

Database Fundamental Project

Dany

Table of Content

Table of Content	1
Introduction	1
Task One - Database Design	2
Task Two – Create & Populate the Database	17
Task Three – Query The Dataset	24

Introduction

This project is intended to assess understanding of database design and implementation including Conceptual data modeling, logical & physical data modeling, Data Definition Language, and Structured Query language.

Task One - Database Design

1. Identify Relevant Entities with Attributes, Types, and Primary Keys

Entities:

- Departments
 - **Attributes:** department_id (PK), name, school, phone_number.
 - **Entity Type:** Strong.
 - **Primary Key (PK):** department_id.
 - Courses
 - **Attributes:** course_id (PK), title, academic_level, department_id (FK), delivery_method, semesters, participants.
 - **Entity Type:** Strong.
 - **Primary Key (PK):** course_id.
 - Students
 - **Attributes:** student_id (PK), first_name, last_name, date_of_birth, address, email, course_id (FK), outstanding_fees_due, year, status.
 - **Entity Type:** Strong.
 - **Primary Key (PK):** student_id.
 - Results
 - **Attributes:** result_id (PK), student_id (FK), grade.
 - **Entity Type:** Strong.
 - **Primary Key (PK):** result_id.
 - Opening_Times
 - **Attributes:** time_id (PK), department_id (FK), day, opening_time, closing_time.
 - **Entity Type:** Weak (dependent on Departments).
 - **Primary Key (PK):** time_id.
 - Staff
 - **Attributes:** staff_id (PK), name, job_title, department_id (FK), years_at_university, days_off.
 - **Entity Type:** Strong.
 - **Primary Key (PK):** staff_id.
-

2. Explain Why Each Relation is Needed

Entities:

- Departments
 - **Purpose:** This entity represents the various academic and administrative units of the university.
 - **Key Attributes:**
 - **department_id:** Unique identifier for each department.
 - **name:** Name of the department.
 - **school:** The larger organizational unit, such as a college or faculty.
 - **phone_number:** University or Department phone number.
 - **Importance:** Departments serve as the organizing body for courses, and associating courses with departments enables the university to track academic programs by organizational structure.
- Courses
 - **Purpose:** This entity stores details of all courses offered by the university, including their academic levels and organizational affiliation.
 - **Key Attributes:**
 - **course_id:** Unique identifier for each course.
 - **title:** The name of the course.
 - **academic_level:** Represents the level of difficulty.
 - **department_id (FK):** Associates the course with the department that offers it.
 - **delivery_method:** Indicates how the course is delivered.
 - **semesters:** Tracks the duration of the course in semesters.
 - **participants:** Participants on the course.
 - **Importance:** This table helps manage course offerings, associate students with courses, and allows analysis of courses by department or academic level.
- Students
 - **Purpose:** This entity tracks all students at the university, both personal and academic details.
 - **Key Attributes:**
 - **student_id:** Unique identifier for each student.
 - **first_name** and **last_name:** Essential for identifying and displaying student information.
 - **date_of_birth:** Used for age-based queries or requirements.
 - **address** and **email:** Necessary for communication and contact.
 - **course_id (FK):** Links each student to their enrolled course.
 - **outstanding_fees_due:** How long have not been paid.

- **year**: Tracks the academic year of the student.
 - **status**: Indicates whether the student is active, graduated, or inactive.
 - **Importance**: Without this table, the university cannot manage student information or associate students with their courses, grades, or statuses.
- **Results**
 - **Purpose**: This entity records the grades or results achieved by students in their courses.
 - **Key Attributes**:
 - **result_id**: Unique identifier for each result entry.
 - **student_id** (FK): Links the result to the student who achieved it.
 - **grade**: Stores the actual grade earned by the student.
 - **Importance**: This table is crucial for tracking academic performance and generating reports for both students and the university.
- **Opening_Times**
 - **Purpose**: Tracks when departments or university facilities are open for students, staff, and visitors.
 - **Key Attributes**:
 - **time_id**: Unique identifier for each time entry.
 - **department_id** (FK): Links the opening time to a specific department.
 - **day**: Indicates the day of the week.
 - **opening_time** and **closing_time**: Specify the department's hours of operation.
 - **Importance**: This table supports operational management by defining when facilities or services are available. It is especially useful for scheduling, planning, and communicating availability.
- **Staff**
 - **Purpose**: Stores details of the university's employees, including faculty and administrative staff.
 - **Key Attributes**:
 - **staff_id**: Unique identifier for each staff member.
 - **name**: Staff member's name.
 - **job_title**: Their role at the university.
 - **department_id** (FK): Links staff to their respective department.
 - **years_at_university**: Tracks experience or seniority within the university.
 - **days_off**: Used to track absences or leaves.
 - **Importance**: This table is essential for HR and administrative purposes, allowing the university to manage its workforce, track departmental roles, and plan workloads.

