FAZAIA BILQUIS COLLEGE
NUR KHAN BASE,RWP

Lab #04 SQL Tasks

SUBMITTED BY: *MARIA ATTA*

ROLL NO: *BSCS-13-F24-01*

SUBJECT: *DATABASE*

LAB 4: SQL (LAB TASK)

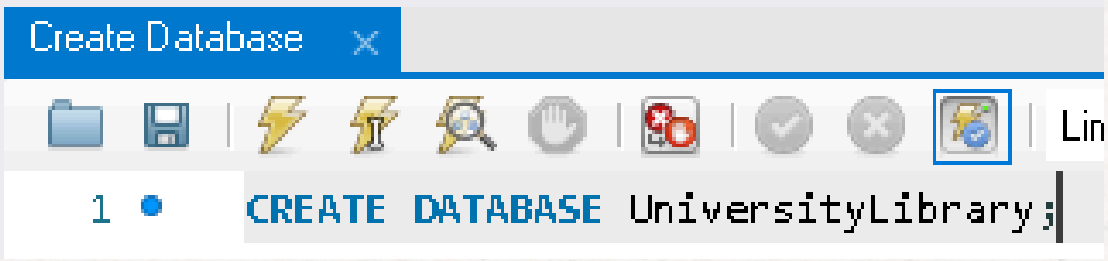
- ## WEEK 4: INTRODUCTION TO SQL

  Lab Objectives:

  The system needs to:

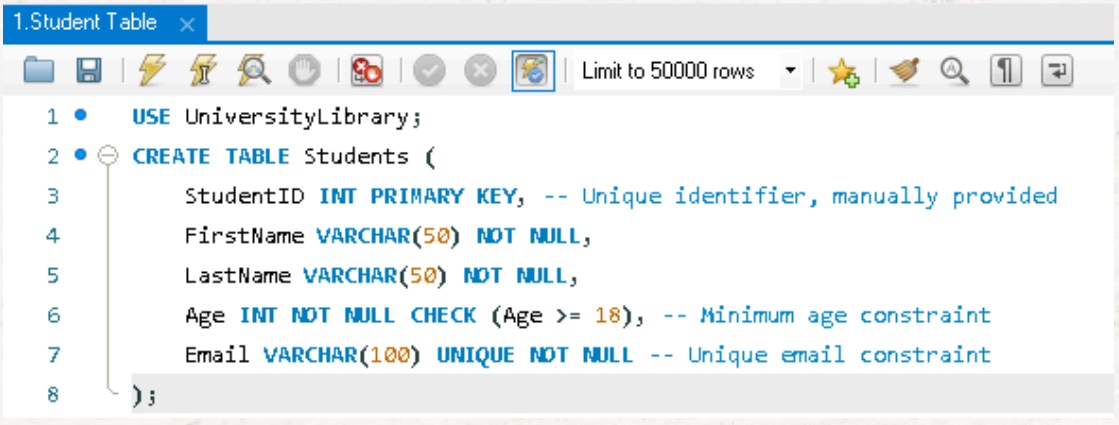
  - Store student details (minimum age 18).
  - Store books (unique titles, publication year after 1900).
  - Record borrowing details with relationships enforced via primary and foreign keys.

- ## CREATE A DATABASE:



- ## OUTPUT:

- ## TABLE OF STUDENTS:



- **OUTPUT:**

Output

Action Output

#	Time	Action	Message
1	15:52:58	USE UniversityLibrary	0 row(s) affected
2	15:52:58	CREATE TABLE Students (StudentID INT PRIMARY KEY, -- Unique identifier, manually provid...	0 row(s) affected

- **BOOKS TABLE:**

2.Books Table

Limit to 50000 rows

```
USE UniversityLibrary;
CREATE TABLE Books (
    BookID INT PRIMARY KEY, -- Unique identifier, manually provided
    Title VARCHAR(100) UNIQUE NOT NULL, -- Unique title constraint
    Genre VARCHAR(50) NOT NULL,
    PublishedYear INT NOT NULL CHECK (PublishedYear >= 1900) -- Year constraint
);
```

