CS 4347.005 - Database Systems

Final Report Deliverable 2

Student Record Keeping System

Mia Sorola Yoshida, Abbas Khawaja, Vaishnavi Josyula, Anveetha Suresh, Justin Lu, Ahimsa Yukta, Abhishek Madhavan, Shubham Patel, Paribesh Upreti

0. Description:

- *Title*: Student Record Keeping System
- *Github Link*: https://github.com/justinlu200014/CS-4347-Group-6-project
- *Group Members:*
    - Paribesh Upreti - pxu200001@utdallas.edu
    - Abhishek Madhavan - axm200159@utdallas.edu
    - Anveetha Suresh - axs220399@utdallas.edu
    - Mia Sorola Yoshida - mas220015@utdallas.edu
    - Vaishnavi Josyula - vxj210016@utdallas.edu
    - Abbas Khawaja - aik210000@utdallas.edu
    - Justin Lu - jjl200014@utdallas.edu
    - Shubham Patel - shp200002@utdallas.edu
    - Ahimsa Yukta - axy210017@utdallas.edu
- *Delegation of tasks for the overall project*:
    - Paribesh: Frontend
        - Worked on the Delete page
    - Justin: Frontend
        - Worked on the Insert page
    - Shubham: Frontend
        - Worked on the Update page
    - Vaishnavi: Frontend
        - Worked on the Home, Quit, and Query page; Firebase authentication; Integration
    - Abhishek: Backend
    - Anveetha: Backend
    - Mia: Backend
    - Abbas: Backend
    - Ahimsa: Backend
- *Delegation of tasks for this report:*
    - Paribesh: Section 4
    - Justin: Section 0 and 4
    - Shubham: Section 0 and 4
    - Vaishnavi: Section 5, 4, 1, and 2
    - Abhishek: Section 3.4
    - Anveetha: Section 3.2
    - Mia: Section 3.1
    - Abbas: Section 3.3
    - Ahimsa: Section 3.5
- *Motivation:*
    - Designing a system to help college students manage various aspects of their lives can significantly improve their efficiency and reduce stress.
    - Our design will be used at U.S. universities to simplify administrative processes, benefiting students and institutions by providing student record management solutions.

    - This project has a personal connection to us, as we are college students who deal with the same information on a regular basis. Class schedules, club meetings, tuition, housing, and meal plans are all topics that are familiar to us.
- *Project timeline:*
    - Week 1 (11/2 - 11/8) - Normalize the EER diagram (3.1), Create Relational Database Model using Normalized EER diagram (3.2), Create the Database using the models (3.3), Query Execution (3.4), Create View (3.5)
    - Week 2 (11/9 - 11/17) - Finish the backend (3), Finish the frontend (4), Connect the frontend and backend, work on presentation slides

1. Introduction:

The project we are developing focuses on creating a comprehensive system to help college students manage multiple aspects of their campus life, including but not limited to academic schedules, finances, dining, housing, and extracurricular activities. As college students, we personally understand the challenges of organizing these essential components of student life. With multiple systems to navigate, from academic portals to dining and housing platforms, the process can become overwhelming.

Our goal is to develop a solution that mitigates this complexity by centralizing everything into a single, user-friendly platform. While many existing systems focus primarily on academic performance and grades, our platform goes beyond that. It aims to encompass the full spectrum of student life by including features that address extracurricular involvement, campus events, club participation, meal plans, housing information, and financial responsibilities.

Additionally, by implementing this program, we expect to contribute to the field of security by ensuring that sensitive information such as academic records, financial data, and housing agreements are protected through secure access control mechanisms, encryption, and privacy-focused design. Our contribution also includes enhancing data security in systems that manage multiple types of personal data, ensuring confidentiality, integrity, and availability.

Overall, we aim to provide students with a holistic tool that not only helps them manage their coursework but also promotes a balanced college experience. Through this platform, we aim to allow students to better organize their academic and non-academic lives, helping them stay on top of deadlines, manage extracurricular commitments, track dining and housing plans, and monitor financial obligations—all from one cohesive system.

2. Background and Related Work:

Background:

The Student Record Keeping System that we are developing is unique as it seeks to integrate various aspects of campus life—academic schedules, finances, dining, and housing—into a single cohesive

platform. While universities and colleges offer systems that manage each of these components, they are typically separated into different portals. For instance, academic scheduling is handled independently through platforms like university portals, while financial accounts are often managed by bursars. Dining services and housing have their own systems, adding another layer of separation. These separate systems force students to juggle multiple logins and interfaces, dividing and complicating the overall management of their campus life.

For example, at The University of Texas at Dallas (UTD), students manage their academic responsibilities through the Galaxy portal, which handles class registration, schedules, and grades, while financial responsibilities such as tuition payments and fee management are processed through bursar account, housing services are handled via another distinct account, and dining plans are overseen by UTD Services, creating multiple systems that students need to navigate. This segmentation results in inefficient and unstreamlined access to critical student services.

Our platform aims to integrate these functionalities, combining academic, financial, housing, and dining services into a unified system. While the individual components already exist in various forms across different institutions, no system has yet to fully integrate them into a single platform for students. The strength of our implementation lies in its ability to consolidate these aspects, making it easier for students to manage all facets of their college experience from one centralized system with a singular database rather than different accounts and websites that each deal with different databases, a concept that has yet to be fully realized despite the availability of these individual services.

Related Work:

Our project builds on several important studies and systems in the areas of student record management. Below is a chronological list of these works, with a brief explanation of their contributions and how our work extends beyond them.

1. Eludire, A. (2011) [1] – The Design and Implementation of Student Academic Record Management System: This paper focuses on tracking student performance through grades and class schedules, with an emphasis on data accuracy and usability. It provides a system to manage academic records efficiently but does not extend to other aspects of student life. Our project builds on this idea by integrating non-academic services like financial payments, housing, and dining to create a more comprehensive platform. This added integration ensures students can manage all aspects of their college experience in one place. As a result, our solution not only tracks academic progress but also simplifies life beyond the classroom.

2. Bidyarthy, A. S., & Kumar, A. (2012) [2] – Student Database Management System: This paper automates the management of academic data, including grades, attendance, and course schedules. While it efficiently handles academic processes, its scope is restricted to academic records. Our project expands on this by integrating other areas of student life, such as housing assignments, dining plans, and financial payments. This holistic approach ensures students have a unified platform to manage both academic and non-academic tasks. With a broader scope, our system meets a wider range of student needs.

3. Tamboli, A. (2017) [3] – Institute Administration Automation and Student Database Management System: This paper emphasizes the automation of institutional processes, focusing on course registration, admissions, and student records through a software platform. Although it improves institutional efficiency, the system is limited to academic and administrative tasks. Our project extends the automation to areas like housing and dining, along with financial accounts, providing students with a seamless experience. Additionally, our platform includes personalized features, such as roommate agreements and housing costs, designed to address the specific needs of college students at institutions like UTD. This personalization makes our solution more relevant to the everyday lives of students.

3. Design & Implementation (Phase I):

3.1. Normalization of EER Conceptual Data Model:

Considering our Original EER design from Final Project Deliverable 1, Section 3.1:

- Student - Already in 3NF
- Event - Not in 3NF
- Course - Not in 3NF
- Professor - Already in 3NF
- TA - Already in 3NF
- Assignment (Supertype) - Already in 3NF
- Finances - Not in 3NF
- Dining - Not in 3NF
- Housing - Already in 3NF
- RoommateAgreement (Weak entity of Housing) - Already in 3NF

Normalizing Event to 3NF:

- Reasoning: Event is not in 3NF but rather in 1NF since it contains a multivalued attribute: EventType.
- Solution: An additional EventType entity will be created to contain information on EventType. This allows for each Event to have multiple EventTypes. EventType is a strong entity since multiple Events can be the same EventType.

Normalizing Course to 3NF:

- Reasoning: Course is not in 3NF and rather in 1NF since it contains a multivalued and composite attribute: Schedule. Although Schedule wasn't previously defined to be multivalued, after further analysis, Schedule should be multivalued as well. For example, CS4347.005 meets on two different days, thus, it would need to contain multiple Schedule values.
- Solution: An additional CourseSchedule weak entity will be created to contain information on the day, start_time, end_time, and ClassroomNo. CourseSchedule should be a weak entity of Course since the existence of CourseSchedule depends on Course.

Normalizing Finances and Dining to 3NF:

- Reasoning: Finances and Dining are both not in 3NF and rather in 1NF since they each contain a multivalued attribute: MealPlanType for Dining and FinancesType for Finances.
- Solution: We will simply define both MealPlanType and FinancesType as regular attributes that are not multivalued. After further analysis, a tuple in the Finances table can only ever be one FinancesType at once. For instance, a Finances tuple cannot be both for tuition and books. Similarly, a tuple in Dining can only ever be one MealPlanType at once since each student is only able to purchase one MealPlanType.

Entities, Attributes, Data Types after normalization:

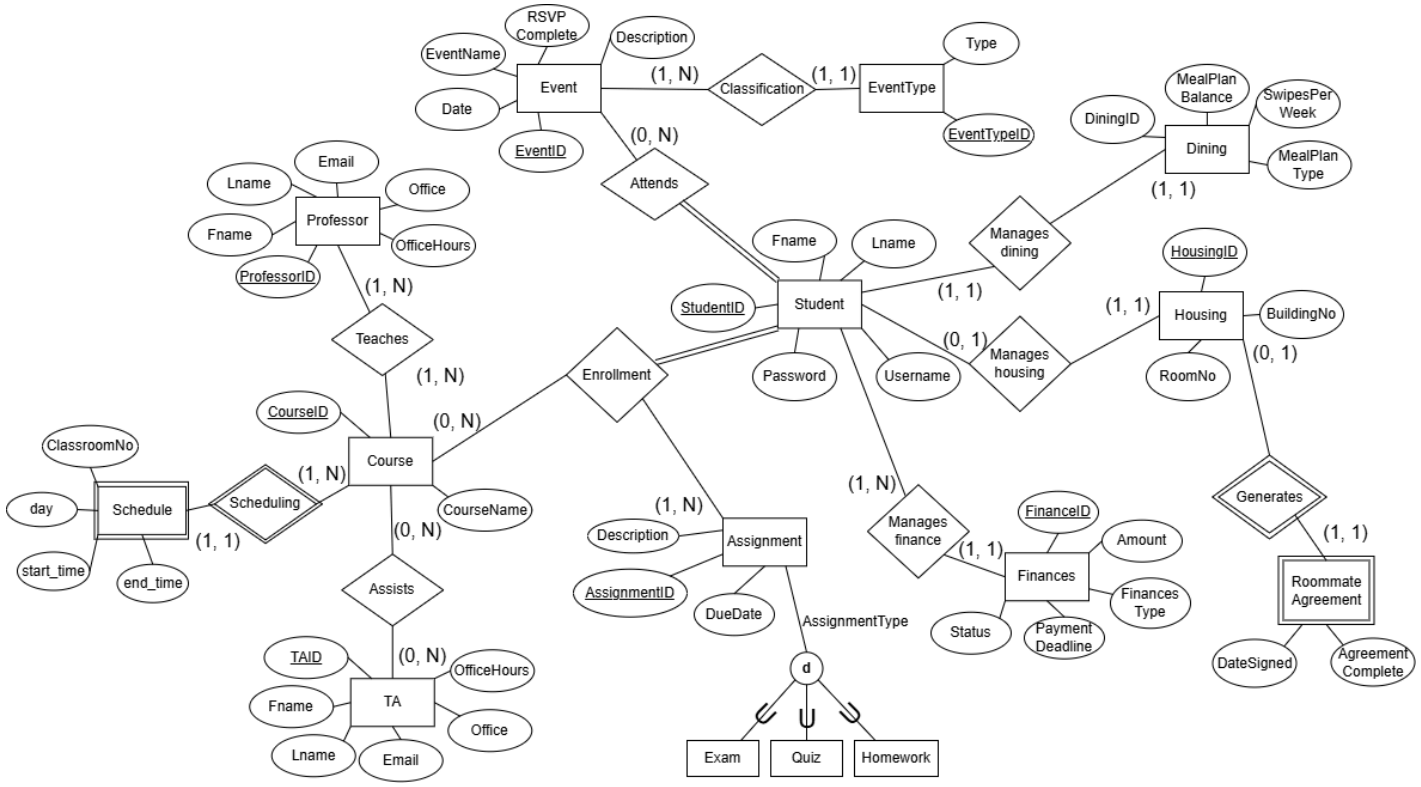
- No changes (already normalized): Student, Professor, TA, Assignment (Supertype), Housing, RoommateAgreement (Weak entity of Housing)
- Event
    - EventID INT
    - Date DATETIME
    - EventName VARCHAR(45)
    - RSVPComplete BIT
    - Description VARCHAR(45)
- EventType
    - EventTypeID INT
    - Type VARCHAR(20)
- Course
    - CourseID INT
    - CourseName VARCHAR(45)
- Schedule (Weak entity of Course)
    - ClassroomNo VARCHAR(10)
    - Day VARCHAR(10)
    - Start_time TIME
    - End_time TIME
- Finances
    - FinancesID
    - Amount DECIMAL
    - FinancesType VARCHAR(45)
        - Regular attribute
    - PaymentDeadline DATETIME
    - Status BIT
- Dining
    - DiningID INT
    - MealPlanBalance DECIMAL
    - SwipesPerWeek INT
    - MealPlanType VARCHAR(45)
        - Regular attribute

Relationships after normalization:

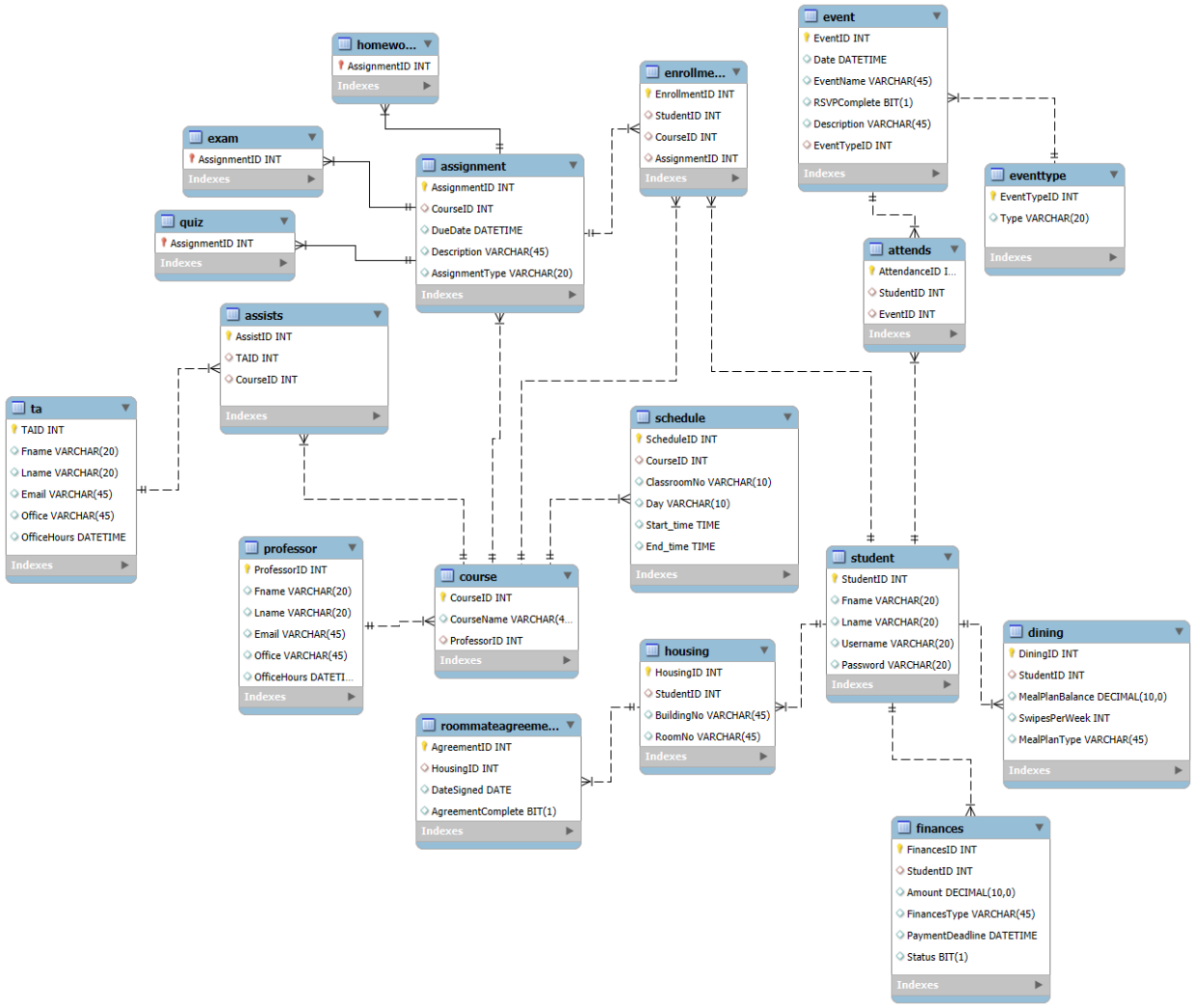
1. Attends: Student (Many) - Event (Many)