3. Identify Relationships, Cardinalities, and Participation Constraints

Relationships:

- Students → Courses:
 - **Relationship:** Enrolls In.
 - **Cardinality:** Many-to-One (a student can enroll in one course; a course can have many students).
 - **Participation:** Mandatory for Students (a student must be enrolled in a course), optional for Courses (a course may not have any students).
- Courses → Departments:
 - **Relationship:** Belongs To.
 - **Cardinality:** Many-to-One (a course belongs to one department; a department can have many courses).
 - **Participation:** Mandatory for Courses, optional for Departments.
- Results → Students:
 - **Relationship:** Grades.
 - **Cardinality:** Many-to-One (a student can have many grades, student can have result for each year)
 - **Participation:** Mandatory for Results. (every grade must be associated with a student).
- Opening_Times → Departments:
 - **Relationship:** Has Opening Times.
 - **Cardinality:** Many-to-One (a department can have multiple opening time entries for different days; each entry belongs to one department).
 - **Participation:** Mandatory for Opening_Times.
- Staff → Departments:
 - **Relationship:** Works In.
 - **Cardinality:** Many-to-One (a staff member works in one department; a department can have many staff members).
 - **Participation:** Mandatory for Staff.

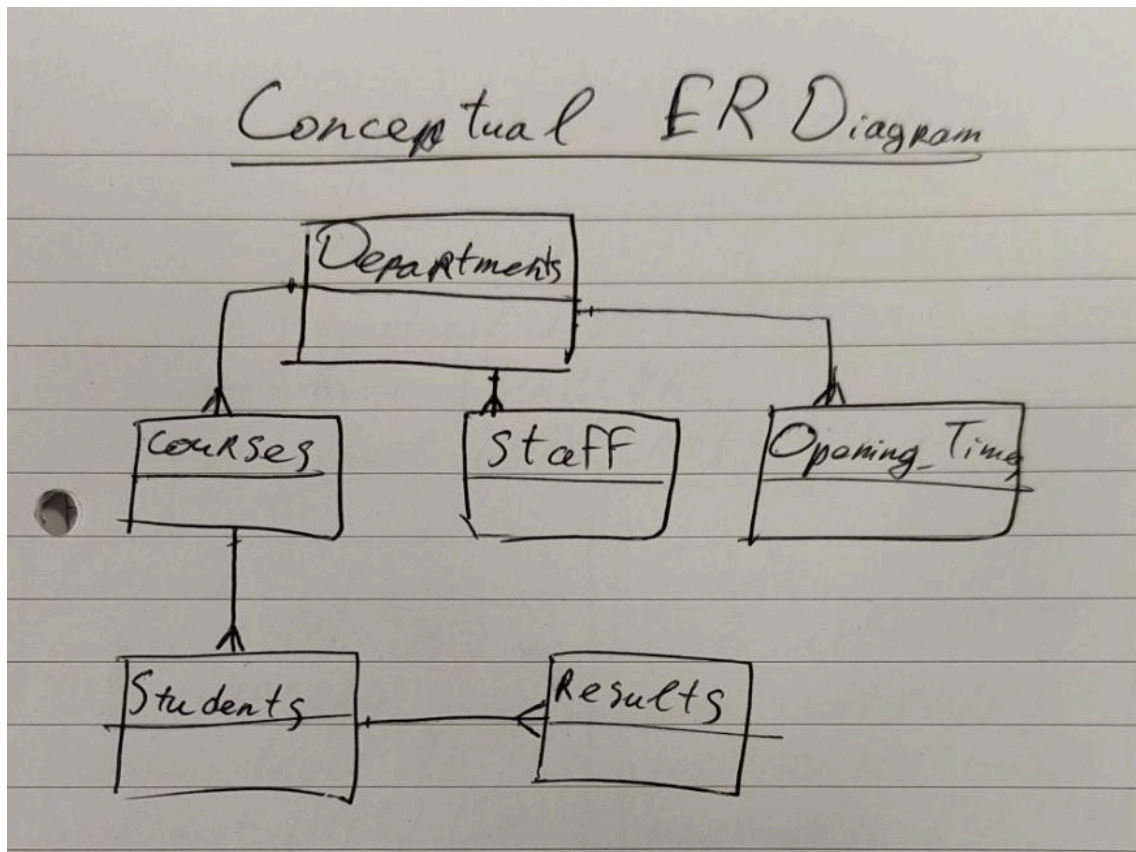
4. Conceptual ER Diagram

Description of the diagram

- Entities:
 - **Departments:** Represents university departments like the School of Computing or School of Business.
 - **Courses:** Represents courses offered by the university.