- **OUTPUT:**

Output

Action Output

#	Time	Action	Message
1	15:54:00	USE UniversityLibrary	0 row(s) affected
2	15:54:00	CREATE TABLE Books (BookID INT PRIMARY KEY, -- Unique identifier, manually provided ...	0 row(s) affected

- **TABLE OF BORROWRECORDS:**

3. BorrowRecords Table

Limit to 50000 rows

```
USE UniversityLibrary;
CREATE TABLE BorrowRecords (
    RecordID INT PRIMARY KEY,
    StudentID INT NOT NULL,
    BookID INT NOT NULL,
    BorrowDate DATE NOT NULL,
    ReturnDate DATE NULL, -- Nullable as books may not yet be returned
    CONSTRAINT FK_Student FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE CASCADE,
    CONSTRAINT FK_Book FOREIGN KEY (BookID) REFERENCES Books(BookID) ON DELETE CASCADE
);
```

- **OUTPUT:**

Output

Action Output

#	Time	Action	Message
1	15:55:00	USE Univer... Select output pane	0 row(s) affected
2	15:55:00	CREATE TABLE BorrowRecords (RecordID INT PRIMARY KEY, -- Unique identifier for each ...	0 row(s) affected

- **INSERT DATA IN STUDENT:**

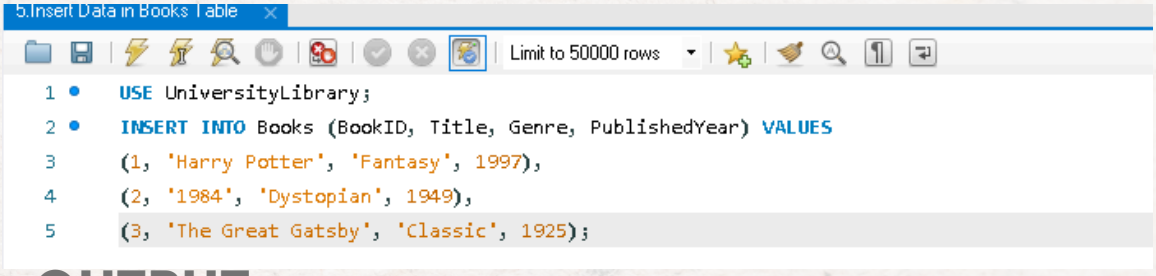
4.Insert Data in Student table

Limit to 50000 rows

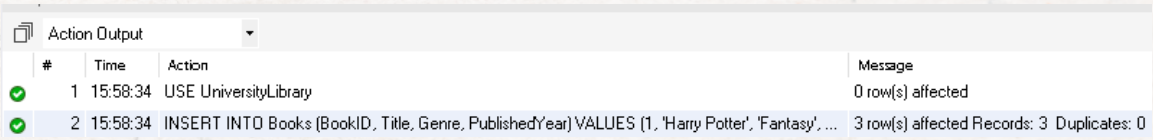
```
USE UniversityLibrary;
INSERT INTO Students (StudentID, FirstName, LastName, Age, Email) VALUES
(1, 'Ali', 'Khan', 20, 'ali.khan@example.com'),
(2, 'Sara', 'Ahmed', 22, 'sara.ahmed@example.com'),
(3, 'Usman', 'Raza', 19, 'usman.raza@example.com');
```