2. Classification: Event (One) - EventType (Many)
3. Teaches: Course (Many) - Professor (One)
4. Scheduling: Course (One) - Schedule (Many)
5. Assists: Course (One) - TA (Many)
6. Enrollment: Student (Many), Course (Many), Assignment (Many)
7. Manages finance: Student (One) - Finances (Many)
8. Manages dining: Student (One) - Dining (One)
9. Manages housing: Student (One) - Housing (One)
10. Generates: Housing (One) - RoommateAgreement (One)
11. Super/Subclass relationship (disjoint) - Assignment superclass with exam, quiz, homework subclasses

New EER Diagram:



3.2. Relational Data Model Design Using Normalized EER Diagram:



3.3. Create your Normalized Database and Populate:

- Student Table

SQLQuery1.sql - D...JBURK9\abbas (58))* | Create Student Rec...JBURK9\abbas (62))

```sql
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Fname VARCHAR(20),
    Lname VARCHAR(20),
    Username VARCHAR(20),
    Password VARCHAR(20)
);

INSERT INTO Student (StudentID, Fname, Lname, Username, Password) VALUES
(1, 'Alice', 'Johnson', 'alicej', 'password1'),
(2, 'Bob', 'Smith', 'bobsmith', 'password2'),
(3, 'Charlie', 'Brown', 'charlieb', 'password3'),
(4, 'Diana', 'Prince', 'dianap', 'password4'),
(5, 'Evan', 'Williams', 'evanw', 'password5'),
(6, 'Fiona', 'White', 'fionaw', 'password6'),
(7, 'George', 'Miller', 'georgem', 'password7'),
(8, 'Hannah', 'Lee', 'hannahlee', 'password8'),
(9, 'Ivy', 'Green', 'ivyg', 'password9'),
(10, 'Jake', 'Long', 'jakelong', 'password10');

SELECT * FROM Student;
```

110 %

Results | Messages

	StudentID	Fname	Lname	Username	Password
1	1	Alice	Johnson	alicej	password1
2	2	Bob	Smith	bobsmith	password2
3	3	Charlie	Brown	charlieb	password3
4	4	Diana	Prince	dianap	password4
5	5	Evan	Williams	evanw	password5
6	6	Fiona	White	fionaw	password6
7	7	George	Miller	georgem	password7
8	8	Hannah	Lee	hannahlee	password8
9	9	Ivy	Green	ivyg	password9
10	10	Jake	Long	jakelong	password10

Query executed successfully. | DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

- Event Table

SQLQuery1.sql - D...JBURK9\abbas (58))* | Create Student Rec...JBURK9\abbas (62))

```sql
CREATE TABLE Event (
    EventID INT PRIMARY KEY,
    Date DATETIME,
    EventName VARCHAR(45),
    RSVPComplete BIT,
    Description VARCHAR(45)
);

INSERT INTO Event (EventID, Date, EventName, RSVPComplete, Description) VALUES
(1, '2024-12-01 18:00:00', 'End of Semester Party', 1, 'Celebrate the end of semester'),
(2, '2024-11-20 15:00:00', 'Career Fair', 0, 'Meet potential employers'),
(3, '2024-10-15 12:00:00', 'Alumni Networking', 1, 'Networking with alumni'),
(4, '2024-09-10 09:00:00', 'Orientation', 1, 'New student orientation'),
(5, '2024-08-25 11:00:00', 'Welcome Back BBQ', 0, 'Kickoff the semester'),
(6, '2024-12-05 17:00:00', 'Holiday Gala', 0, 'End-of-year celebration'),
(7, '2024-10-20 10:00:00', 'Leadership Workshop', 1, 'Develop leadership skills'),
(8, '2024-11-01 14:00:00', 'Tech Expo', 1, 'Explore new technologies'),
(9, '2024-12-15 13:00:00', 'Winter Wonderland', 0, 'Winter-themed event'),
(10, '2024-11-25 09:30:00', 'Community Service Day', 1, 'Day of giving back');

SELECT * FROM Event;
```

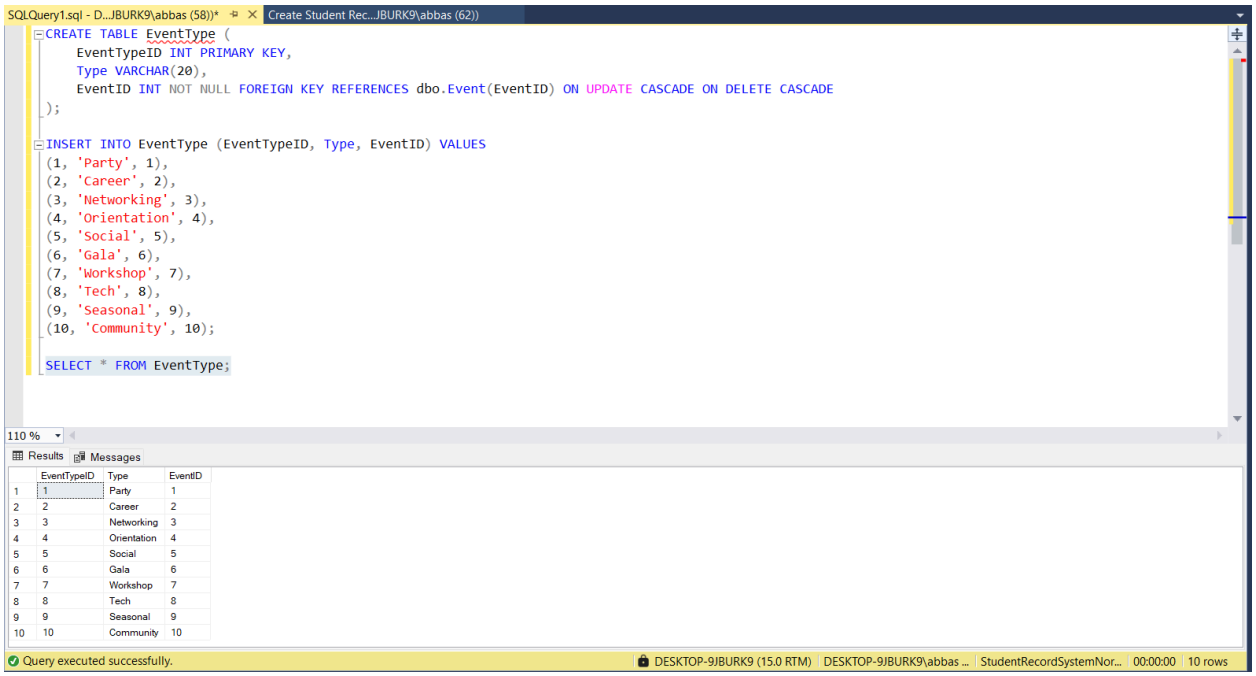
110 %

Results | Messages

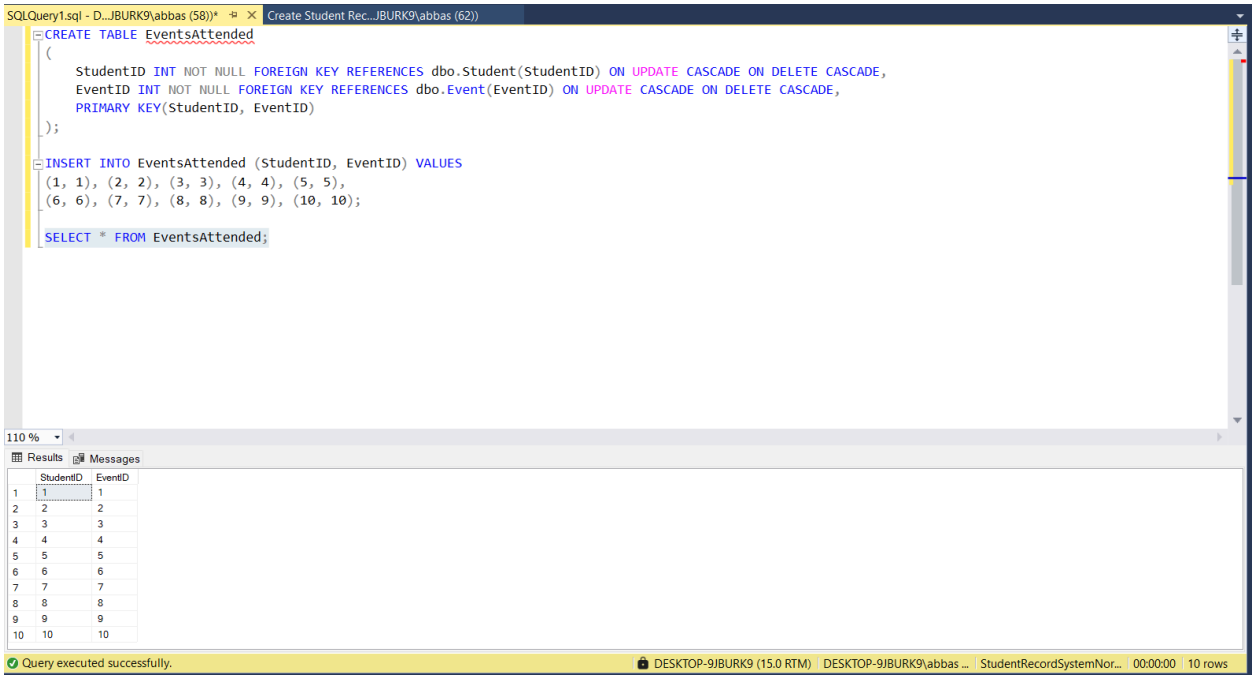
	EventID	Date	EventName	RSVPComplete	Description
1	1	2024-12-01 18:00:00.000	End of Semester Party	1	Celebrate the end of semester
2	2	2024-11-20 15:00:00.000	Career Fair	0	Meet potential employers
3	3	2024-10-15 12:00:00.000	Alumni Networking	1	Networking with alumni
4	4	2024-09-10 09:00:00.000	Orientation	1	New student orientation
5	5	2024-08-25 11:00:00.000	Welcome Back BBQ	0	Kickoff the semester
6	6	2024-12-05 17:00:00.000	Holiday Gala	0	End-of-year celebration
7	7	2024-10-20 10:00:00.000	Leadership Workshop	1	Develop leadership skills
8	8	2024-11-01 14:00:00.000	Tech Expo	1	Explore new technologies
9	9	2024-12-15 13:00:00.000	Winter Wonderland	0	Winter-themed event
10	10	2024-11-25 09:30:00.000	Community Service Day	1	Day of giving back

Query executed successfully. | DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

- EventType Table



- EventsAttended Table



- Professor Table

```sql
CREATE TABLE Professor (
    ProfessorID INT PRIMARY KEY,
    Fname VARCHAR(20),
    Lname VARCHAR(20),
    Email VARCHAR(45),
    Office VARCHAR(45),
    OfficeHours DATETIME
);

INSERT INTO Professor (ProfessorID, Fname, Lname, Email, Office, OfficeHours) VALUES
(1, 'John', 'Doe', 'jdoe@university.edu', 'Bldg 1 Rm 101', '2024-12-05 10:00:00'),
(2, 'Emily', 'Smith', 'esmith@university.edu', 'Bldg 1 Rm 102', '2024-12-05 11:00:00'),
(3, 'Mark', 'Brown', 'mbrown@university.edu', 'Bldg 1 Rm 103', '2024-12-05 12:00:00'),
(4, 'Sarah', 'Johnson', 'sjohnson@university.edu', 'Bldg 2 Rm 201', '2024-12-05 09:00:00'),
(5, 'Robert', 'Davis', 'rdavis@university.edu', 'Bldg 2 Rm 202', '2024-12-05 13:00:00'),
(6, 'Linda', 'Clark', 'lclark@university.edu', 'Bldg 3 Rm 301', '2024-12-05 14:00:00'),
(7, 'James', 'Lewis', 'jlewis@university.edu', 'Bldg 3 Rm 302', '2024-12-05 15:00:00'),
(8, 'Patricia', 'Walker', 'pwalker@university.edu', 'Bldg 4 Rm 401', '2024-12-05 08:00:00'),
(9, 'Michael', 'Hall', 'mhall@university.edu', 'Bldg 4 Rm 402', '2024-12-05 16:00:00'),
(10, 'Barbara', 'Young', 'byoung@university.edu', 'Bldg 5 Rm 501', '2024-12-05 17:00:00');

SELECT * FROM Professor;
```

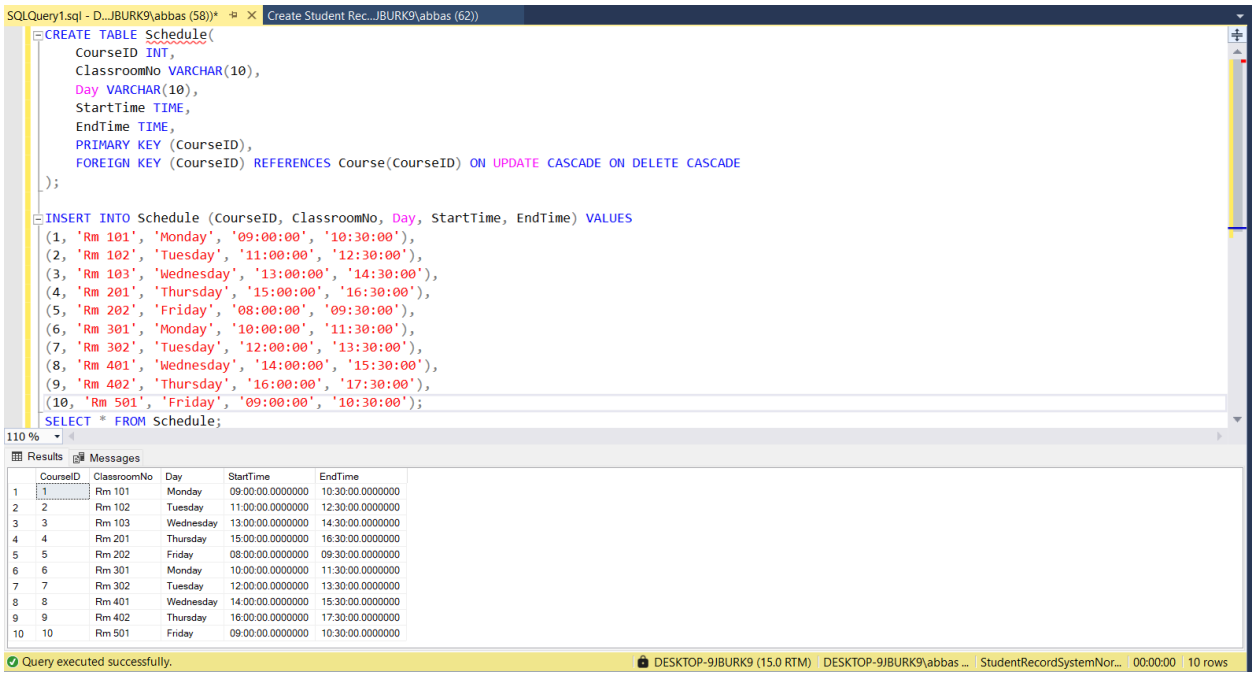
	ProfessorID	Fname	Lname	Email	Office	OfficeHours
1	1	John	Doe	jdoe@university.edu	Bldg 1 Rm 101	2024-12-05 10:00:00.000
2	2	Emily	Smith	esmith@university.edu	Bldg 1 Rm 102	2024-12-05 11:00:00.000
3	3	Mark	Brown	mbrown@university.edu	Bldg 1 Rm 103	2024-12-05 12:00:00.000
4	4	Sarah	Johnson	sjohnson@university.edu	Bldg 2 Rm 201	2024-12-05 09:00:00.000
5	5	Robert	Davis	rdavis@university.edu	Bldg 2 Rm 202	2024-12-05 13:00:00.000
6	6	Linda	Clark	lclark@university.edu	Bldg 3 Rm 301	2024-12-05 14:00:00.000
7	7	James	Lewis	jlewis@university.edu	Bldg 3 Rm 302	2024-12-05 15:00:00.000
8	8	Patricia	Walker	pwalker@university.edu	Bldg 4 Rm 401	2024-12-05 08:00:00.000
9	9	Michael	Hall	mhall@university.edu	Bldg 4 Rm 402	2024-12-05 16:00:00.000
10	10	Barbara	Young	byoung@university.edu	Bldg 5 Rm 501	2024-12-05 17:00:00.000

- Course Table

```sql
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(45),
    ProfessorID INT FOREIGN KEY REFERENCES dbo.Professor(ProfessorID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO Course (CourseID, CourseName, ProfessorID) VALUES
(1, 'Intro to Computer Science', 1),
(2, 'Calculus I', 2),
(3, 'Physics I', 3),
(4, 'Chemistry I', 4),
(5, 'Biology I', 5),
(6, 'Psychology', 6),
(7, 'Sociology', 7),
(8, 'Art History', 8),
(9, 'World Literature', 9),
(10, 'Political Science', 10);

SELECT * FROM Course;
```

	CourseID	CourseName	ProfessorID
1	1	Intro to Computer Science	1
2	2	Calculus I	2
3	3	Physics I	3
4	4	Chemistry I	4
5	5	Biology I	5
6	6	Psychology	6
7	7	Sociology	7
8	8	Art History	8
9	9	World Literature	9
10	10	Political Science	10