- **Students:** Tracks student information.
- **Results:** Tracks student grades in their courses.
- **Opening_Times:** Stores department opening hours.
- **Staff:** Represents university employees.
- Relationships:
 - **Students** → **Courses** ("Enrolls In"): A student can enroll in one course, but a course can have many students. Many-to-One relationship.
 - **Courses** → **Departments** ("Belongs To"): A course belongs to one department, but a department can offer many courses. Many-to-One relationship.
 - **Results** → **Students** ("Grades"): A student can have many grades (one for each course), but each result is linked to a single student. Many-to-One relationship.
 - **Opening_Times** → **Departments** ("Has Opening Times"): Each department can have multiple opening times, but each opening time belongs to a department. Many-to-One relationship.
 - **Staff** → **Departments** ("Works In"): A staff member works in one department, but each department can have many staff members. Many-to-One relationship.
- Cardinalities:
 - For each relationship, indicate the **cardinality**: many-to-one, one-to-many, or many-to-many.
 - Indicate **participation constraints** (mandatory vs. optional). For instance, a student must enroll in a course (mandatory participation for students) but a course may not have any students enrolled initially (optional participation for courses).

Draw ER Diagram



5. Logical ER Diagram

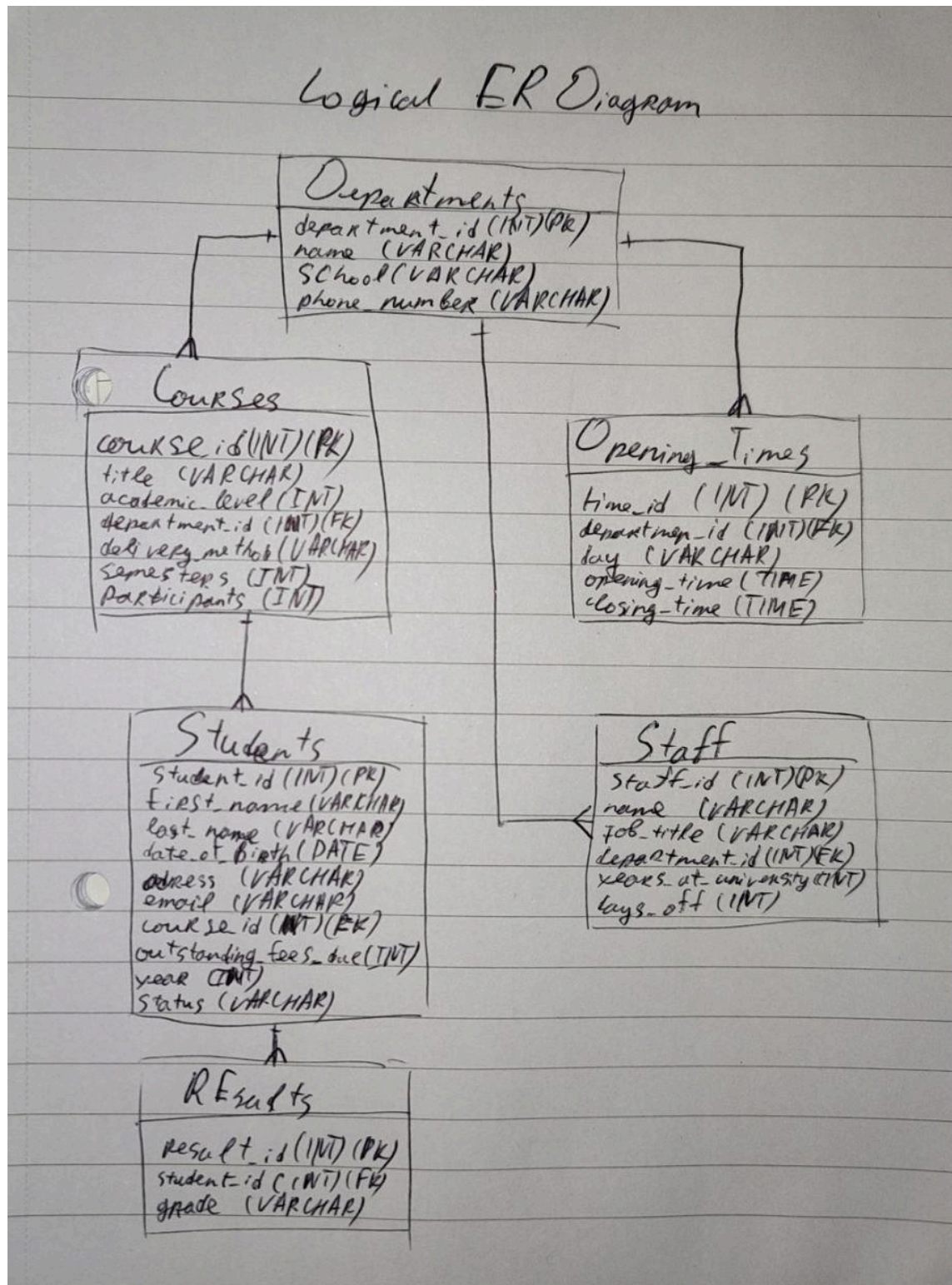
Description of the diagram

- Entities and Attributes in the Logical ERD
 - **Departments:**
 - **department_id** (PK) – Primary Key.
 - **name** (VARCHAR) – Name of the department.
 - **school** (VARCHAR) – School to which the department belongs.
 - **phone_number** (VARCHAR) – Phone number.
 - **Courses:**
 - **course_id** (PK) – Primary Key.
 - **title** (VARCHAR) – Name of the course.
 - **academic_level** (INT) – Academic level.
 - **department_id** (FK) – Foreign Key referencing **Departments(department_id)**.