- **OUTPUT:**

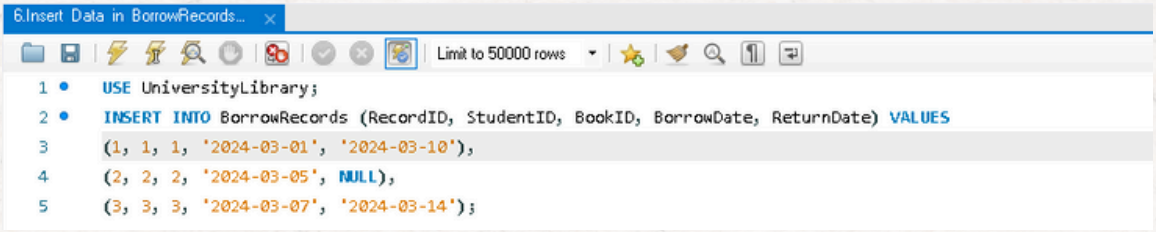
- **INSERT DATA IN BOOKS TABLE:**



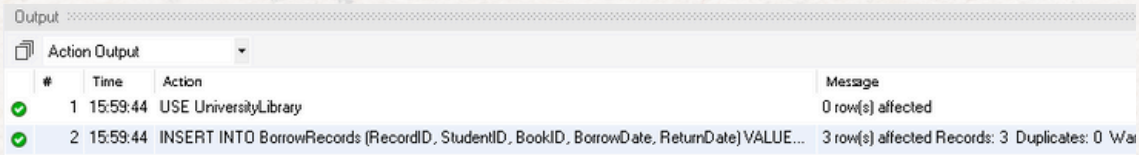
- **OUTPUT:**



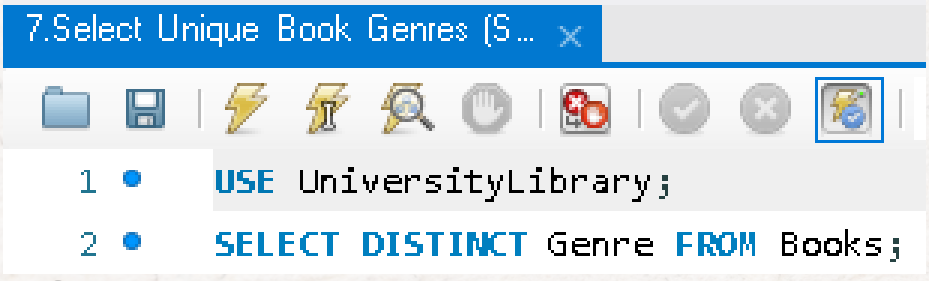
- **INSERT DATA IN BORROWRECORDS:**



- **OUTPUT:**



- **SELECT UNIQUE BOOK GENRES:**



- **OUTPUT:**

Genre
Fantasy
Dystopian
Classic

- **RETRIEVE ALL BOOKS:**

8.Retrieve All Books

```
USE UniversityLibrary;
SELECT * FROM Books;
```

- **OUTPUT:**

BookID	Title	Genre	PublishedYear
1	Harry Potter	Fantasy	1997
2	1984	Dystopian	1949
3	The Great Gatsby	Classic	1925
NULL	NULL	NULL	NULL

- **RETRIEVE ALL STUDENTS:**

9.Retrieve All Students

```
USE UniversityLibrary;
SELECT * FROM Students;
```

- **OUTPUT:**

StudentID	FirstName	LastName	Age	Email
1	Ali	Khan	20	ali.khan@example.com
2	Sara	Ahmed	22	sara.ahmed@example.com
3	Usman	Raza	19	usman.raza@example.com
NULL	NULL	NULL	NULL	NULL

- **RETRIEVE BOOKS PUBLISHED AFTER 1950:**

10.Retrieve Books Published Aft...

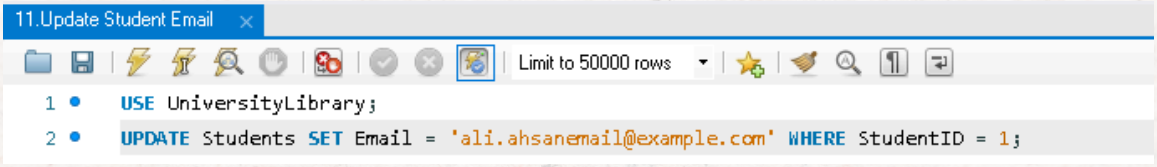
```
USE UniversityLibrary;
SELECT Title, PublishedYear FROM Books WHERE PublishedYear > 1950;
```

- **OUTPUT:**

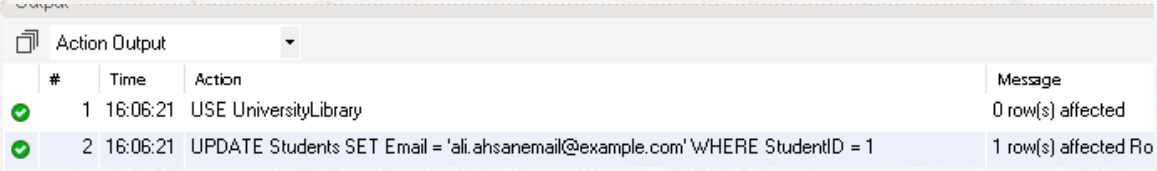
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Title	PublishedYear
Harry Potter	1997