- Schedule Table



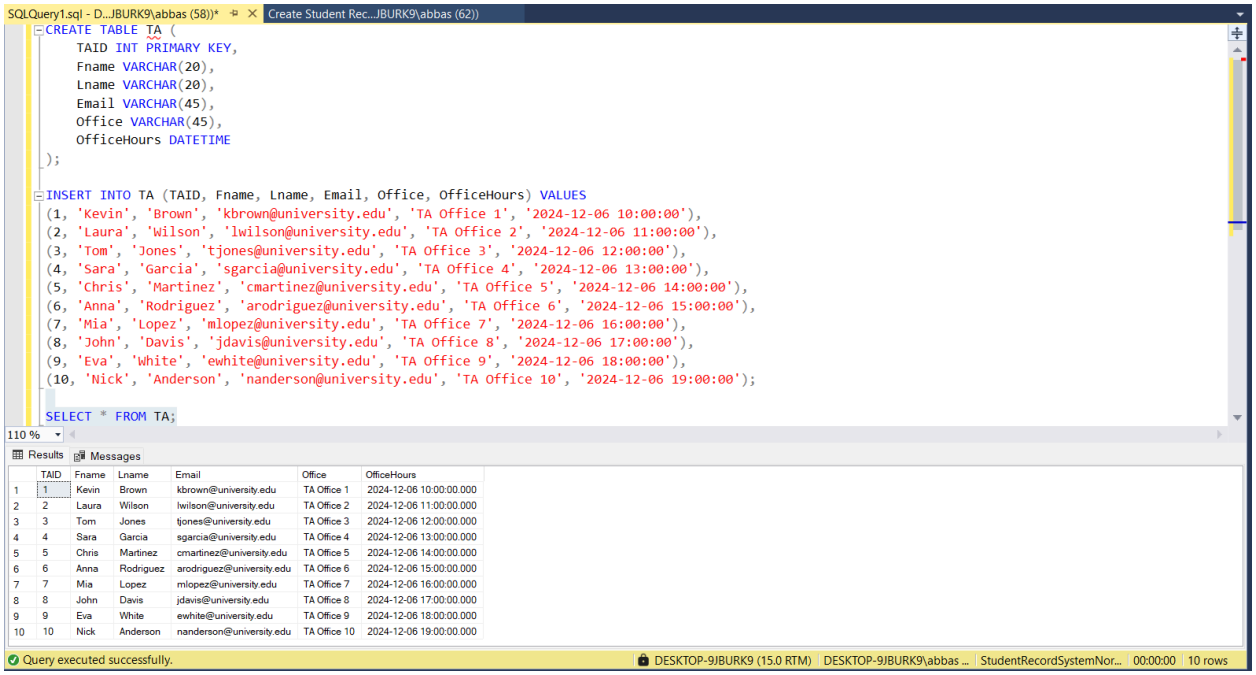
```sql
CREATE TABLE Schedule(
    CourseID INT,
    ClassroomNo VARCHAR(10),
    Day VARCHAR(10),
    StartTime TIME,
    EndTime TIME,
    PRIMARY KEY (CourseID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO Schedule (CourseID, ClassroomNo, Day, StartTime, EndTime) VALUES
(1, 'Rm 101', 'Monday', '09:00:00', '10:30:00'),
(2, 'Rm 102', 'Tuesday', '11:00:00', '12:30:00'),
(3, 'Rm 103', 'Wednesday', '13:00:00', '14:30:00'),
(4, 'Rm 201', 'Thursday', '15:00:00', '16:30:00'),
(5, 'Rm 202', 'Friday', '08:00:00', '09:30:00'),
(6, 'Rm 301', 'Monday', '10:00:00', '11:30:00'),
(7, 'Rm 302', 'Tuesday', '12:00:00', '13:30:00'),
(8, 'Rm 401', 'Wednesday', '14:00:00', '15:30:00'),
(9, 'Rm 402', 'Thursday', '16:00:00', '17:30:00'),
(10, 'Rm 501', 'Friday', '09:00:00', '10:30:00');
SELECT * FROM Schedule;
```

Results

	CourseID	ClassroomNo	Day	StartTime	EndTime
1	1	Rm 101	Monday	09:00:00.0000000	10:30:00.0000000
2	2	Rm 102	Tuesday	11:00:00.0000000	12:30:00.0000000
3	3	Rm 103	Wednesday	13:00:00.0000000	14:30:00.0000000
4	4	Rm 201	Thursday	15:00:00.0000000	16:30:00.0000000
5	5	Rm 202	Friday	08:00:00.0000000	09:30:00.0000000
6	6	Rm 301	Monday	10:00:00.0000000	11:30:00.0000000
7	7	Rm 302	Tuesday	12:00:00.0000000	13:30:00.0000000
8	8	Rm 401	Wednesday	14:00:00.0000000	15:30:00.0000000
9	9	Rm 402	Thursday	16:00:00.0000000	17:30:00.0000000
10	10	Rm 501	Friday	09:00:00.0000000	10:30:00.0000000

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ... StudentRecordSystemNor... 00:00:00 10 rows

- TA Table



```sql
CREATE TABLE TA (
    TAID INT PRIMARY KEY,
    Fname VARCHAR(20),
    Lname VARCHAR(20),
    Email VARCHAR(45),
    Office VARCHAR(45),
    OfficeHours DATETIME
);

INSERT INTO TA (TAID, Fname, Lname, Email, Office, OfficeHours) VALUES
(1, 'Kevin', 'Brown', 'kbrown@university.edu', 'TA Office 1', '2024-12-06 10:00:00'),
(2, 'Laura', 'Wilson', 'lwilson@university.edu', 'TA Office 2', '2024-12-06 11:00:00'),
(3, 'Tom', 'Jones', 'tjones@university.edu', 'TA Office 3', '2024-12-06 12:00:00'),
(4, 'Sara', 'Garcia', 'sgarcia@university.edu', 'TA Office 4', '2024-12-06 13:00:00'),
(5, 'Chris', 'Martinez', 'cmartinez@university.edu', 'TA Office 5', '2024-12-06 14:00:00'),
(6, 'Anna', 'Rodriguez', 'arodriguez@university.edu', 'TA Office 6', '2024-12-06 15:00:00'),
(7, 'Mia', 'Lopez', 'mlopez@university.edu', 'TA Office 7', '2024-12-06 16:00:00'),
(8, 'John', 'Davis', 'jdavis@university.edu', 'TA Office 8', '2024-12-06 17:00:00'),
(9, 'Eva', 'White', 'ewhite@university.edu', 'TA Office 9', '2024-12-06 18:00:00'),
(10, 'Nick', 'Anderson', 'nanderson@university.edu', 'TA Office 10', '2024-12-06 19:00:00');

SELECT * FROM TA;
```

Results

	TAID	Fname	Lname	Email	Office	OfficeHours
1	1	Kevin	Brown	kbrown@university.edu	TA Office 1	2024-12-06 10:00:00.000
2	2	Laura	Wilson	lwilson@university.edu	TA Office 2	2024-12-06 11:00:00.000
3	3	Tom	Jones	tjones@university.edu	TA Office 3	2024-12-06 12:00:00.000
4	4	Sara	Garcia	sgarcia@university.edu	TA Office 4	2024-12-06 13:00:00.000
5	5	Chris	Martinez	cmartinez@university.edu	TA Office 5	2024-12-06 14:00:00.000
6	6	Anna	Rodriguez	arodriguez@university.edu	TA Office 6	2024-12-06 15:00:00.000
7	7	Mia	Lopez	mlopez@university.edu	TA Office 7	2024-12-06 16:00:00.000
8	8	John	Davis	jdavis@university.edu	TA Office 8	2024-12-06 17:00:00.000
9	9	Eva	White	ewhite@university.edu	TA Office 9	2024-12-06 18:00:00.000
10	10	Nick	Anderson	nanderson@university.edu	TA Office 10	2024-12-06 19:00:00.000

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ... StudentRecordSystemNor... 00:00:00 10 rows

- TAAssists Table

```sql
CREATE TABLE TAAssists (
    TAID INT,
    CourseID INT,
    PRIMARY KEY (TAID, CourseID),
    FOREIGN KEY (TAID) REFERENCES TA(TAID) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO TAAssists (TAID, CourseID) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
(6, 6), (7, 7), (8, 8), (9, 9), (10, 10);

SELECT * FROM TAAssists;
```

Results | Messages

	TAID	CourseID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

- Assignment Table

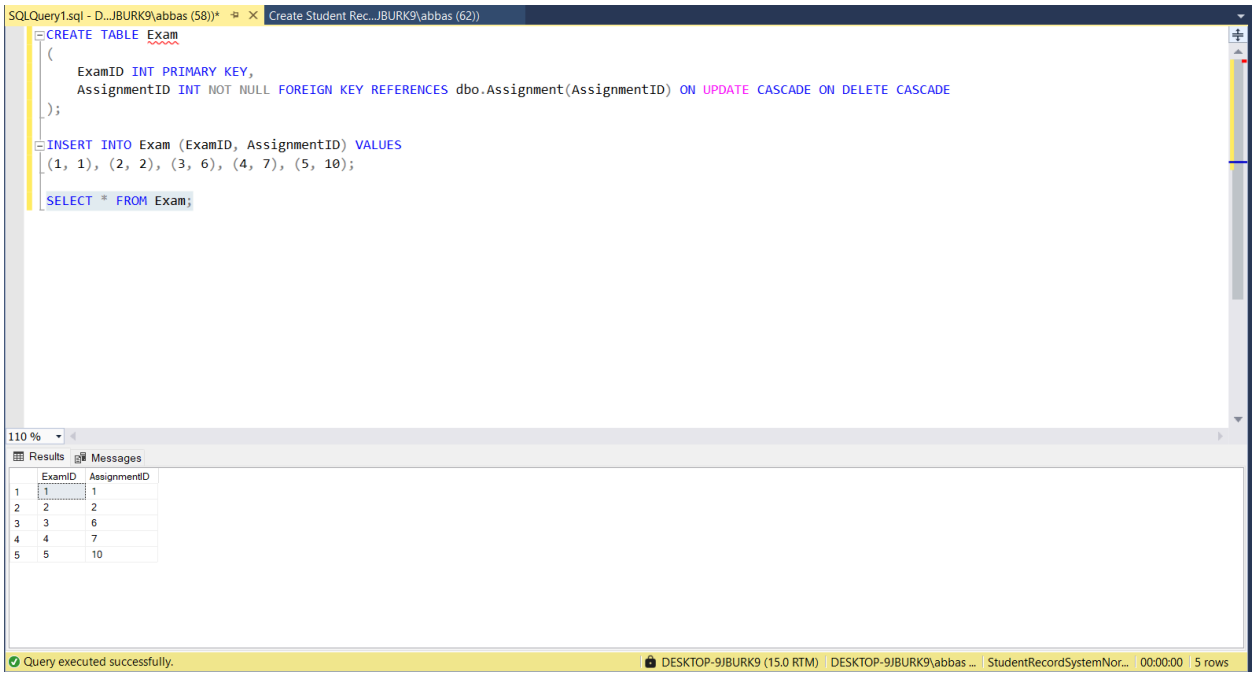
```sql
CREATE TABLE Assignment (
    AssignmentID INT PRIMARY KEY,
    CourseID INT FOREIGN KEY REFERENCES dbo.Course(CourseID) ON UPDATE CASCADE ON DELETE CASCADE,
    DueDate DATETIME,
    Description VARCHAR(45),
    AssignmentType VARCHAR(20) -- 'Exam', 'Homework', 'Quiz'
);

INSERT INTO Assignment (AssignmentID, CourseID, DueDate, Description, AssignmentType) VALUES
(1, 1, '2024-11-15 23:59:00', 'Midterm Exam', 'Exam'),
(2, 2, '2024-12-01 23:59:00', 'Final Exam', 'Exam'),
(3, 3, '2024-11-20 23:59:00', 'Homework 1', 'Homework'),
(4, 4, '2024-11-25 23:59:00', 'Quiz 1', 'Quiz'),
(5, 5, '2024-11-30 23:59:00', 'Homework 2', 'Homework'),
(6, 6, '2024-12-05 23:59:00', 'Final Exam', 'Exam'),
(7, 7, '2024-12-10 23:59:00', 'Project Presentation', 'Exam'),
(8, 8, '2024-12-15 23:59:00', 'Quiz 2', 'Quiz'),
(9, 9, '2024-12-20 23:59:00', 'Homework 3', 'Homework'),
(10, 10, '2024-12-25 23:59:00', 'Final Paper', 'Exam');

SELECT * FROM Assignment;
```

Results | Messages

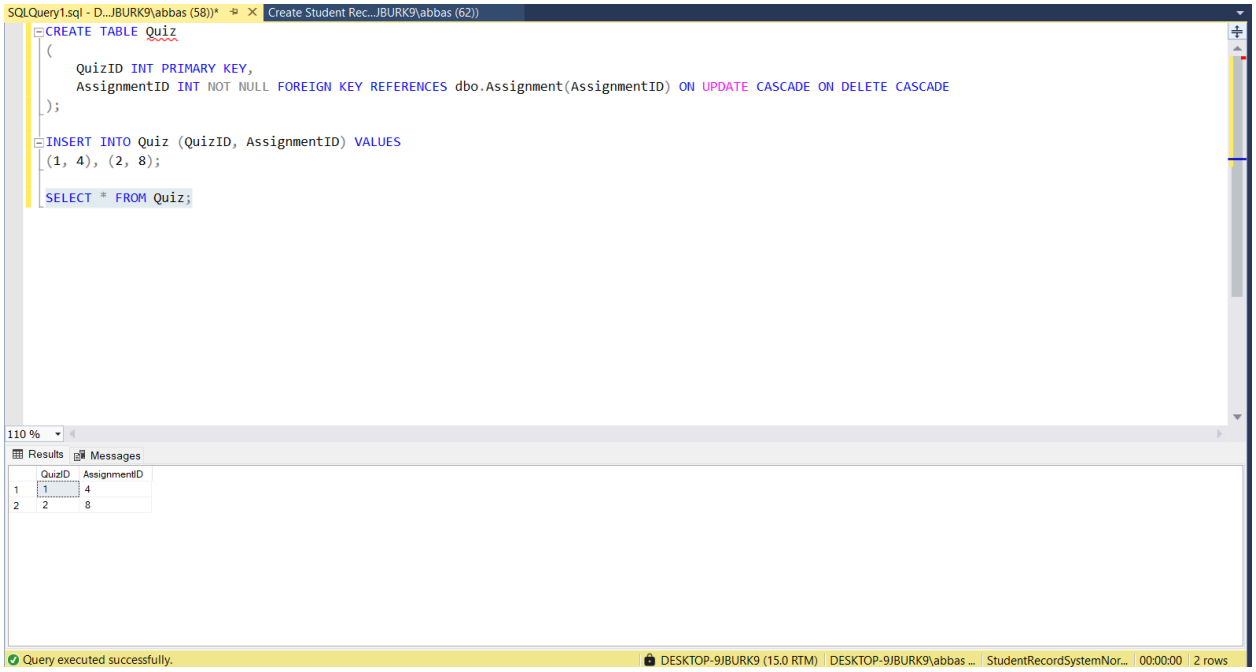
	AssignmentID	CourseID	DueDate	Description	AssignmentType
1	1	1	2024-11-15 23:59:00.000	Midterm Exam	Exam
2	2	2	2024-12-01 23:59:00.000	Final Exam	Exam
3	3	3	2024-11-20 23:59:00.000	Homework 1	Homework
4	4	4	2024-11-25 23:59:00.000	Quiz 1	Quiz
5	5	5	2024-11-30 23:59:00.000	Homework 2	Homework
6	6	6	2024-12-05 23:59:00.000	Final Exam	Exam
7	7	7	2024-12-10 23:59:00.000	Project Presentation	Exam
8	8	8	2024-12-15 23:59:00.000	Quiz 2	Quiz
9	9	9	2024-12-20 23:59:00.000	Homework 3	Homework
10	10	10	2024-12-25 23:59:00.000	Final Paper	Exam