- **delivery_method** (VARCHAR) – How the course is delivered.
 - **semesters** (INT) – Number of semesters the course runs.
 - **participants** (INT) – Participants on the course.
- **Students:**
 - **student_id** (PK) – Primary Key.
 - **first_name** (VARCHAR) – Student's first name.
 - **last_name** (VARCHAR) – Student's last name.
 - **date_of_birth** (DATE) – Date of birth.
 - **address** (VARCHAR) – Student's address.
 - **email** (VARCHAR) – Student's email address.
 - **course_id** (FK) – Foreign Key referencing **Courses(course_id)**.
 - **outstanding_fees_due** (INT) – How long have not been paid.
 - **year** (INT) – Year of study.
 - **status** (VARCHAR) – Current status.
- **Results:**
 - **result_id** (PK) – Primary Key.
 - **student_id** (FK) – Foreign Key referencing **Students(student_id)**.
 - **grade** (VARCHAR) – Grade awarded to the student.
- **Opening_Times:**
 - **time_id** (PK) – Primary Key.
 - **department_id** (FK) – Foreign Key referencing **Departments(department_id)**.
 - **day** (VARCHAR) – Day of the week the department is open.
 - **opening_time** (TIME) – Time the department opens.
 - **closing_time** (TIME) – Time the department closes.
- **Staff:**
 - **staff_id** (PK) – Primary Key.
 - **name** (VARCHAR) – Name of the staff member.
 - **job_title** (VARCHAR) – Job title.
 - **department_id** (FK) – Foreign Key referencing **Departments(department_id)**.
 - **years_at_university** (INT) – Number of years the staff member has worked at the university.
 - **days_off** (INT) – Number of days off taken by the staff member.
- Relationships in the Logical ERD
 - **Students** → **Courses** ("Enrolls In")
 - **Cardinality: Many-to-One** (Many students can enroll in a single course, but each student can only enroll in one course at a time.)

- **Foreign Key:** `course_id` in `Students` table referencing `Courses(course_id)`.
- **Courses → Departments** ("Belongs To")
 - **Cardinality: Many-to-One** (Each course belongs to one department, but a department can offer many courses.)
 - **Foreign Key:** `department_id` in `Courses` table referencing `Departments(department_id)`.
- **Results → Students** ("Grades")
 - **Cardinality: Many-to-One** (A student can have many results (grades), but each result corresponds to one student.)
 - **Foreign Key:** `student_id` in `Results` table referencing `Students(student_id)`.
- **Opening_Times → Departments** ("Has Opening Times")
 - **Cardinality: Many-to-One** (A department can have multiple opening times, but each opening time belongs to a specific department.)
 - **Foreign Key:** `department_id` in `Opening_Times` table referencing `Departments(department_id)`.
- **Staff → Departments** ("Works In")
 - **Cardinality: Many-to-One** (Each staff member works in a single department, but a department can have many staff members.)
 - **Foreign Key:** `department_id` in `Staff` table referencing `Departments(department_id)`.