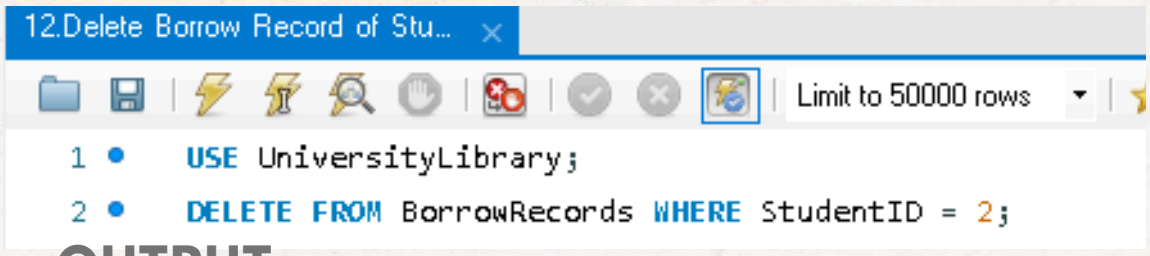
- **UPDATE STUDENT EMAIL:**



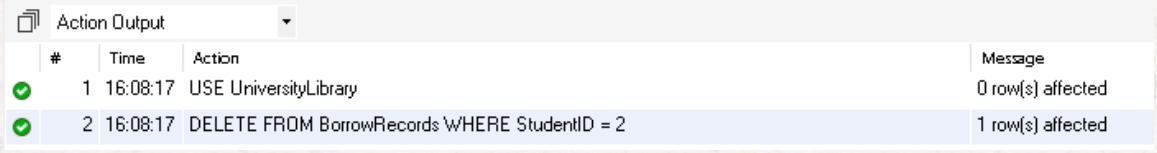
- **OUTPUT:**



- **DELETE BORROW RECORD OF STUDENT 2:**



- **OUTPUT:**



LAB 4: SQL (HOME TASK)

SCENARIO:

A small movie rental shop wants to store and manage its data digitally using a database system. The shop rents movies to customers and tracks rental records. The system should:

✓ Store customer details, ensuring each customer has a unique email.

✓ Maintain a catalog of available movies.

✓ Track movie rentals, recording who rented which movie and when.

✓ Enforce relationships between tables using primary and foreign keys.

- ***Create Database***

Create Database

Limit to 50000 rows

```
1  CREATE DATABASE MovieRentalSystem;
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:29:04	CREATE DATABASE MovieRentalSystem	1 row(s) affected

- ***Create Customer table :***

1.Customer Table

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  CREATE TABLE Customers (
3      CustomerID INT PRIMARY KEY,
4      FirstName VARCHAR(50) NOT NULL,
5      LastName VARCHAR(50) NOT NULL,
6      Email VARCHAR(100) UNIQUE NOT NULL
7  );
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:29:04	CREATE DATABASE MovieRentalSystem	1 row(s) affected
2	14:31:06	USE MovieRentalSystem	0 row(s) affected
3	14:31:06	CREATE TABLE Customers (CustomerID INT PRIMARY KEY, FirstName VARCHAR(50) N...	0 row(s) affected

- ***Create Movies table :***

2.Movie Table

```sql
USE MovieRentalSystem;
CREATE TABLE Movies (
    MovieID INT PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Genre VARCHAR(50) NOT NULL,
    ReleaseYear INT CHECK (ReleaseYear > 1900),
    AvailableCopies INT CHECK (AvailableCopies >= 0)
);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:32:14	USE MovieRentalSystem	0 row(s) affected
2	14:32:14	CREATE TABLE Movies (MovieID INT PRIMARY KEY, Title VARCHAR(100) NOT NULL, ...	0 row(s) affected