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

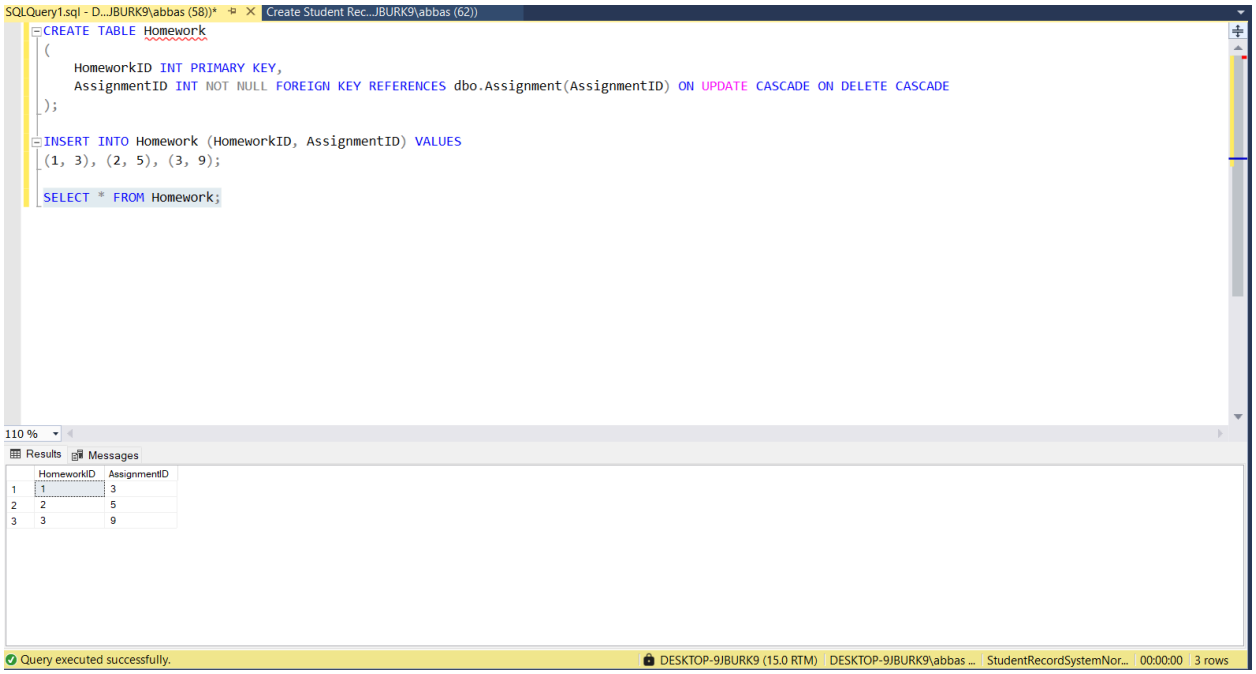
- Exam Table



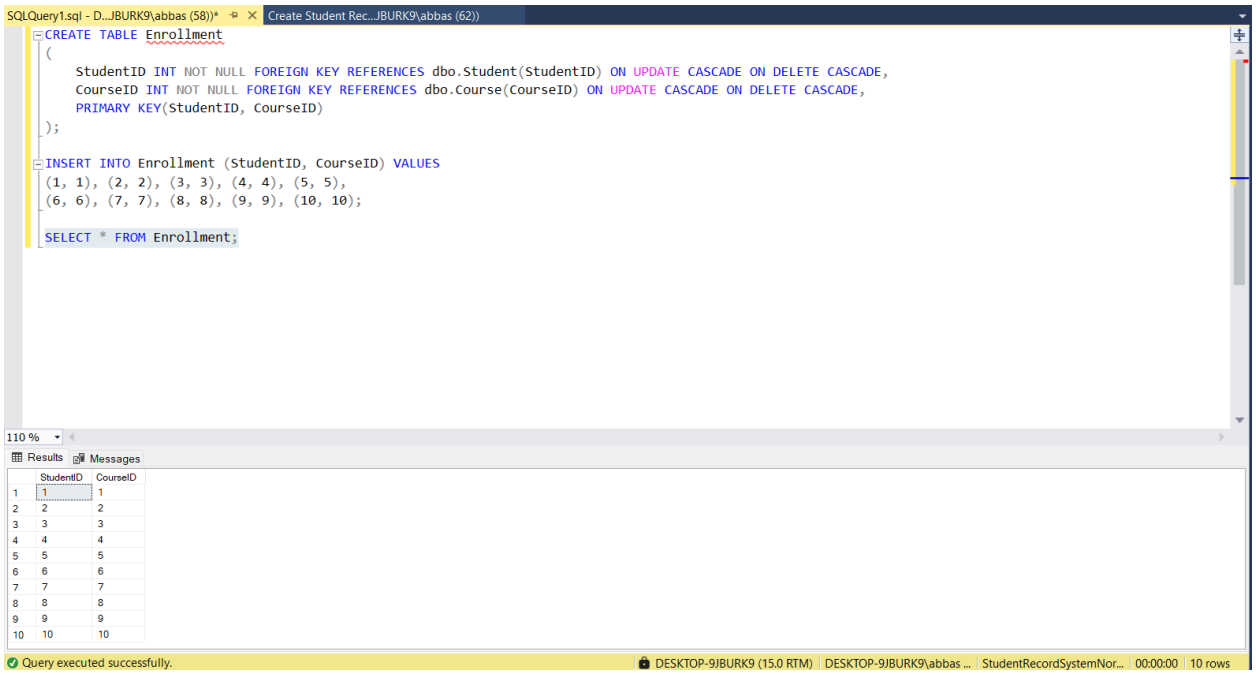
- Quiz Table



- Homework Table



- Enrollment Table



- Finances Table

SQLQuery1.sql - D...JBURK9\abbas (58))* | Create Student Rec...JBURK9\abbas (62))

```sql
CREATE TABLE Finances (
    FinancesID INT PRIMARY KEY,
    StudentID INT,
    Amount DECIMAL(10, 2),
    FinancesType VARCHAR(45),
    PaymentDeadline DATETIME,
    Status BIT,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO Finances (FinancesID, StudentID, Amount, FinancesType, PaymentDeadline, Status) VALUES
(1, 1, 5000.00, 'Tuition', '2024-12-10', 1),
(2, 2, 2500.00, 'Dorm', '2024-11-30', 0),
(3, 3, 200.00, 'Meal Plan', '2024-12-15', 1),
(4, 4, 1500.00, 'Tuition', '2024-12-10', 1),
(5, 5, 1000.00, 'Library Fee', '2024-11-25', 0),
(6, 6, 300.00, 'Meal Plan', '2024-12-05', 1),
(7, 7, 150.00, 'Club Dues', '2024-12-20', 1),
(8, 8, 75.00, 'Lab Fee', '2024-11-28', 0),
(9, 9, 500.00, 'Parking', '2024-12-01', 1),
(10, 10, 800.00, 'Gym Membership', '2024-12-18', 0);
SELECT * FROM Finances;
```

110 %

Results | Messages

	FinancesID	StudentID	Amount	FinancesType	PaymentDeadline	Status
1	1	1	5000.00	Tuition	2024-12-10 00:00:00.000	1
2	2	2	2500.00	Dorm	2024-11-30 00:00:00.000	0
3	3	3	200.00	Meal Plan	2024-12-15 00:00:00.000	1
4	4	4	1500.00	Tuition	2024-12-10 00:00:00.000	1
5	5	5	1000.00	Library Fee	2024-11-25 00:00:00.000	0
6	6	6	300.00	Meal Plan	2024-12-05 00:00:00.000	1
7	7	7	150.00	Club Dues	2024-12-20 00:00:00.000	1
8	8	8	75.00	Lab Fee	2024-11-28 00:00:00.000	0
9	9	9	500.00	Parking	2024-12-01 00:00:00.000	1
10	10	10	800.00	Gym Membership	2024-12-18 00:00:00.000	0

Query executed successfully. | DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

- Dining Table

SQLQuery1.sql - D...JBURK9\abbas (58))* | Create Student Rec...JBURK9\abbas (62))

```sql
CREATE TABLE Dining (
    DiningID INT PRIMARY KEY,
    StudentID INT,
    MealPlanBalance DECIMAL(10, 2),
    SwipesPerWeek INT,
    MealPlanType VARCHAR(45),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO Dining (DiningID, StudentID, MealPlanBalance, SwipesPerWeek, MealPlanType) VALUES
(1, 1, 1200.00, 14, 'Unlimited'),
(2, 2, 800.00, 10, 'Standard'),
(3, 3, 1500.00, 21, 'Premium'),
(4, 4, 1000.00, 14, 'Unlimited'),
(5, 5, 500.00, 7, 'Basic'),
(6, 6, 750.00, 10, 'Standard'),
(7, 7, 1300.00, 21, 'Premium'),
(8, 8, 600.00, 10, 'Standard'),
(9, 9, 900.00, 14, 'Unlimited'),
(10, 10, 400.00, 7, 'Basic');

SELECT * FROM Dining;
```

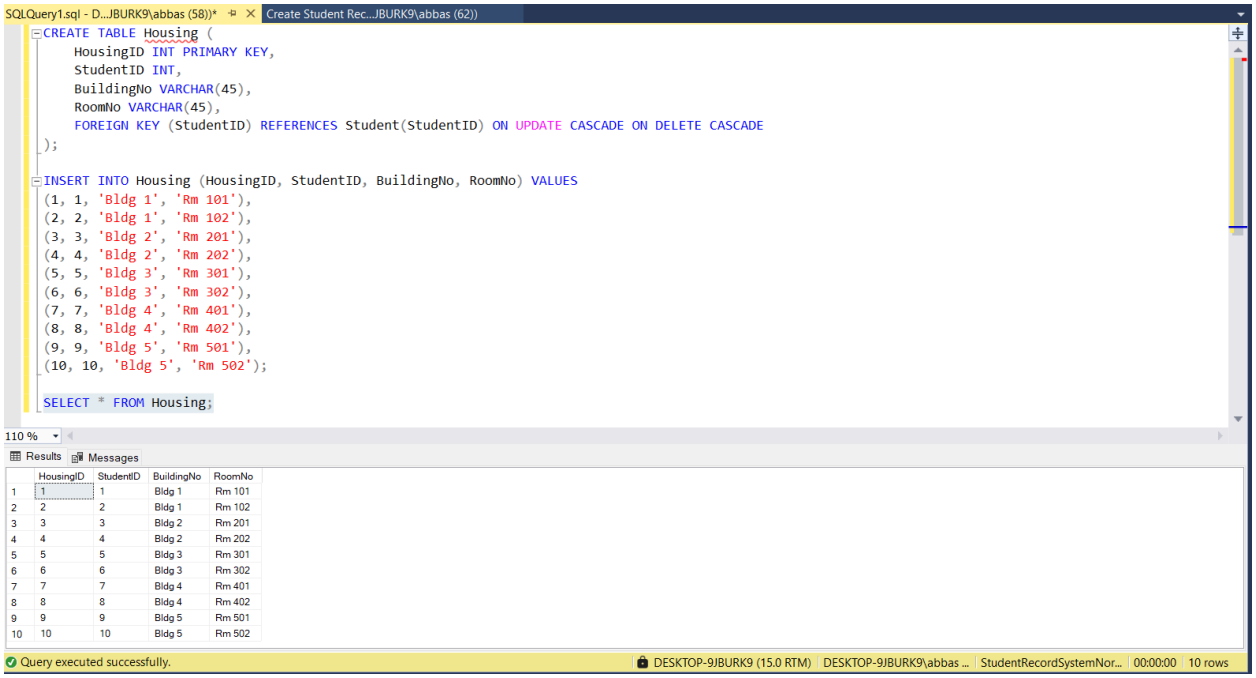
110 %

Results | Messages

	DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1	1	1200.00	14	Unlimited
2	2	2	800.00	10	Standard
3	3	3	1500.00	21	Premium
4	4	4	1000.00	14	Unlimited
5	5	5	500.00	7	Basic
6	6	6	750.00	10	Standard
7	7	7	1300.00	21	Premium
8	8	8	600.00	10	Standard
9	9	9	900.00	14	Unlimited
10	10	10	400.00	7	Basic

Query executed successfully. | DESKTOP-9JBURK9 (15.0 RTM) | DESKTOP-9JBURK9\abbas ... | StudentRecordSystemNor... | 00:00:00 | 10 rows

- Housing Table



```
CREATE TABLE Housing (
    HousingID INT PRIMARY KEY,
    StudentID INT,
    BuildingNo VARCHAR(45),
    RoomNo VARCHAR(45),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO Housing (HousingID, StudentID, BuildingNo, RoomNo) VALUES
(1, 1, 'Bldg 1', 'Rm 101'),
(2, 2, 'Bldg 1', 'Rm 102'),
(3, 3, 'Bldg 2', 'Rm 201'),
(4, 4, 'Bldg 2', 'Rm 202'),
(5, 5, 'Bldg 3', 'Rm 301'),
(6, 6, 'Bldg 3', 'Rm 302'),
(7, 7, 'Bldg 4', 'Rm 401'),
(8, 8, 'Bldg 4', 'Rm 402'),
(9, 9, 'Bldg 5', 'Rm 501'),
(10, 10, 'Bldg 5', 'Rm 502');

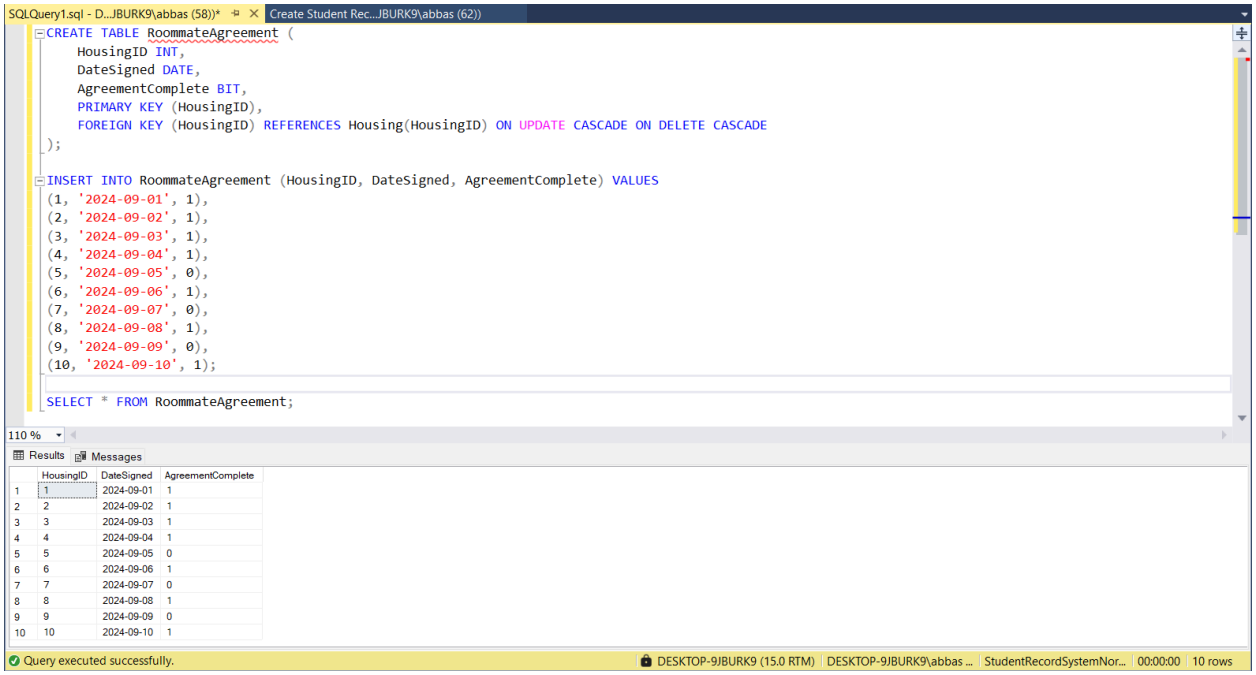
SELECT * FROM Housing;
```

Results

	HousingID	StudentID	BuildingNo	RoomNo
1	1	1	Bldg 1	Rm 101
2	2	2	Bldg 1	Rm 102
3	3	3	Bldg 2	Rm 201
4	4	4	Bldg 2	Rm 202
5	5	5	Bldg 3	Rm 301
6	6	6	Bldg 3	Rm 302
7	7	7	Bldg 4	Rm 401
8	8	8	Bldg 4	Rm 402
9	9	9	Bldg 5	Rm 501
10	10	10	Bldg 5	Rm 502

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ... StudentRecordSystemNor... 00:00:00 10 rows

- RoommateAgreement Table



```
CREATE TABLE RoommateAgreement (
    HousingID INT,
    DateSigned DATE,
    AgreementComplete BIT,
    PRIMARY KEY (HousingID),
    FOREIGN KEY (HousingID) REFERENCES Housing(HousingID) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO RoommateAgreement (HousingID, DateSigned, AgreementComplete) VALUES
(1, '2024-09-01', 1),
(2, '2024-09-02', 1),
(3, '2024-09-03', 1),
(4, '2024-09-04', 1),
(5, '2024-09-05', 0),
(6, '2024-09-06', 1),
(7, '2024-09-07', 0),
(8, '2024-09-08', 1),
(9, '2024-09-09', 0),
(10, '2024-09-10', 1);

SELECT * FROM RoommateAgreement;
```

Results

	HousingID	DateSigned	AgreementComplete
1	1	2024-09-01	1
2	2	2024-09-02	1
3	3	2024-09-03	1
4	4	2024-09-04	1
5	5	2024-09-05	0
6	6	2024-09-06	1
7	7	2024-09-07	0
8	8	2024-09-08	1
9	9	2024-09-09	0
10	10	2024-09-10	1

Query executed successfully. DESKTOP-9JBURK9 (15.0 RTM) DESKTOP-9JBURK9\abbas ... StudentRecordSystemNor... 00:00:00 10 rows

3.4. Database Query Execution on your Normalized Database (from inside your SQL client):

Use an SQL platform to provide sample executions for each of the following operations on each table of your normalized database

- Student Table:

- Query:
  - ```
    SELECT * FROM Student WHERE Fname LIKE 'C%';
    ```

    100% 1:2

    Result Grid Filter Rows: Search Edit:

StudentID	Fname	Lname	Username	Password
3	Charlie	Brown	charlieb	password3
NULL	NULL	NULL	NULL	NULL
- Insert:
  - ```
    INSERT INTO Student (StudentID, Fname, Lname, Username, Password)
    VALUES (11, 'Isabella', 'Taylor', 'isabellat', 'password11');

    SELECT * FROM STUDENT;
    ```

    100% 23:4

    Result Grid Filter Rows: Search Edit: Export/Import:

StudentID	Fname	Lname	Username	Password
6	Fiona	White	fionaw	password6
7	George	Miller	georgem	password7
8	Hannah	Lee	hannahlee	password8
9	Ivy	Green	ivyg	password9
10	Jake	Long	jakelong	password10
11	Isabella	Taylor	isabellat	password11
- Delete:
  - ```
    DELETE FROM Student WHERE StudentID = 11;

    SELECT * FROM STUDENT;
    ```

    100% 1:4

    Result Grid Filter Rows: Search Edit:

StudentID	Fname	Lname	Username	Password
5	Evan	Williams	evanw	password5
6	Fiona	White	fionaw	password6
7	George	Miller	georgem	password7
8	Hannah	Lee	hannahlee	password8
9	Ivy	Green	ivyg	password9
10	Jake	Long	jakelong	password10
NULL	NULL	NULL	NULL	NULL
- Update:
  - ```
    UPDATE Student SET Username = 'newusername' WHERE StudentID = 1;

    SELECT * FROM STUDENT;
    ```

    100% 1:2

    Result Grid Filter Rows: Search Edit: Export/Import:

StudentID	Fname	Lname	Username	Password
1	Alice	Johnson	newusername	password1
2	Bob	Smith	bobsmith	password2
3	Charlie	Brown	charlieb	password3
4	Diana	Prince	dianap	password4
5	Evan	Williams	evanw	password5
- Event Table:
  - Query:

```
SELECT * FROM Event WHERE EventDate > '2024-12-01';
```

100% 1:2

Result Grid Filter Rows: Search Edit: Export/Import:

EventID	EventDate	EventName	RSVPComplete	Description
1	2024-12-01 18:00:00	End of Semester Party	1	Celebrate the end of semester
6	2024-12-05 17:00:00	Holiday Gala	0	End-of-year celebration
9	2024-12-15 13:00:00	Winter Wonderland	0	Winter-themed event
NULL	NULL	NULL	NULL	NULL

- Insert:

```
INSERT INTO Event (EventID, EventDate, EventName, RSVPComplete, Description)
VALUES (11, '2024-12-31 20:00:00', 'New Year Eve Party', 0, 'Celebrate the new year');

SELECT * FROM Event;
```

100% 21:4

Result Grid Filter Rows: Search Edit: Export/Import:

EventID	EventDate	EventName	RSVPComplete	Description
6	2024-12-05 17:00:00	Holiday Gala	0	End-of-year celebration
7	2024-10-20 10:00:00	Leadership Workshop	1	Develop leadership skills
8	2024-11-01 14:00:00	Tech Expo	1	Explore new technologies
9	2024-12-15 13:00:00	Winter Wonderland	0	Winter-themed event
10	2024-11-25 09:30:00	Community Service Day	1	Day of giving back
11	2024-12-31 20:00:00	New Year Eve Party	0	Celebrate the new year
NULL	NULL	NULL	NULL	NULL

- Delete:

```
DELETE FROM Event WHERE EventID = 5;

SELECT * FROM Event;
```

100% 37:1

Result Grid Filter Rows: Search Edit: Export/Import:

EventID	EventDate	EventName	RSVPComplete	Description
1	2024-12-01 18:00:00	End of Semester Party	1	Celebrate the end of semester
2	2024-11-20 15:00:00	Career Fair	0	Meet potential employers
3	2024-10-15 12:00:00	Alumni Networking	1	Networking with alumni
4	2024-09-10 09:00:00	Orientation	1	New student orientation
6	2024-12-05 17:00:00	Holiday Gala	0	End-of-year celebration

- Update:

```
UPDATE Event SET RSVPComplete = 1 WHERE EventID = 6;

SELECT * FROM Event;
```

0% 1:2

Result Grid Filter Rows: Search Edit: Export/Import:

EventID	EventDate	EventName	RSVPComplete	Description
1	2024-12-01 18:00:00	End of Semester Party	1	Celebrate the end of semester
2	2024-11-20 15:00:00	Career Fair	0	Meet potential employers
3	2024-10-15 12:00:00	Alumni Networking	1	Networking with alumni
4	2024-09-10 09:00:00	Orientation	1	New student orientation
6	2024-12-05 17:00:00	Holiday Gala	1	End-of-year celebration

- EventType Table:
  - Query:

```
SELECT * FROM EventType WHERE Type = 'Workshop';
```

100% 1:2

Result Grid Filter Rows: Search Edit:

EventTypeID	Type	EventID
7	Workshop	7
NULL	NULL	NULL

  - Insert:

```
INSERT INTO EventType (EventTypeID, Type, EventID)
VALUES (11, 'Seminar', 11);

SELECT * FROM EventType;
```

100% 25:4

Result Grid | Filter Rows: Search | Edit: | Expor

EventTypeID	Type	EventID
7	Workshop	7
8	Tech	8
9	Seasonal	9
10	Community	10
11	Seminar	11
NULL	NULL	NULL

- Delete:

```
DELETE FROM EventType WHERE EventTypeID = 10;

SELECT * FROM EventType;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

EventTypeID	Type	EventID
6	Gala	6
7	Workshop	7
8	Tech	8
9	Seasonal	9
11	Seminar	11
NULL	NULL	NULL

- Update:

```
UPDATE EventType SET Type = 'Webinar' WHERE EventTypeID = 7;

SELECT * FROM EventType;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/Import:

EventTypeID	Type	EventID
1	Party	1
2	Career	2
3	Networking	3
4	Orientation	4
6	Gala	6
7	Webinar	7

- EventsAttended Table:
  - Query:

```
SELECT * FROM EventsAttended WHERE EventID = 3;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

StudentID	EventID
3	3
NULL	NULL

  - Insert:

```
INSERT INTO EventsAttended (StudentID, EventID) VALUES (3, 7);

SELECT * FROM EventsAttended;
```

100% 63:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

StudentID	EventID
3	3
4	4
6	6
3	7
7	7

- Delete:
  - ```
    DELETE FROM EventsAttended WHERE StudentID = 2 AND EventID = 2;

    SELECT * FROM EventsAttended;
    ```

    100% 63:1

    Result Grid | Filter Rows: Search | Edit: | Export/Import:

StudentID	EventID
1	1
3	3
4	4
6	6
- Update:
  - ```
    UPDATE EventsAttended SET EventID = 8 WHERE StudentID = 3 AND EventID = 7;

    SELECT * FROM EventsAttended;
    ```

    100% 39:1

    Result Grid | Filter Rows: Search | Edit: | Export/Import:

StudentID	EventID
1	1
3	3
4	4
6	6
7	7
3	8
8	8
- Professor Table:
  - Query:
    - ```
      SELECT * FROM Professor WHERE HOUR(OfficeHours) < 12;
      ```

      100% 1:2

      Result Grid | Filter Rows: Search | Edit: | Export/Import:

ProfessorID	Fname	Lname	Email	Office	OfficeHours
1	John	Doe	jdoe@university.edu	Bldg 1 Rm 101	2024-12-05 10:00:00
2	Emily	Smith	esmith@university.edu	Bldg 1 Rm 102	2024-12-05 11:00:00
4	Sarah	Johnson	sjohnson@university.edu	Bldg 2 Rm 201	2024-12-05 09:00:00
8	Patricia	Walker	pwalker@university.edu	Bldg 4 Rm 401	2024-12-05 08:00:00
  - Insert:
    - ```
      INSERT INTO Professor (ProfessorID, Fname, Lname, Email, Office, OfficeHours)
      VALUES (11, 'Alice', 'Cooper', 'acooper@university.edu', 'Bldg 6 Rm 601', '2024-12-06 09:00:00');

      SELECT * FROM Professor
      ```

      100% 24:4

      Result Grid | Filter Rows: Search | Edit: | Export/Import:

ProfessorID	Fname	Lname	Email	Office	OfficeHours
8	Patricia	Walker	pwalker@university.edu	Bldg 4 Rm 401	2024-12-05 08:00:00
9	Michael	Hall	mhall@university.edu	Bldg 4 Rm 402	2024-12-05 16:00:00
10	Barbara	Young	byoung@university.edu	Bldg 5 Rm 501	2024-12-05 17:00:00
11	Alice	Cooper	acooper@university.edu	Bldg 6 Rm 601	2024-12-06 09:00:00
  - Delete:
    - ```
      DELETE FROM Professor WHERE ProfessorID = 3;

      SELECT * FROM Professor
      ```

      100% 45:1

      Result Grid | Filter Rows: Search | Edit: | Export/Impor

ProfessorID	Fname	Lname	Email	Office	OfficeHours
1	John	Doe	jdoe@university.edu	Bldg 1 Rm 101	2024-12-05 10:00:00
2	Emily	Smith	esmith@university.edu	Bldg 1 Rm 102	2024-12-05 11:00:00
4	Sarah	Johnson	sjohnson@university.edu	Bldg 2 Rm 201	2024-12-05 09:00:00
  - Update:
      ```

```
UPDATE Professor SET Office = 'Bldg 2 Rm 203' WHERE ProfessorID = 4;

SELECT * FROM Professor
```

Result Grid | Filter Rows: Search | Edit: | Export/Import:

ProfessorID	Fname	Lname	Email	Office	OfficeHours
1	John	Doe	jdoe@university.edu	Bldg 1 Rm 101	2024-12-05 10:00:00
2	Emily	Smith	esmith@university.edu	Bldg 1 Rm 102	2024-12-05 11:00:00
4	Sarah	Johnson	sjohnson@university.edu	Bldg 2 Rm 203	2024-12-05 09:00:00

- Course Table:
  - Query:

```
SELECT * FROM Course WHERE ProfessorID = 2;
```

Result Grid | Filter Rows: Search | Edit:

CourseID	CourseName	ProfessorID
2	Calculus I	2

  - Insert:

```
INSERT INTO Course (CourseID, CourseName, ProfessorID) VALUES (11, 'Data Science Basics', 5);

SELECT * FROM Course;
```

Result Grid | Filter Rows: Search | Edit: | Export/Import:

CourseID	CourseName	ProfessorID
6	Psychology	6
7	Sociology	7
8	Art History	8
9	World Literature	9
10	Political Science	10
11	Data Science Basics	5

  - Delete:

```
DELETE FROM Course WHERE CourseID = 10;

SELECT * FROM Course;
```

Result Grid | Filter Rows: Search | Edit:

CourseID	CourseName	ProfessorID
5	Biology I	5
6	Psychology	6
7	Sociology	7
8	Art History	8
9	World Literature	9
11	Data Science Basics	5

  - Update:

```
UPDATE Course SET ProfessorID = 6 WHERE CourseID = 1;

SELECT * FROM Course;
```

Result Grid | Filter Rows: Search | Edit:

CourseID	CourseName	ProfessorID
1	Intro to Computer Science	6
2	Calculus I	2
4	Chemistry I	4

- Schedule Table:
  - Query:

```
SELECT * FROM Schedule WHERE ClassDay = 'Monday';
```

Result Grid | Filter Rows: Search | Edit:

CourseID	ClassroomNo	ClassDay	StartTime	EndTime
1	Rm 101	Monday	09:00:00	10:30:00
6	Rm 301	Monday	10:00:00	11:30:00
NULL	NULL	NULL	NULL	NULL

- Insert:

```
INSERT INTO Schedule (CourseID, ClassroomNo, ClassDay, StartTime, EndTime)
VALUES (11, 'Rm 601', 'Friday', '10:00:00', '11:30:00');

SELECT * FROM Schedule;
```

Result Grid | Filter Rows: Search | Edit: | Export/Import:

CourseID	ClassroomNo	ClassDay	StartTime	EndTime
6	Rm 301	Monday	10:00:00	11:30:00
7	Rm 302	Tuesday	12:00:00	13:30:00
8	Rm 401	Wednesday	14:00:00	15:30:00
9	Rm 402	Thursday	16:00:00	17:30:00
11	Rm 601	Friday	10:00:00	11:30:00
NULL	NULL	NULL	NULL	NULL

- Delete:

```
DELETE FROM Schedule WHERE CourseID IN (2, 3);

SELECT * FROM Schedule;
```

Result Grid | Filter Rows: Search | Edit:

CourseID	ClassroomNo	ClassDay	StartTime	EndTime
1	Rm 101	Monday	09:00:00	10:30:00
4	Rm 201	Thursday	15:00:00	16:30:00
5	Rm 202	Friday	08:00:00	09:30:00

- Update:

```
UPDATE Schedule SET ClassroomNo = 'Rm 202' WHERE CourseID = 1;

SELECT * FROM Schedule;
```

Result Grid | Filter Rows: Search | Edit: | Export/Import:

CourseID	ClassroomNo	ClassDay	StartTime	EndTime
1	Rm 202	Monday	09:00:00	10:30:00
4	Rm 201	Thursday	15:00:00	16:30:00
5	Rm 202	Friday	08:00:00	09:30:00

- TA Table:
  - Query:

```
SELECT * FROM TA WHERE HOUR(OfficeHours) >= 17;
```

Result Grid | Filter Rows: Search | Edit:

TAID	Fname	Lname	Email	Office	OfficeHours
8	John	Davis	jdavis@university.edu	TA Office 8	2024-12-06 17:00:00
9	Eva	White	ewhite@university.edu	TA Office 9	2024-12-06 18:00:00
10	Nick	Anderson	nanderson@university.edu	TA Office 10	2024-12-06 19:00:00
NULL	NULL	NULL	NULL	NULL	NULL

  - Insert:

```
INSERT INTO TA (TAID, Fname, Lname, Email, Office, OfficeHours)
VALUES (11, 'Ben', 'Taylor', 'btaylor@university.edu', 'TA Office 11', '2024-12-06 14:00:00');

SELECT * FROM TA;
```

100% 18:4

Result Grid | Filter Rows: Search | Edit: | Export/Import:

TAID	Fname	Lname	Email	Office	OfficeHours
7	Mia	Lopez	mlopez@university.edu	TA Office 7	2024-12-06 16:00:00
8	John	Davis	jdavis@university.edu	TA Office 8	2024-12-06 17:00:00
9	Eva	White	ewhite@university.edu	TA Office 9	2024-12-06 18:00:00
10	Nick	Anderson	nanderson@university.edu	TA Office 10	2024-12-06 19:00:00
11	Ben	Taylor	btaylor@university.edu	TA Office 11	2024-12-06 14:00:00
NULL	NULL	NULL	NULL	NULL	NULL

- Delete:

```
DELETE FROM TA WHERE TAID = 4;

SELECT * FROM TA;
```

100% 31:1

Result Grid | Filter Rows: Search | Edit: | Export/I

TAID	Fname	Lname	Email	Office	OfficeHours
1	Kevin	Brown	kbrown@university.edu	TA Office 1	2024-12-06 10:00:00
2	Laura	Wilson	lwilson@university.edu	TA Office 2	2024-12-06 11:00:00
3	Tom	Jones	tjones@university.edu	TA Office 3	2024-12-06 12:00:00
5	Chris	Martinez	cmartinez@university.edu	TA Office 5	2024-12-06 14:00:00

- Update:

```
UPDATE TA SET Email = 'newemail@university.edu' WHERE TAID = 1;

SELECT * FROM TA;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/Import:

TAID	Fname	Lname	Email	Office	OfficeHours
1	Kevin	Brown	newemail@university.edu	TA Office 1	2024-12-06 10:00:00
2	Laura	Wilson	lwilson@university.edu	TA Office 2	2024-12-06 11:00:00
3	Tom	Jones	tjones@university.edu	TA Office 3	2024-12-06 12:00:00

- TAAssists Table:
  - Query:

```
SELECT * FROM TAAssists WHERE TAID = 1;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

TAID	CourseID
1	1
NULL	NULL

  - Insert:

```
INSERT INTO TAAssists (TAID, CourseID) VALUES (11, 1);

SELECT * FROM TAAssists;
```

100% 25:3

Result Grid | Filter Rows: Search | Edit: | Export/

TAID	CourseID
1	1
11	1
2	2
5	5

  - Delete:

```
DELETE FROM TAAssists WHERE TAID = 2 AND CourseID = 2;

SELECT * FROM TAAssists;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/

TAID	CourseID
1	1
11	1
5	5

- Update:

```
UPDATE TAAssists SET CourseID = 5 WHERE TAID IN (5, 6);

SELECT * FROM TAAssists;
```

100% 54:1

Result Grid | Filter Rows: Search | Edit: | Export/

TAID	CourseID
1	1
11	1
5	5
6	5

- Assignment Table:
  - Query:

```
SELECT * FROM Assignment WHERE MONTH(DueDate) = 12 AND YEAR(DueDate) = 2024;
```

100% 77:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

AssignmentID	CourseID	DueDate	Description	AssignmentType
2	2	2024-12-01 23:59:00	Final Exam	Exam
6	6	2024-12-05 23:59:00	Final Exam	Exam
7	7	2024-12-10 23:59:00	Project Presentation	Exam
8	8	2024-12-15 23:59:00	Quiz 2	Quiz
9	9	2024-12-20 23:59:00	Homework 3	Homework
NULL	NULL	NULL	NULL	NULL

  - Insert:

```
INSERT INTO Assignment (AssignmentID, CourseID, DueDate, Description, AssignmentType)
VALUES (11, 1, '2024-12-20 23:59:00', 'Extra Credit Project', 'Homework');

SELECT * FROM Assignment;
```