Draw ER Diagram



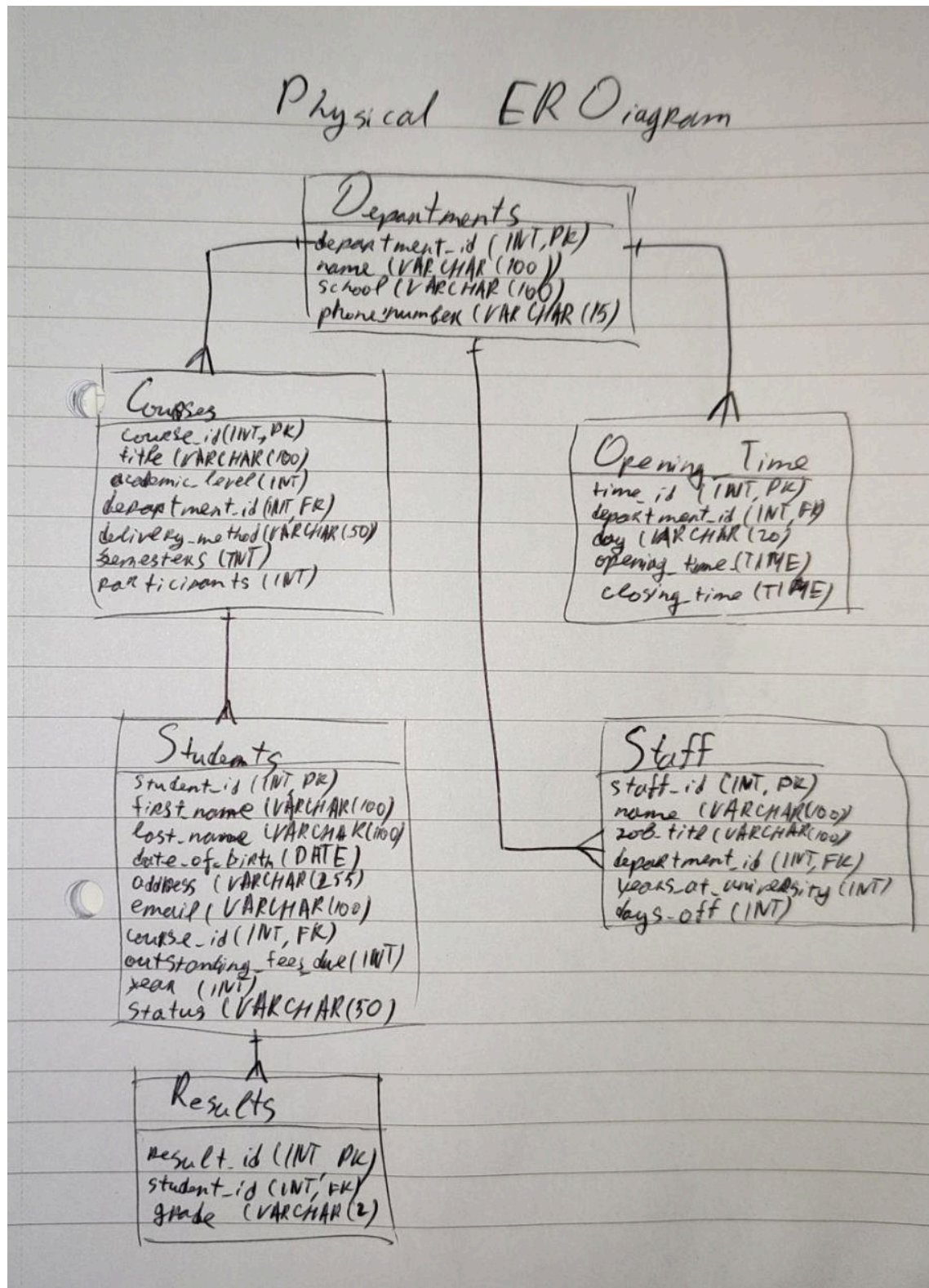
6. Physical ER Diagram

Description of the diagram

- Entities, Attributes, and Foreign Keys in the Physical ERD
 - **Departments:**
 - `department_id` (INT, PK) – **Primary Key**.
 - `name` (VARCHAR(100)) – Name of the department.
 - `school` (VARCHAR(100)) – School under which the department falls.
 - `phone_number` (VARCHAR(15)) – Phone number.
 - **Courses:**
 - `course_id` (INT, PK) – **Primary Key**.
 - `title` (VARCHAR(100)) – Name of the course.
 - `academic_level` (INT) – Academic level.
 - `department_id` (INT, FK) – **Foreign Key**: referencing `Departments(department_id)`.
 - `delivery_method` (VARCHAR(50)) – The delivery method.
 - `semesters` (INT) – Number of semesters the course runs.
 - `participants` (INT) – Participants on the course
 - **Students:**
 - `student_id` (INT, PK) – **Primary Key**.
 - `first_name` (VARCHAR(100)) – Student's first name.
 - `last_name` (VARCHAR(100)) – Student's last name.
 - `date_of_birth` (DATE) – Date of birth.
 - `address` (VARCHAR(255)) – Student's address.
 - `email` (VARCHAR(100)) – Student's email address.
 - `course_id` (INT, FK) – **Foreign Key**: referencing `Courses(course_id)`.
 - `outstanding_fees_due` (INT) – How long have not been paid.
 - `year` (INT) – Year of study.
 - `status` (VARCHAR(50)) – Status.
 - **Results:**
 - `result_id` (INT, PK) – **Primary Key**.
 - `student_id` (INT, FK) – **Foreign Key**: referencing `Students(student_id)`.
 - `grade` (VARCHAR(2)) – The grade the student received.
 - **Opening_Times:**
 - `time_id` (INT, PK) – **Primary Key**.

- `department_id` (INT, FK) – **Foreign Key**: referencing `Departments(department_id)`.
- `day` (VARCHAR(20)) – Day of the week.
- `opening_time` (TIME) – Opening time of the department.
- `closing_time` (TIME) – Closing time of the department.
- **Staff:**
 - `staff_id` (INT, PK) – **Primary Key**.
 - `name` (VARCHAR(100)) – Name of the staff member.
 - `job_title` (VARCHAR(100)) – Job title.
 - `department_id` (INT, FK) – **Foreign Key**: referencing `Departments(department_id)`.
 - `years_at_university` (INT) – Number of years the staff member has worked at the university.
 - `days_off` (INT) – Number of days off taken by the staff member.

