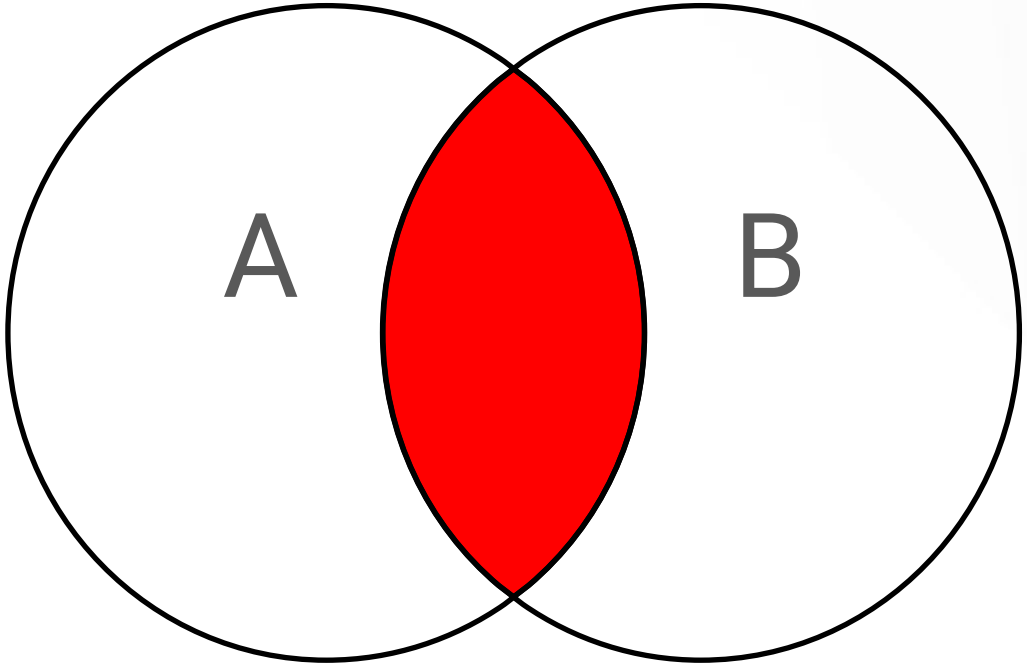
- **NOT NULL Constraint** − Ensures that a column cannot have NULL value.
- **DEFAULT Constraint** − Provides a default value for a column when none is specified.
- **UNIQUE Constraint** − Ensures that all values in a column are different.
- **PRIMARY Key** − Uniquely identifies each row/record in a database table.
- **CHECK Constraint** − Ensures that all values in a column satisfies certain conditions.
- **FOREIGN Constraint -** SQL foreign key constraints are used to enforce "exists" relationships between tables.

What Are JOINS?

- Joins combine rows from multiple tables by linking them through a shared column.
- Types of Joins
  - **Inner Join**
  - **Left Outer Join**

What Is an INNER JOIN ?

An INNER JOIN selects all rows from both tables when there is a match between the columns.



INNER JOIN

Syntax:

```
SELECT column_name(s) FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

- This SQL syntax combines rows from 'table1' and 'table2' based on the matching condition specified after the INNER JOIN keyword using the ON clause.
- Adjust 'column_name' to the specific columns you want to use for matching in both tables

Hero

Example

Let's say we have two table:

Temperature

city	Temperature
Delhi	38
Mumbai	32
Chennai	35

Humidity

city	Humidity
Mumbai	22
Delhi	48
Bangalore	55

Example

```
SELECT * FROM Temperature T
INNER JOIN Humidity H
ON T.city = H.city;
```

OR

```
SELECT * FROM Temperature T
 JOIN Humidity H
ON T.city = H.city;
```

city	Temperature	city	Humidity
Delhi	38	Delhi	48
Mumbai	32	Mumbai	22

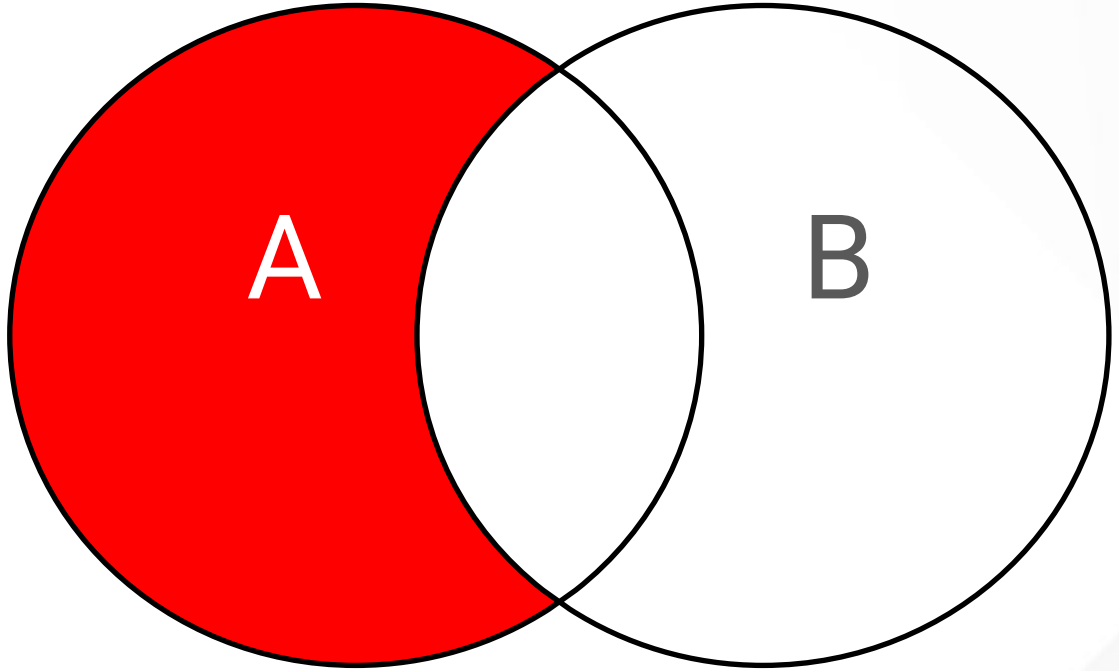
Example

```
SELECT T.city, T.temperature,
H.humidity FROM Temperature T
INNER JOIN Humidity H
ON T.city = H.city;
```

city	Temperature	Humidity
Delhi	38	48
Mumbai	32	22

What Is a LEFT OUTER JOIN?

A **LEFT OUTER JOIN** returns all records from the left table and the matching records from the right table.



LEFT OUTER JOIN

Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON
table1.column_name =
table2.column_name;
```

Example

```
SELECT t.city, t.Temperature,
h.Humidity FROM Temperature T
LEFT JOIN Humidity H
ON T.city = H.city;
```

city	Temperature	Humidity
Delhi	38	48
Mumbai	32	22
Chennai	35	*NULL*

Hero

Summary

- Expanding our SQL journey, we honed essential skills for adept data management and analysis. We began with mastering date functions for efficient date handling and applied conditional logic through CASE statements for dynamic data creation based on specific conditions.
- Further, we delved into comprehensive database control, covering operations like data manipulation, querying, and understanding data relationships, including Primary and Foreign Keys.
- Additionally, we explored join operations, dissecting INNER JOIN and LEFT OUTER JOIN for seamless data integration across tables.

Thank You!