100% 26:4

Result Grid | Filter Rows: Search | Edit: | Export/Import:

AssignmentID	CourseID	DueDate	Description	AssignmentType
6	6	2024-12-05 23:59:00	Final Exam	Exam
7	7	2024-12-10 23:59:00	Project Presentation	Exam
8	8	2024-12-15 23:59:00	Quiz 2	Quiz
9	9	2024-12-20 23:59:00	Homework 3	Homework
11	1	2024-12-20 23:59:00	Extra Credit Project	Homework
NULL	NULL	NULL	NULL	NULL

  - Delete:

```
DELETE FROM Assignment WHERE AssignmentID = 9;

SELECT * FROM Assignment;
```

100% 47:1

Result Grid | Filter Rows: Search | Edit: | Expo

AssignmentID	CourseID	DueDate	Description	AssignmentType
5	5	2024-11-30 23:59:00	Homework 2	Homework
6	6	2024-12-05 23:59:00	Final Exam	Exam
7	7	2024-12-10 23:59:00	Project Presentation	Exam
8	8	2024-12-15 23:59:00	Quiz 2	Quiz
11	1	2024-12-20 23:59:00	Extra Credit Project	Homework
NULL	NULL	NULL	NULL	NULL

  - Update:

```
1  UPDATE Assignment SET DueDate = '2024-12-25 23:59:00' WHERE AssignmentID = 2;
2
3  SELECT * FROM Assignment;
```

100% 1:2

Result Grid Filter Rows: Search Edit: Export/Import:

AssignmentID	CourseID	DueDate	Description	AssignmentType
1	1	2024-11-15 23:59:00	Midterm Exam	Exam
2	2	2024-12-25 23:59:00	Final Exam	Exam
4	4	2024-11-25 23:59:00	Quiz 1	Quiz

- Exam Table:
  - Query:

```
1  SELECT * FROM Exam;
2
```

100% 1:2

Result Grid Filter Rows:

ExamID	AssignmentID
1	1
2	2
3	6
4	7
5	10
NULL	NULL

  - Insert:

```
1  INSERT INTO Exam (ExamID, AssignmentID) VALUES (6, 1);
2
3  SELECT * FROM Exam;
```

100% 50:1

Result Grid Filter Rows: Search Edit: Expo

ExamID	AssignmentID
1	1
6	1
2	2
3	6
4	7
5	10
NULL	NULL

  - Delete:

```
1  DELETE FROM Exam WHERE ExamID = 3;
2
3  SELECT * FROM Exam;
```

100% 1:2

Result Grid Filter Rows: Search

ExamID	AssignmentID
1	1
6	1
2	2
4	7
5	10
NULL	NULL

  - Update:

```
UPDATE Exam SET AssignmentID = 9 WHERE ExamID = 4;

SELECT * FROM Exam;
```

100% 33:1

Result Grid Filter Rows: Search Edit:

ExamID	AssignmentID
1	1
6	1
2	2
4	9

- Quiz Table:
  - Query:

```
SELECT * FROM Quiz;
```

100% 1:2

Result Grid Filter Rows:

QuizID	AssignmentID
1	4
2	8
NULL	NULL

  - Insert:

```
INSERT INTO Quiz (QuizID, AssignmentID) VALUES (4, 4);

SELECT * FROM Quiz;
```

100% 1:2

Result Grid Filter Rows: Search Edit: Expor

QuizID	AssignmentID
1	4
4	4
2	8
NULL	NULL

  - Delete:

```
DELETE FROM Quiz WHERE QuizID = 1;

SELECT * FROM Quiz;
```

100% 1:2

Result Grid Filter Rows: Search

QuizID	AssignmentID
4	4
2	8
NULL	NULL

  - Update:

```
UPDATE Quiz SET AssignmentID = 5 WHERE QuizID = 2;

SELECT * FROM Quiz;
```

00% 1:2

Result Grid Filter Rows: Search Edit:

QuizID	AssignmentID
4	4
2	5
NULL	NULL

- Homework Table:
  - Query:

```
SELECT * FROM Homework;
```

100% 1:2

Result Grid | Filter Rows: | Search

HomeworkID	AssignmentID
1	3
2	5
3	9
NULL	NULL

- Insert:

```
INSERT INTO Homework (HomeworkID, AssignmentID) VALUES (4, 8);
SELECT * FROM Homework;
```

100% 1:2

Result Grid | Filter Rows: | Search | Edit: | Export/Import:

HomeworkID	AssignmentID
1	3
2	5
4	8
3	9

- Delete:

```
DELETE FROM Homework WHERE HomeworkID = 2;
SELECT * FROM Homework;
```

100% 1:2

Result Grid | Filter Rows: | Search | Edit:

HomeworkID	AssignmentID
1	3
4	8
3	9
NULL	NULL

- Update:

```
UPDATE Homework SET AssignmentID = 7 WHERE HomeworkID = 1;
SELECT * FROM Homework;
```

100% 1:2

Result Grid | Filter Rows: | Search | Edit: | Export/Impo

HomeworkID	AssignmentID
1	7
4	8

- Enrollment Table:
  - Query:

```
SELECT * FROM Enrollment WHERE CourseID = 1;
```

100% 1:2

Result Grid | Filter Rows: | Search | Edit:

StudentID	CourseID
1	1
NULL	NULL

  - Insert:

```
INSERT INTO Enrollment (StudentID, CourseID) VALUES (2, 2), (3, 3), (4, 2), (4, 4), (5, 5), (6, 6), (7, 7);

SELECT * FROM Enrollment;
```

100% 108:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

StudentID	CourseID
1	1
2	2
4	2
3	3
4	4
5	5
6	6
7	7

- Delete:

```
DELETE FROM Enrollment WHERE StudentID = 4 AND CourseID = 4;

SELECT * FROM Enrollment;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/Import:

StudentID	CourseID
1	1
2	2
4	2
3	3
5	5
6	6

- Update:

```
UPDATE Enrollment SET CourseID = 2 WHERE StudentID = 6;

SELECT * FROM Enrollment;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/I

StudentID	CourseID
1	1
2	2
4	2
6	2
3	3

- Finances Table:
  - Query:

```
SELECT * FROM Finances WHERE StudentID = 1;
```

100% 44:1

Result Grid | Filter Rows: Search | Edit:

FinancesID	StudentID	Amount	FinancesType	PaymentDeadline	Status
1	1	5000.00	Tuition	2024-12-10 00:00:00	1
NULL	NULL	NULL	NULL	NULL	NULL

  - Insert:

```
INSERT INTO Finances (FinancesID, StudentID, Amount, FinancesType, PaymentDeadline, Status)
VALUES (11, 11, 1000.00, 'Tuition', '2024-12-15', 1);

SELECT * FROM Finances;
```

100% 24:4

Result Grid | Filter Rows: Search | Edit: | Export/Import:

FinancesID	StudentID	Amount	FinancesType	PaymentDeadline	Status
1	1	5000.00	Tuition	2024-12-10 00:00:00	1
2	2	2500.00	Dorm	2024-11-30 00:00:00	0
3	3	200.00	Meal Plan	2024-12-15 00:00:00	1
4	4	1500.00	Tuition	2024-12-10 00:00:00	1
5	5	1000.00	Library Fee	2024-11-25 00:00:00	0
6	6	300.00	Meal Plan	2024-12-05 00:00:00	1
7	7	150.00	Club Dues	2024-12-20 00:00:00	1
8	8	75.00	Lab Fee	2024-11-28 00:00:00	0
9	9	500.00	Parking	2024-12-01 00:00:00	1
10	10	800.00	Gym Membership	2024-12-18 00:00:00	0
11	11	1000.00	Tuition	2024-12-15 00:00:00	1

  - Delete:

```
1  DELETE FROM Finances WHERE FinancesID = 5;
2
3  SELECT * FROM Finances;
```

100% 43:1

Result Grid Filter Rows: Search Edit: Exp

FinancesID	StudentID	Amount	FinancesType	PaymentDeadline	Status
1	1	5000.00	Tuition	2024-12-10 00:00:00	1
2	2	2500.00	Dorm	2024-11-30 00:00:00	0
3	3	200.00	Meal Plan	2024-12-15 00:00:00	1
4	4	1500.00	Tuition	2024-12-10 00:00:00	1
6	6	300.00	Meal Plan	2024-12-05 00:00:00	1

- Update:

```
1  UPDATE Finances SET Status = 1 WHERE FinancesID = 6;
2
3  SELECT * FROM Finances;
```

100% 1:2

Result Grid Filter Rows: Search Edit: Expo

FinancesID	StudentID	Amount	FinancesType	PaymentDeadline	Status
1	1	5000.00	Tuition	2024-12-10 00:00:00	1
2	2	2500.00	Dorm	2024-11-30 00:00:00	0
3	3	200.00	Meal Plan	2024-12-15 00:00:00	1
4	4	1500.00	Tuition	2024-12-10 00:00:00	1
6	6	300.00	Meal Plan	2024-12-05 00:00:00	1
7	7	150.00	Club Dues	2024-12-20 00:00:00	1

- Dining Table:
  - Query:

```
1  SELECT * FROM Dining WHERE MealPlanType = 'Unlimited';
2
```

100% 1:2

Result Grid Filter Rows: Search Edit: Export

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1	1200.00	14	Unlimited
4	4	1000.00	14	Unlimited
9	9	900.00	14	Unlimited
NULL	NULL	NULL	NULL	NULL

  - Insert:

```
1  INSERT INTO Dining (DiningID, StudentID, MealPlanBalance, SwipesPerWeek, MealPlanType)
2  VALUES (11, 1, 1300.00, 16, 'Unlimited');
3
4  SELECT * FROM Dining;
5
```

100% 22:4

Result Grid Filter Rows: Search Edit: Export/Import:

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1	1200.00	14	Unlimited
2	2	800.00	10	Standard
3	3	1500.00	21	Premium
4	4	1000.00	14	Unlimited
5	5	500.00	7	Basic
6	6	750.00	10	Standard
7	7	1300.00	21	Premium
8	8	600.00	10	Standard
9	9	900.00	14	Unlimited
10	10	400.00	7	Basic
11	1	1300.00	16	Unlimited
NULL	NULL	NULL	NULL	NULL

  - Delete:

```
DELETE FROM Dining WHERE DiningID = 4;

SELECT * FROM Dining;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1	1200.00	14	Unlimited
2	2	800.00	10	Standard
3	3	1500.00	21	Premium
5	5	500.00	7	Basic
6	6	750.00	10	Standard

- Update:

```
UPDATE Dining SET MealPlanBalance = 1300.00 WHERE StudentID = 2;

SELECT * FROM Dining;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/Import:

DiningID	StudentID	MealPlanBalance	SwipesPerWeek	MealPlanType
1	1	1200.00	14	Unlimited
2	2	1300.00	10	Standard
3	3	1500.00	21	Premium

- Housing Table:
  - Query:

```
SELECT * FROM Housing WHERE BuildingNo = 'Bldg 1';
```

100% 1:2

Result Grid | Filter Rows: Search | Edit:

HousingID	StudentID	BuildingNo	RoomNo
1	1	Bldg 1	Rm 101
2	2	Bldg 1	Rm 102
NULL	NULL	NULL	NULL

  - Insert:

```
INSERT INTO Housing (HousingID, StudentID, BuildingNo, RoomNo)
VALUES (11, 11, 'Bldg 6', 'Rm 601');

SELECT * FROM Housing;
```

100% 1:3

Result Grid | Filter Rows: Search | Edit: | Export/Import:

HousingID	StudentID	BuildingNo	RoomNo
1	1	Bldg 1	Rm 101
2	2	Bldg 1	Rm 102
3	3	Bldg 2	Rm 201
4	4	Bldg 2	Rm 202
5	5	Bldg 3	Rm 301
6	6	Bldg 3	Rm 302
7	7	Bldg 4	Rm 401
8	8	Bldg 4	Rm 402
9	9	Bldg 5	Rm 501
10	10	Bldg 5	Rm 502
11	11	Bldg 6	Rm 601
NULL	NULL	NULL	NULL

  - Delete:

```
1  DELETE FROM Housing WHERE HousingID = 3;
2
3  SELECT * FROM Housing;
4
5
```

100% 41:1

Result Grid Filter Rows: Search Edit:

HousingID	StudentID	BuildingNo	RoomNo
1	1	Bldg 1	Rm 101
2	2	Bldg 1	Rm 102
4	4	Bldg 2	Rm 202

- Update:

```
1  UPDATE Housing SET RoomNo = 'Rm 103' WHERE HousingID = 2;
2
3  SELECT * FROM Housing;
4
5
```

100% 1:2

Result Grid Filter Rows: Search Edit: Export/Impo

HousingID	StudentID	BuildingNo	RoomNo
1	1	Bldg 1	Rm 101
2	2	Bldg 1	Rm 103

- RoommateAgreement Table:
  - Query:

```
1  SELECT * FROM RoommateAgreement WHERE AgreementComplete = 1;
```

100% 61:1

Result Grid Filter Rows: Search Edit: Export/Import

HousingID	DateSigned	AgreementCompl...
1	2024-09-01	1
2	2024-09-02	1
4	2024-09-04	1
6	2024-09-06	1
8	2024-09-08	1
10	2024-09-10	1
NULL	NULL	NULL

  - Insert:

```
1  INSERT INTO RoommateAgreement (HousingID, DateSigned, AgreementComplete)
2  VALUES (11, '2025-03-11', 1);
3
4  SELECT * FROM RoommateAgreement;
```

100% 1:3

Result Grid Filter Rows: Search Edit: Export/Import:

HousingID	DateSigned	AgreementCompl...
1	2024-09-01	1
2	2024-09-02	1
4	2024-09-04	1
5	2024-09-05	0
6	2024-09-06	1
7	2024-09-07	0
8	2024-09-08	1
9	2024-09-09	0
10	2024-09-10	1
11	2025-03-11	1
NULL	NULL	NULL

  - Delete:

```
DELETE FROM RoommateAgreement WHERE HousingID = 5;

SELECT * FROM RoommateAgreement;
```

100% 51:1

Result Grid | Filter Rows: Search | Edit:

HousingID	DateSigned	AgreementCompl...
1	2024-09-01	1
2	2024-09-02	1
4	2024-09-04	1
6	2024-09-06	1
7	2024-09-07	0

- Update:

```
UPDATE RoommateAgreement SET AgreementComplete = 1 WHERE HousingID = 7;

SELECT * FROM RoommateAgreement;
```

100% 1:2

Result Grid | Filter Rows: Search | Edit: | Export/Import:

HousingID	DateSigned	AgreementCompl...
1	2024-09-01	1
2	2024-09-02	1
4	2024-09-04	1
6	2024-09-06	1
7	2024-09-07	1

3.5. Create View:

Student Financial Summary

This view aggregates students' financial information to provide a comprehensive summary of their financial obligations. It includes details such as the type of financial record (e.g., tuition, housing, or books), the amount due, the payment deadline, and whether the payment is pending or completed.

Purpose: This view simplifies financial management by allowing administrators, financial officers, or advisors to quickly identify students with overdue payments or upcoming deadlines.

Use Case: A financial aid officer can use this view to generate a report of students with pending tuition payments and contact them for reminders.

```sql
CREATE VIEW StudentFinancialSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    f.FinancesType,
    f.Amount,
    f.PaymentDeadline,
    CASE
        WHEN f.Status = 0 THEN 'Pending'
        WHEN f.Status = 1 THEN 'Paid'
        ELSE 'Unknown'
    END AS PaymentStatus
FROM
    Student s
JOIN
    Finances f
ON
    s.StudentID = f.StudentID;
```

Event Attendance

This view links students to the events they attended, displaying details like event names, dates, and the names of attendees. It centralizes attendance records, which are typically scattered across multiple systems or manual lists.

Purpose: To provide an organized way of tracking student participation in extracurricular activities and events.

Use Case: Event coordinators can analyze attendance trends, identify active participants, and evaluate the success of campus events. Additionally, students can use this data to document extracurricular involvement for resumes or applications.

```sql
CREATE VIEW EventAttendance AS
SELECT
    e.EventID,
    e.EventName,
    e.Date,
    s.StudentID,
    s.StudentName
FROM
    EventsAttended ea
JOIN
    Student s
ON
    ea.StudentID = s.StudentID
JOIN
    Event e
ON
    ea.EventID = e.EventID;
```

Dining Plan Summary

This view compiles information about students' dining plans, showing their meal plan type, remaining balance, and weekly swipe allocations. It centralizes data that would otherwise need to be retrieved from multiple sources.

Purpose: To help dining services manage meal plan usage and monitor remaining balances.

Use Case: Dining administrators can use this view to predict demand for meal swipes and notify students about low balances. Students can also check their plan status to avoid running out of meal swipes unexpectedly.

```
CREATE VIEW DiningPlanSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    d.MealPlanType,
    d.MealPlanBalance,
    d.SwipesPerWeek
FROM
    Student s
JOIN
    Dining d
ON
    s.StudentID = d.StudentID;
```

Course Enrollment

This view lists all students enrolled in each course, along with course names and enrollment dates. It connects student records to course data, creating a clear relationship between the two.

Purpose: To provide a comprehensive enrollment overview for professors, teaching assistants, and academic advisors.