Draw ER Diagram



7. Relational Model (Tables in 3NF)

Steps to Ensure 3NF

1. **1NF (First Normal Form):**
 - a. Ensure all attributes contain atomic (indivisible) values.
 - b. Ensure there are no repeating groups or arrays.
2. **2NF (Second Normal Form):**
 - a. Ensure the table is in 1NF.
 - b. Eliminate partial dependencies (non-key attributes depending on a part of the primary key).
3. **3NF (Third Normal Form):**
 - a. Ensure the table is in 2NF.
 - b. Eliminate transitive dependencies (non-key attributes depending on other non-key attributes).

Relational Model

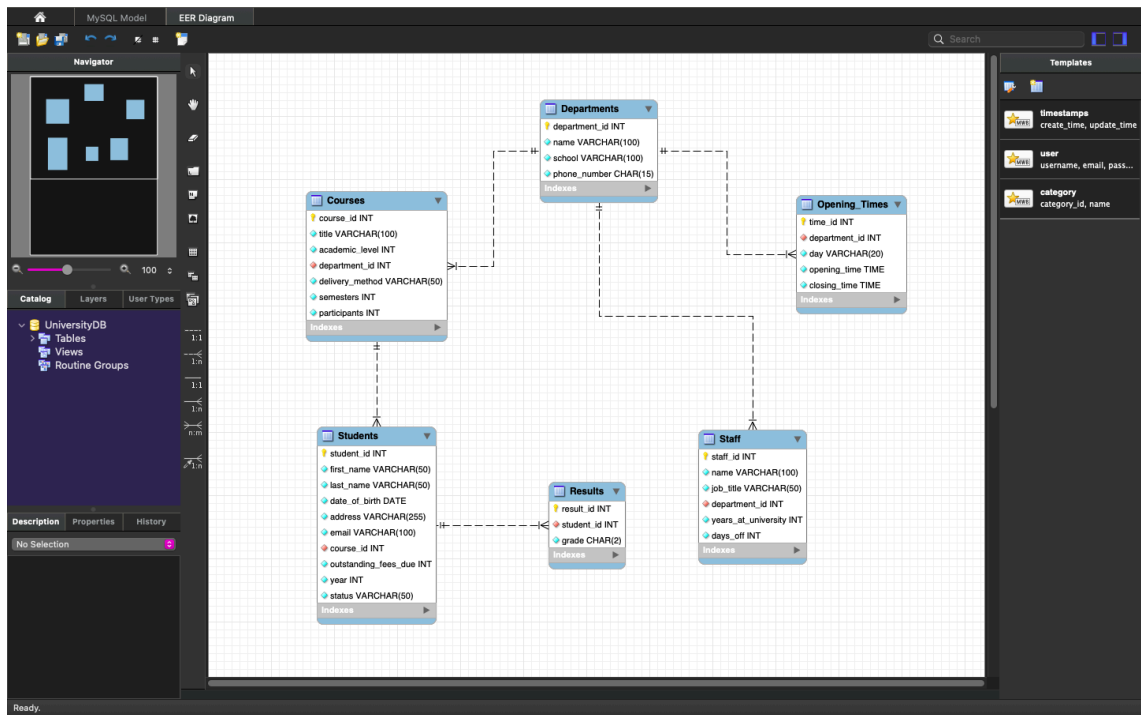
- Entities and Attributes
 - **Departments Table:**
 - `department_id` (PK)
 - `name`
 - `school`
 - `phone_number`
 - **Courses Table:**
 - `course_id` (PK)
 - `title`
 - `academic_level`
 - `department_id` (FK referencing `Departments(department_id)`)
 - `delivery_method`
 - `semesters`
 - `participants`
 - **Students Table:**
 - `student_id` (PK)
 - `first_name`
 - `last_name`
 - `date_of_birth`
 - `address`
 - `email` (unique constraint)
 - `course_id` (FK referencing `Courses(course_id)`)
 - `outstanding_fees_due`

- year
- status
- **Results Table:**
 - result_id (PK)
 - student_id (FK referencing Students(student_id))
 - grade
- **Opening_Times Table:**
 - time_id (PK)
 - department_id (FK referencing Departments(department_id))
 - day
 - opening_time
 - closing_time
- **Staff Table:**
 - staff_id (PK)
 - name
 - job_title
 - department_id (FK referencing Departments(department_id))
 - years_at_university
 - days_off

Relational Model Representation

Table	Attributes
Students	student_id (PK), first_name, last_name, date_of_birth, address, email, course_id (FK), year, status
Courses	course_id (PK), title, academic_level, department_id (FK), delivery_method, semesters
Departments	department_id (PK), name, school
Results	result_id (PK), student_id (FK), grade
Opening_Times	time_id (PK), department_id (FK), day, opening_time, closing_time