- ***Create Rentals table :***

3.Rentals Table

```sql
USE MovieRentalSystem;
CREATE TABLE Rentals (
    RentalID INT PRIMARY KEY,
    CustomerID INT,
    MovieID INT,
    RentalDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),
    CHECK (ReturnDate IS NULL OR ReturnDate > RentalDate)
);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:32:14	USE MovieRentalSystem	0 row(s) affected
2	14:32:14	CREATE TABLE Movies (MovieID INT PRIMARY KEY, Title VARCHAR(100) NOT NULL, ...	0 row(s) affected
3	14:33:35	USE MovieRentalSystem	0 row(s) affected
4	14:33:35	CREATE TABLE Rentals (RentalID INT PRIMARY KEY, CustomerID INT, MovieID INT, ...	0 row(s) affected

- ***Insert Data in Customer Table:***

4.Insert Data in Customer table

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  INSERT INTO Customers (CustomerID,FirstName,LastName,Email) VALUES
3  (1, 'Maria','Atta','mariaatta33@gmail.com'),
4  (2,'Maham','Usman','maham55usman@gmail.com'),
5  (3,'Noor','Ain','noorulain12@gmail.com');
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:41:15	USE MovieRentalSystem	0 row(s) affected
2	14:41:15	INSERT INTO Customers (CustomerID,FirstName,LastName,Email) VALUES (1, 'Maria','Atta','mari...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

- ***Insert Data in Movie Table:***

5.Insert Data in Movie Table

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  INSERT INTO Movies (MovieID,Title,Genre,ReleaseYear,AvailableCopies) VALUES
3  (1, 'Harry Potter','Scientific',2000,210),
4  (2,'Simson','Animated',1998,150),
5  (3,'Avatar','Action',2009,100);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:46:11	USE MovieRentalSystem	0 row(s) affected
2	14:46:11	INSERT INTO Movies (MovieID,Title,Genre,ReleaseYear,AvailableCopies) VALUES (1, 'Harry Pot...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

- ***Insert Data in Rentals Table:***

6.Insert data in Rentals Table

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  INSERT INTO Rentals (RentalID, CustomerID, MovieID, RentalDate, ReturnDate) VALUES
3  (1, 1, 1, '2025-03-01', '2025-03-10'),
4  (2, 2, 2, '2025-03-15', NULL),
5  (3, 3, 3, '2025-03-20', NULL);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	14:47:53	USE MovieRentalSystem	0 row(s) affected
2	14:47:53	INSERT INTO Rentals (RentalID, CustomerID, MovieID, RentalDate, ReturnDate) VALUES (1, 1,...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

- ***Retrieve all Data from Movie Table:***

7.Retrieve all available movies fr...

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  SELECT * FROM Movies WHERE (AvailableCopies > 0);
```

- ***Output:***

Result Grid | Filter Rows: | Edit:

MovieID	Title	Genre	ReleaseYear	AvailableCopies
1	Harry Potter	Scientific	2000	210
2	Simson	Animated	1998	150
3	Avatar	Action	2009	100
NULL	NULL	NULL	NULL	NULL

- ***Retrieve unique movie genres:***

8.Find all unique movie genres

```
1  USE MovieRentalSystem;
2  SELECT DISTINCT Genre FROM Movies;
```

- ***Output:***

Result Grid | Filter Rows: | Export: | Wrap Cell

Genre
Scientific
Animated
Action

- ***Update Data decrease its AvailableCopies:***

10.increase its AvailableCopies...

Limit to 50000 rows

```
1  USE MovieRentalSystem;
2  UPDATE Movies SET AvailableCopies = AvailableCopies + 1 WHERE (MovieID = 2);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	15:01:12	USE MovieRentalSystem	0 row(s) affected
2	15:01:12	UPDATE Movies SET AvailableCopies = AvailableCopies + 1 WHERE (MovieID = 2)	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

- ***Update Data increase its AvailableCopies:***

11.Delete the rental record wher...

Limit to 50000 r

```
1  USE MovieRentalSystem;
2  DELETE FROM Rentals WHERE (CustomerID = 2);
```

- ***Output:***

Output

Action Output

#	Time	Action	Message
1	15:03:52	USE MovieRentalSystem	0 row(s) affected
2	15:03:52	DELETE FROM Rentals WHERE (CustomerID = 2)	1 row(s) affected