Use Case: Professors can use this view to monitor class sizes and track student participation in courses. Academic advisors can use it to ensure students are enrolled in the appropriate courses for their degree progress.

```sql
CREATE VIEW CourseEnrollment AS
SELECT
    c.CourseID,
    c.CourseName,
    s.StudentID,
    s.StudentName,
    e.EnrollmentDate
FROM
    Enrollment e
JOIN
    Student s
ON
    e.StudentID = s.StudentID
JOIN
    Course c
ON
    e.CourseID = c.CourseID;
```

Housing Summary

This view provides detailed housing information for each student, including housing type (e.g., dormitory, off-campus), costs, and roommate agreements. By integrating housing data, it allows for easy management of student accommodations.

Purpose: To assist housing staff in tracking room assignments, roommate pairings, and associated costs for each student.

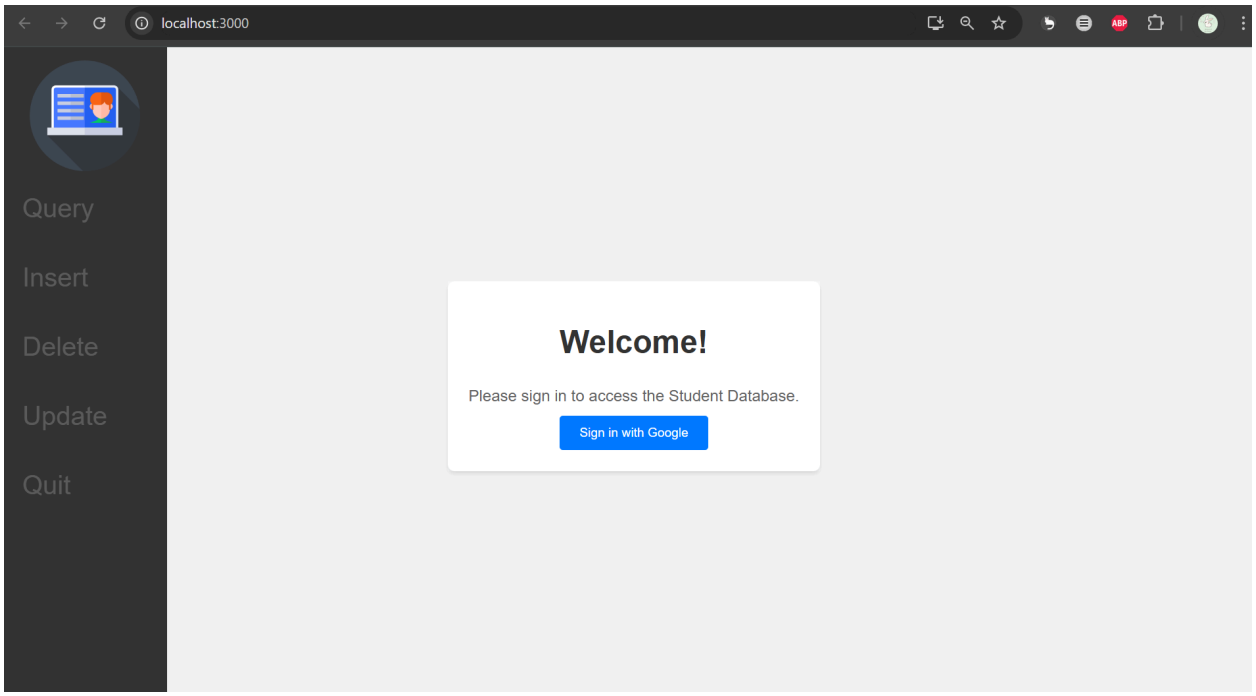
Use Case: Housing administrators can quickly resolve inquiries about housing costs, roommate conflicts, or room assignments. Students can use this information to verify their housing agreements or costs.

```sql
CREATE VIEW HousingSummary AS
SELECT
    s.StudentID,
    s.StudentName,
    h.HousingType,
    h.Cost,
    r.RoommateName
FROM
    Student s
JOIN
    Housing h
ON
    s.StudentID = h.StudentID
LEFT JOIN
    RoommateAgreement r
ON
    h.HousingID = r.HousingID;
```

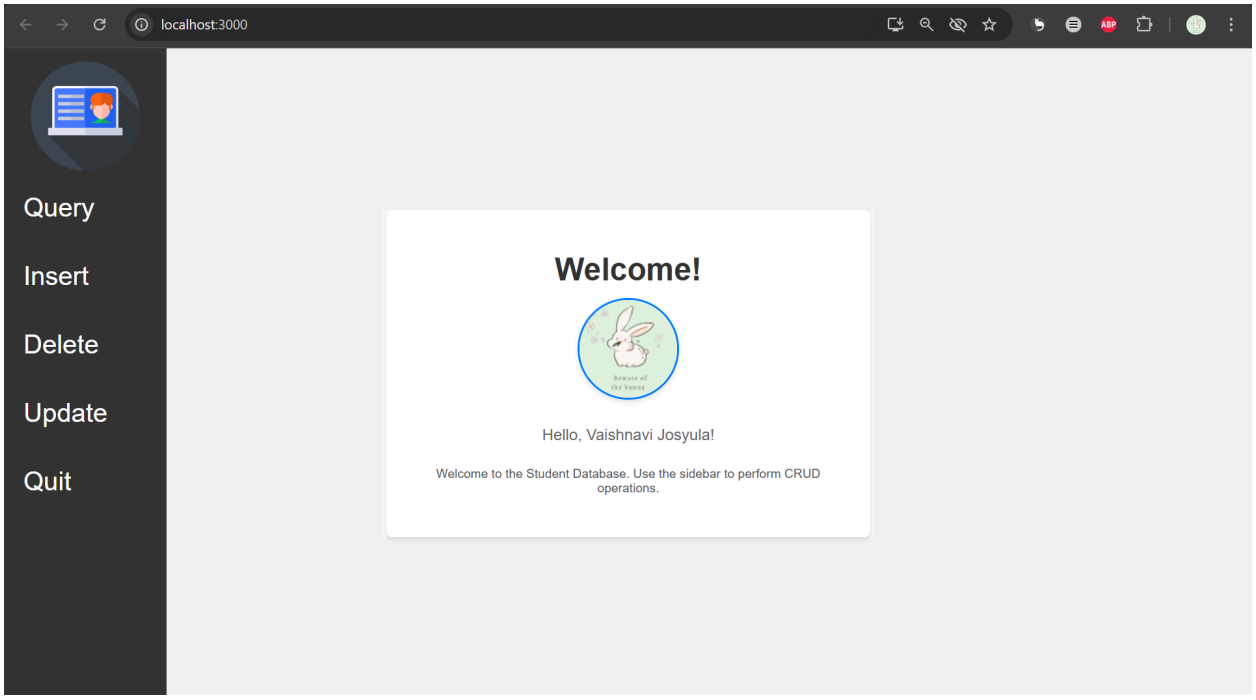
4. Front End User Interface:

We used React JS for our front end with Axios to make HTTP requests. We also used Firebase for Google Authentication.

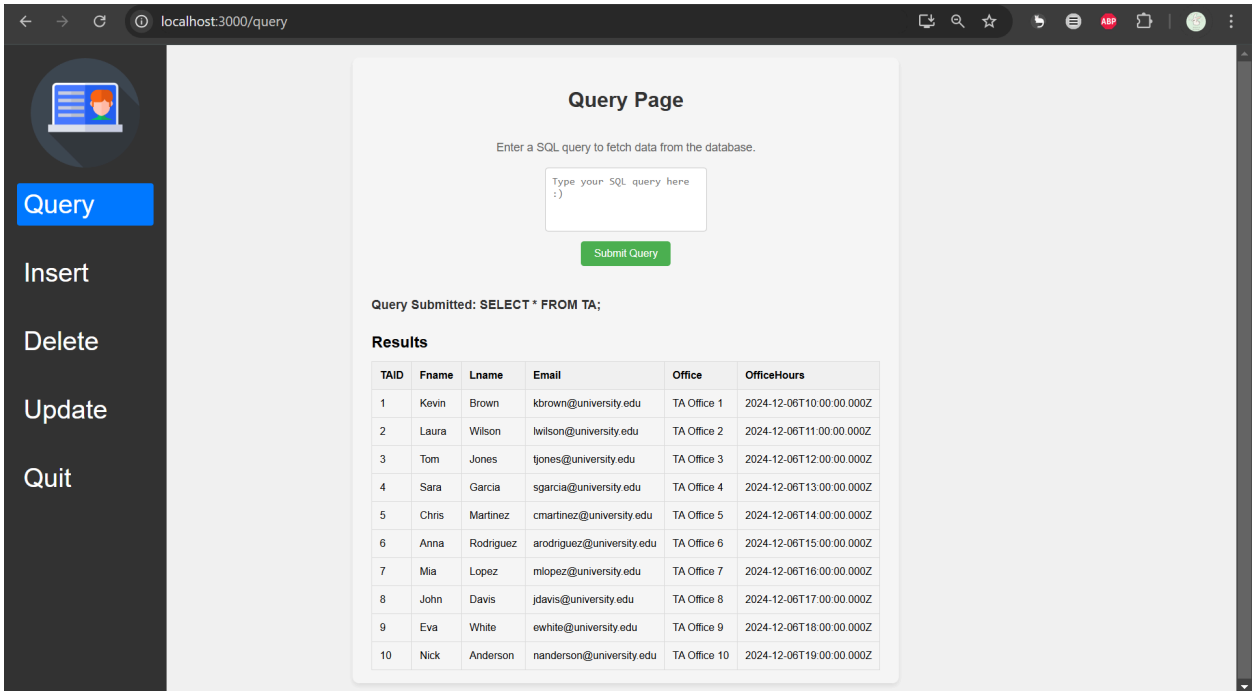
Login Page:

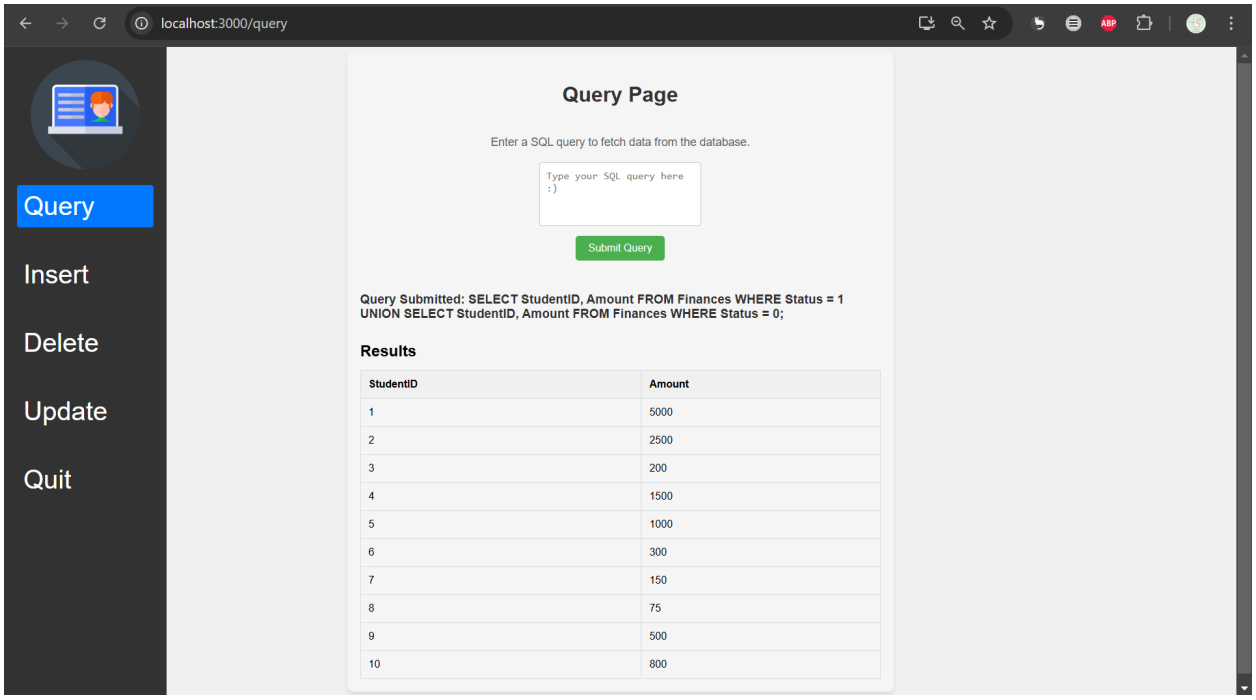


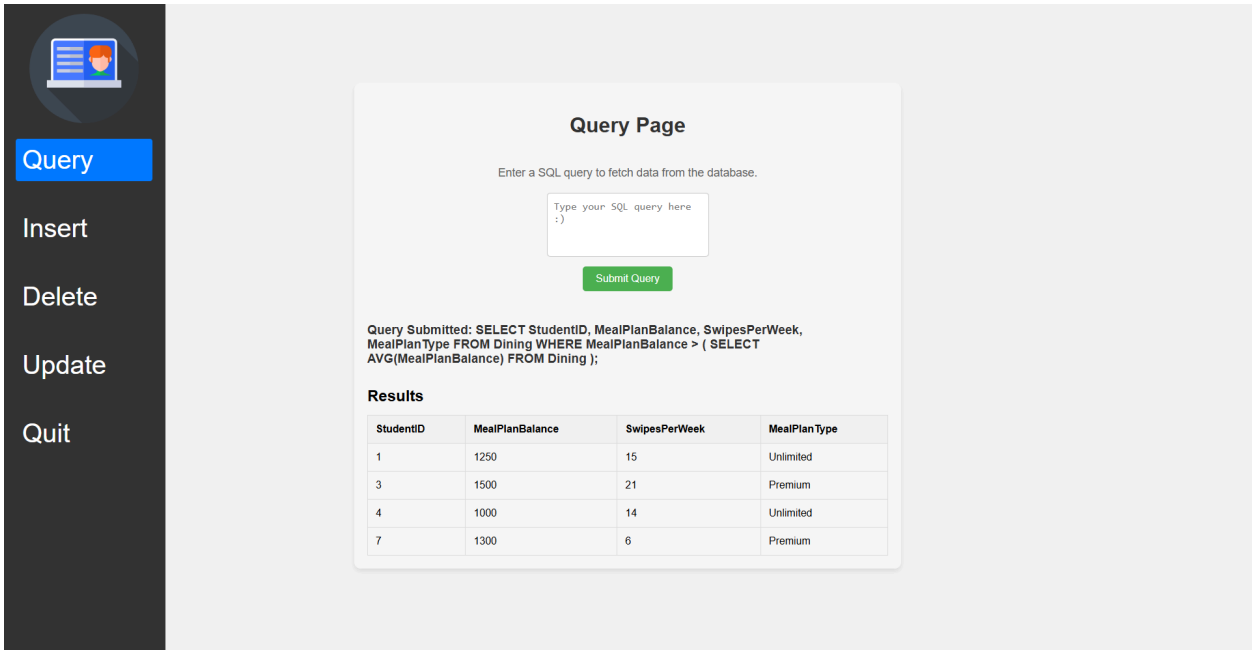
Home Page:

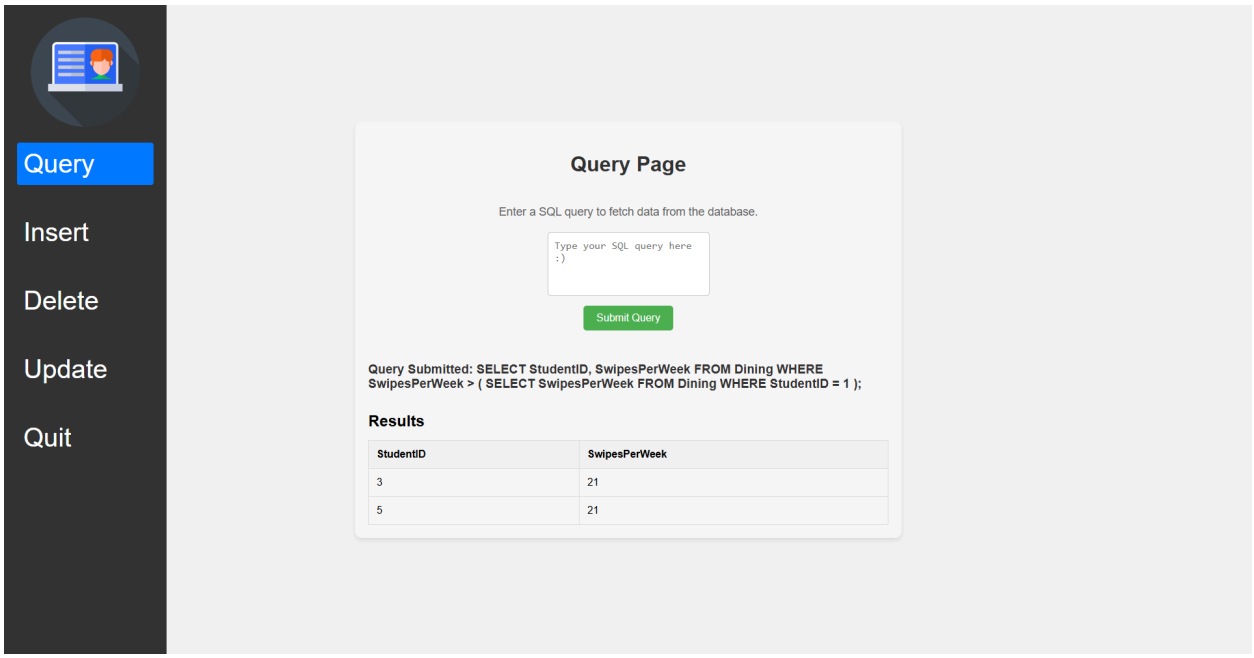


1. Query

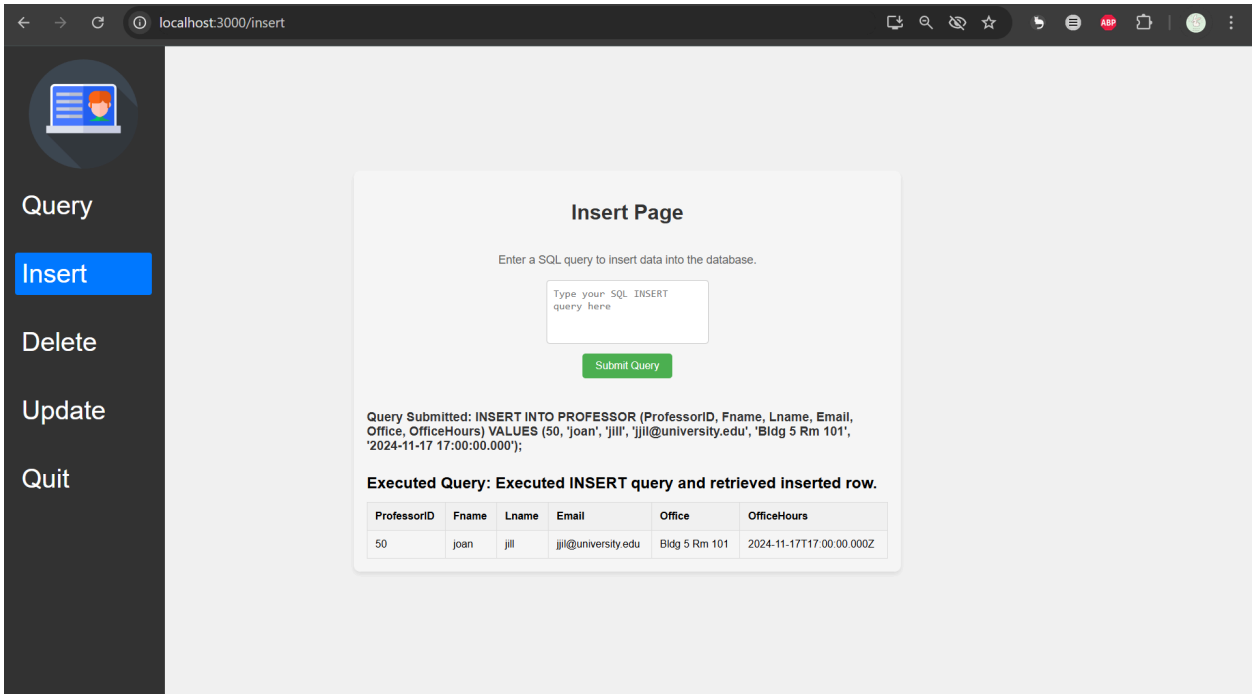

localhost:3000/query
Query
Insert
Delete
Update
Quit
Query Page
Enter a SQL query to fetch data from the database.
Type your SQL query here :)
Submit Query
Query Submitted: SELECT * FROM TA;
Results
TAID | Fname | Lname | Email | Office | OfficeHours
1 | Kevin | Brown | kbrown@university.edu | TA Office 1 | 2024-12-06T10:00:00.000Z
2 | Laura | Wilson | lwilson@university.edu | TA Office 2 | 2024-12-06T11:00:00.000Z
3 | Tom | Jones | tjones@university.edu | TA Office 3 | 2024-12-06T12:00:00.000Z
4 | Sara | Garcia | sgarcia@university.edu | TA Office 4 | 2024-12-06T13:00:00.000Z
5 | Chris | Martinez | cmartinez@university.edu | TA Office 5 | 2024-12-06T14:00:00.000Z
6 | Anna | Rodriguez | arodriguez@university.edu | TA Office 6 | 2024-12-06T15:00:00.000Z
7 | Mia | Lopez | mlopez@university.edu | TA Office 7 | 2024-12-06T16:00:00.000Z
8 | John | Davis | jdavis@university.edu | TA Office 8 | 2024-12-06T17:00:00.000Z
9 | Eva | White | ewhite@university.edu | TA Office 9 | 2024-12-06T18:00:00.000Z
10 | Nick | Anderson | nanderson@university.edu | TA Office 10 | 2024-12-06T19:00:00.000Z


localhost:3000/query
Query
Insert
Delete
Update
Quit
Query Page
Enter a SQL query to fetch data from the database.
Type your SQL query here :)
Submit Query
Query Submitted: SELECT StudentID, Amount FROM Finances WHERE Status = 1 UNION SELECT StudentID, Amount FROM Finances WHERE Status = 0;
Results
StudentID | Amount
1 | 5000
2 | 2500
3 | 200
4 | 1500
5 | 1000
6 | 300
7 | 150
8 | 75
9 | 500
10 | 800

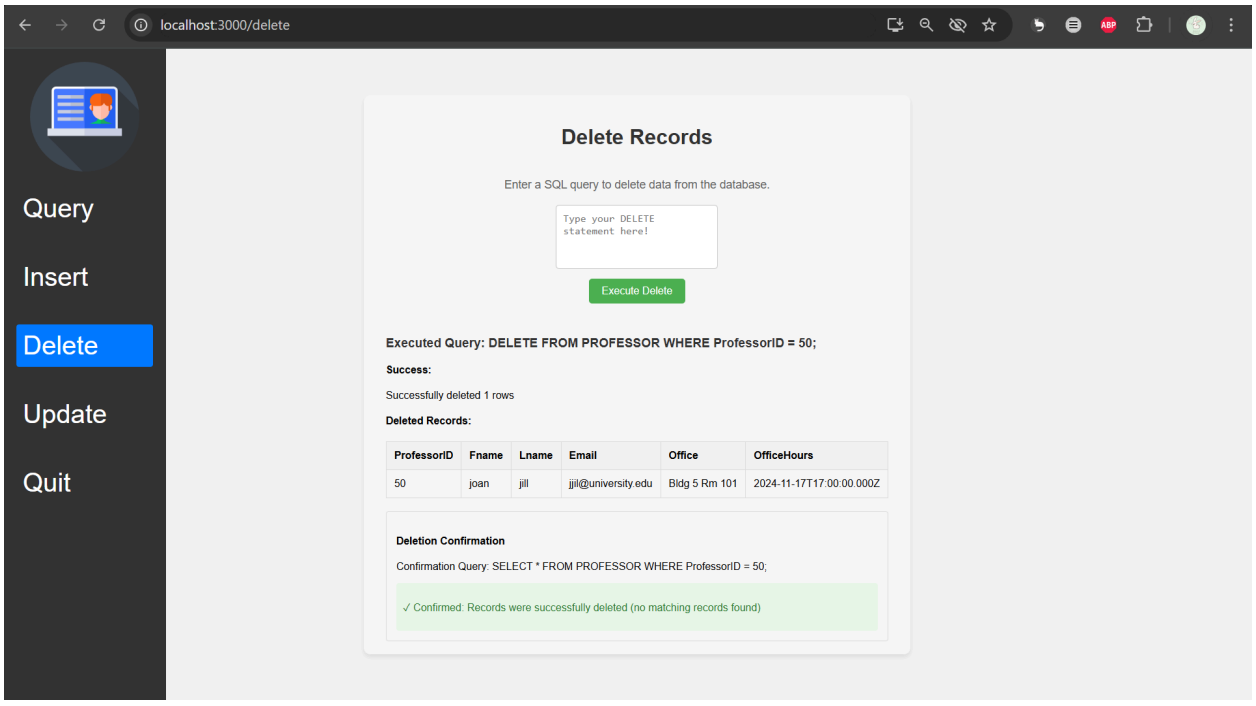

Query
Insert
Delete
Update
Quit
Query Page
Enter a SQL query to fetch data from the database.
Type your SQL query here :)
Submit Query
Query Submitted: SELECT StudentID, MealPlanBalance, SwipesPerWeek, MealPlanType FROM Dining WHERE MealPlanBalance > (SELECT AVG(MealPlanBalance) FROM Dining);
Results
StudentID | MealPlanBalance | SwipesPerWeek | MealPlanType
1 | 1250 | 15 | Unlimited
3 | 1500 | 21 | Premium
4 | 1000 | 14 | Unlimited
7 | 1300 | 6 | Premium


Query
Insert
Delete
Update
Quit
Query Page
Enter a SQL query to fetch data from the database.
Type your SQL query here :)
Submit Query
Query Submitted: SELECT StudentID, SwipesPerWeek FROM Dining WHERE SwipesPerWeek > (SELECT SwipesPerWeek FROM Dining WHERE StudentID = 1);
Results
StudentID | SwipesPerWeek
3 | 21
5 | 21

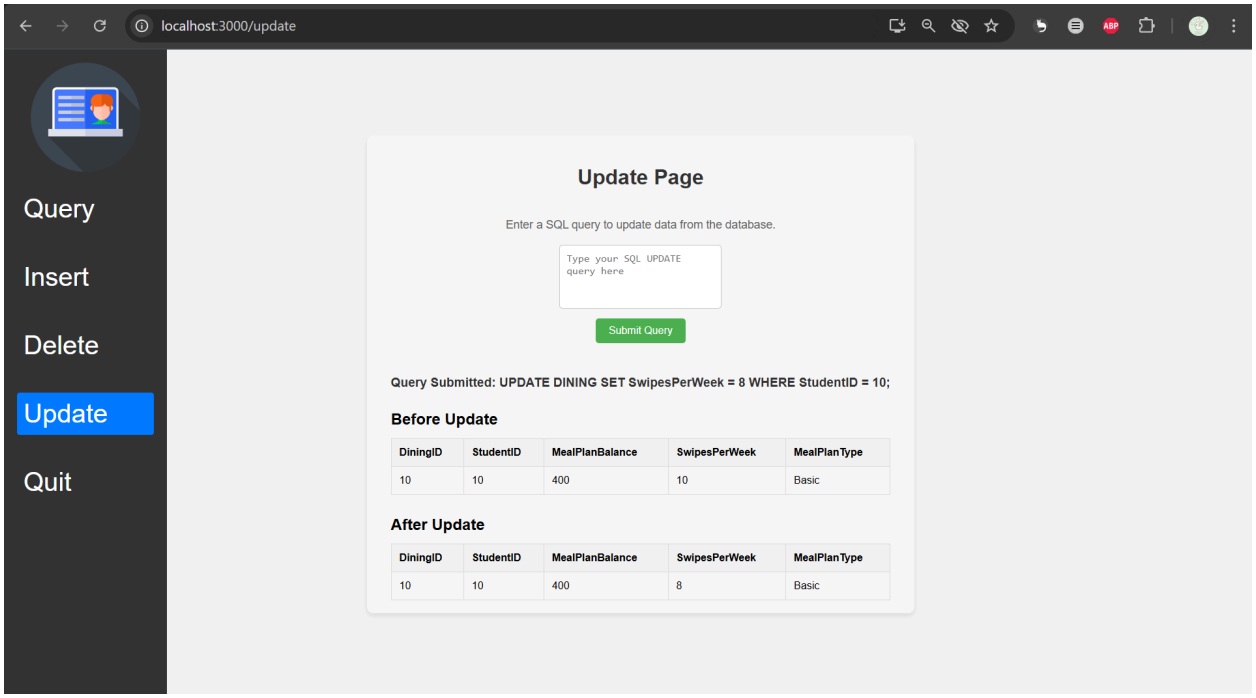
2. Insert



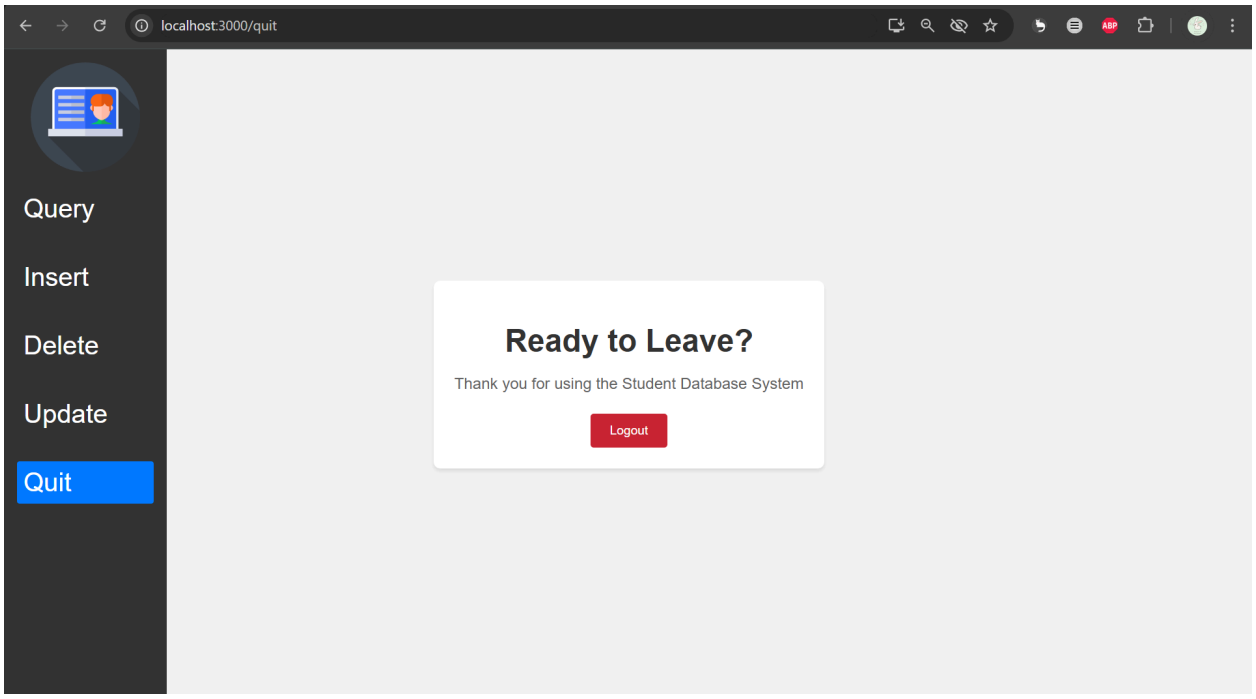
3. Delete



4. Update


localhost:3000/update
Query
Insert
Delete
Update
Quit
Update Page
Enter a SQL query to update data from the database.
Type your SQL UPDATE query here
Submit Query
Query Submitted: UPDATE DINING SET SwipesPerWeek = 8 WHERE StudentID = 10;
Before Update
DiningID StudentID MealPlanBalance SwipesPerWeek MealPlanType
10 10 400 10 Basic
After Update
DiningID StudentID MealPlanBalance SwipesPerWeek MealPlanType
10 10 400 8 Basic

5. Quit


localhost:3000/quit
Query
Insert
Delete
Update
Quit
Ready to Leave?
Thank you for using the Student Database System
Logout

5. Conclusion and Future Work:

Overall, we implemented a full-stack student record-keeping database system that aims to help college students manage various aspects of their campus life, such as academics, finances, housing, dining, extracurricular activities, and more. The system was designed to centralize these areas into a single, user-friendly platform, reducing the complexity students face when navigating multiple systems. Some challenges we faced came when we needed to normalize the database. Our database implementation from Deliverable 1 was not in 3NF resulting in drastic changes to the database design. These changes were reflected in the new ER Diagram and database schema. It was also a little challenging to connect our frontend with the backend due to TCP/IP errors, although we were able to eventually resolve that issue.

To further streamline the process of keeping track of a college student's life, we could look into creating a mobile app that complements the web platform. A mobile app would provide better accessibility for students who are always on the move, allowing them to check schedules, event updates, and important notifications directly from their phones. Additionally, making the website more collaborative among students on the same campus could create a stronger sense of community and increase student engagement. This could take shape through shared event calendars, course reviews, and more. This could promote more interaction between students, potentially leading to greater involvement in extracurricular activities, campus events, as well as other opportunities. Lastly, it would be highly beneficial to expand the current platform to integrate with existing systems like the library management system or student health services. Integration with these systems would centralize all relevant student information into one platform, reducing the need for students to navigate multiple websites or apps. For example, students could directly check out library books, view their health records, or schedule doctor appointments without leaving the platform. This all-in-one solution would save time, increase convenience, and make managing a student's life on campus much easier.

6. References:

1. A. Eludire, "The Design and Implementation of Student Academic Record Management System," International Journal of Computing and ICT Research, vol. 5, no. 2, pp. 20-25, 2011.

2. A. S. Bidyarthy, A. Kumar, "Student Database Management System," International Journal of Emerging Technologies, vol. 3, no. 2, pp. 28-35, 2012.

3. A. Tamboli, "Institute Administration Automation and Student Database Management System," Journal of Automation and Control Engineering, vol. 5, no. 4, pp. 210-216, 2017.

4. R. Elmasri, S. Navathe, "Relational Database Design by ER- and EER-to-Relational Mapping" in Fundamentals of Database Systems, 7th ed., USA: Pearson, 2021, ch. 9, pp. 289 - 306.