Draw ER Diagram



Task Two – Create & Populate the Database

1. DDL Statements

Code in “DDL Statements.sql”

```
-- Create Database
CREATE DATABASE UniversityDB;
USE UniversityDB;

-- Create Tables
-- Departments Table
CREATE TABLE Departments (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    school VARCHAR(100) NOT NULL,
    phone_number CHAR(15) NOT NULL
);

-- Courses Table
CREATE TABLE Courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    academic_level INT NOT NULL,
    department_id INT NOT NULL,
    delivery_method VARCHAR(50) NOT NULL,
    semesters INT NOT NULL,
    participants INT NOT NULL,
    FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);

-- Students Table
CREATE TABLE Students (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
```

```

        date_of_birth DATE NOT NULL,
        address VARCHAR(255) NOT NULL,
        email VARCHAR(100) UNIQUE NOT NULL,
        course_id INT NOT NULL,
        outstanding_fees_due INT DEFAULT 0 NOT NULL,
        year INT NOT NULL,
        status VARCHAR(50) NOT NULL,
        FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    );

-- Results Table
CREATE TABLE Results (
    result_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT NOT NULL,
    grade CHAR(2) NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);

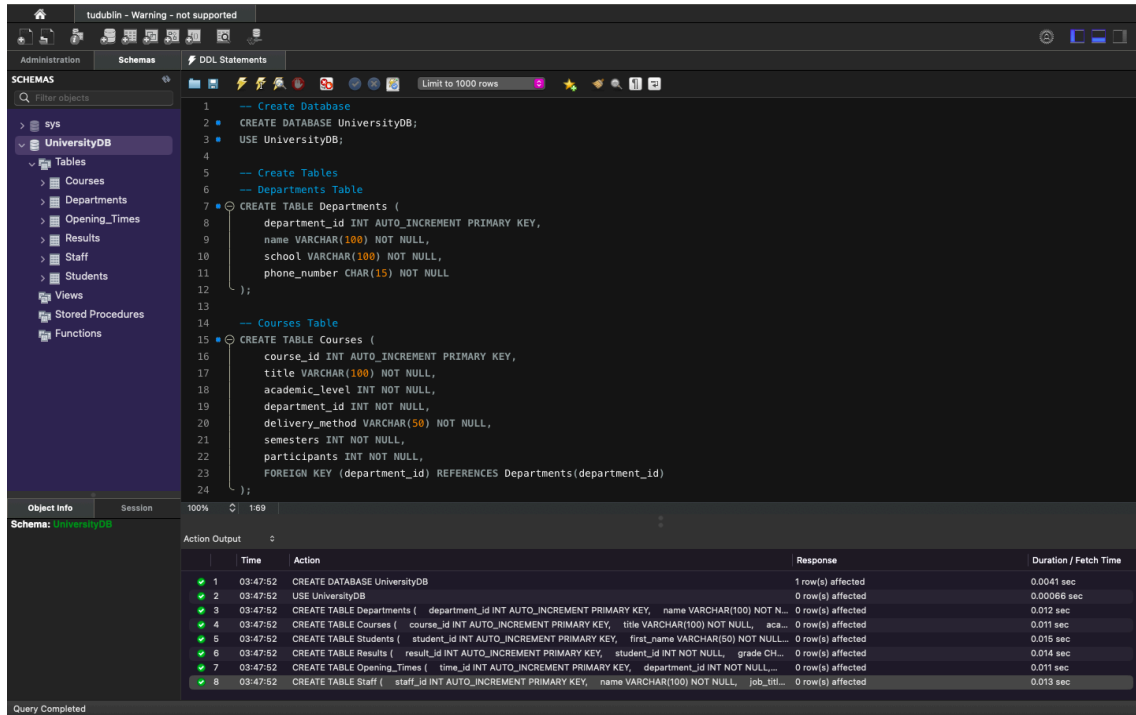
-- Opening_Times Table
CREATE TABLE Opening_Times (
    time_id INT AUTO_INCREMENT PRIMARY KEY,
    department_id INT NOT NULL,
    day VARCHAR(20) NOT NULL,
    opening_time TIME NOT NULL,
    closing_time TIME NOT NULL,
    FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);

-- Staff Table
CREATE TABLE Staff (
    staff_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    job_title VARCHAR(50) NOT NULL,
    department_id INT NOT NULL,
    years_at_university INT NOT NULL,
    days_off INT NOT NULL,

```

```
FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);
```

Screenshot



2. DML Statements

Code in "DML Statements.sql"

```
-- Populate Tables
```

```
-- Departments Table
```

```
INSERT INTO Departments (name, school, phone_number) VALUES
('School of Computing', 'Computing', '123456789'),
('School of Business', 'Business', '123456789'),
('School of Magic', 'Magic', '123456789'),
('School of Arts', 'Arts', '123456789'),
('School of Sciences', 'Sciences', '123456789'),
('School of Law', 'Law', '123456789'),
('School of Medicine', 'Medicine', '123456789'),
('School of Education', 'Education', '123456789');
```

```
('School of Social Sciences', 'Social Sciences', '123456789'),
('School of Music', 'Music', '123456789');
```

```
-- Courses Table
```

```
INSERT INTO Courses (title, academic_level, department_id,
delivery_method, semesters, participants) VALUES
('Cyber Security', 8, 1, 'On-Campus', 8, 30),
('Computer Science', 7, 1, 'Online', 6, 50),
('Business Administration', 6, 2, 'Online', 3, 25),
('Accounting Disappearance', 10, 2, 'Hybrid', 1, 10),
('Houdini Tricks', 7, 3, 'On-Campus', 3, 10),
('Advanced Potions', 9, 3, 'Hybrid', 1, 20),
('Fine Arts', 8, 4, 'On-Campus', 6, 15),
('Black Square is the Best Art?', 6, 4, 'Online', 2, 5),
('Physics', 9, 5, 'Online', 8, 30),
('Astrophysics', 10, 5, 'Online', 2, 12),
('Corporate Law', 9, 6, 'On-Campus', 2, 18),
('Batman\'s Law', 10, 6, 'On-Campus', 10, 1),
('Pharmacy', 9, 7, 'On-Campus', 8, 30),
('Medical Sciences', 10, 7, 'Hybrid', 9, 35),
('Education Theory', 5, 8, 'Online', 1, 22),
('Artificial Intelligence', 10, 8, 'Online', 6, 15),
('Cultural Studies', 7, 9, 'On-Campus', 2, 28),
('Civil Engineering', 8, 3, 'Hybrid', 6, 18),
('Musical Composition', 10, 10, 'Hybrid', 9, 10),
('Stop Listenig Pop', 4, 10, 'Online', 1, 10);
```

```
-- Students Table
```

```
INSERT INTO Students (first_name, last_name, date_of_birth,
address, email, course_id, outstanding_fees_due, year, status)
VALUES
('Dany', 'qwerty', '2004-12-08', 'Far Far Away',
'qwerty@dany.com', 1, 0, 2, 'Active'),
('John', 'Doe', '2000-05-15', '123 Maple Street',
'johndoe@gmail.com', 1, 0, 1, 'Active'),
('Jane', 'Smith', '1999-10-22', '456 Oak Avenue',
'janesmith@gmail.com', 2, 2, 2, 'Active'),
```

```
(
'Alice', 'Brown', '2001-01-17', '789 Pine Lane',
'alicebrown@gmail.com', 3, 6, 1, 'Active'),
('Bob', 'White', '1998-07-10', '321 Elm Drive',
'bobwhite@gmail.com', 4, 0, 3, 'Active'),
('Charlie', 'Green', '2002-09-25', '654 Birch Road',
'charliegreen@gmail.com', 5, 3, 1, 'Active'),
('David', 'Black', '2000-02-14', '987 Cedar Boulevard',
'davidblack@gmail.com', 6, 0, 2, 'Active'),
('Emma', 'Gray', '1999-06-30', '111 Spruce Street',
'emmagray@gmail.com', 7, 0, 4, 'Completed'),
('Frank', 'Blue', '2001-12-12', '222 Walnut Way',
'frankblue@gmail.com', 8, 4, 3, 'Active'),
('Grace', 'Yellow', '2000-03-21', '333 Chestnut Court',
'gracey@gmail.com', 9, 0, 2, 'Active'),
('Hannah', 'Purple', '1998-11-11', '444 Aspen Place',
'hannahp@gmail.com', 10, 0, 4, 'Completed'),
('Ivy', 'Orange', '2001-05-05', '555 Maple Lane',
'ivyorange@gmail.com', 12, 0, 1, 'Active'),
('Jack', 'Red', '1997-07-16', '666 Pine Court',
'jackred@gmail.com', 13, 8, 3, 'Active'),
('Karen', 'Pink', '2003-11-23', '777 Cedar Drive',
'karenpink@gmail.com', 14, 0, 2, 'Active'),
('Leo', 'Brown', '2002-08-10', '888 Birch Avenue',
'leobrown@gmail.com', 15, 1, 1, 'Active'),
('Maria', 'Silver', '1999-09-18', '999 Aspen Street',
'mariasilver@gmail.com', 16, 0, 4, 'Completed'),
('Nathan', 'Gold', '2000-04-12', '101 Elm Road',
'nathangold@gmail.com', 17, 0, 3, 'Active'),
('Olivia', 'Teal', '2001-06-28', '202 Walnut Boulevard',
'oliviateal@gmail.com', 18, 0, 2, 'Active'),
('Paul', 'Green', '1998-02-20', '303 Spruce Lane',
'paulgreen@gmail.com', 19, 6, 4, 'Completed'),
('Quinn', 'Blue', '1997-01-14', '404 Chestnut Drive',
'quinnblue@gmail.com', 20, 0, 1, 'Active');

```

```
-- Results Table
```

```
INSERT INTO Results (student_id, grade) VALUES
(1, 'A'), (1, 'A'),
```

```

(2, 'B'),
(3, 'C'), (3, 'B'),
(4, 'B'),
(5, 'A'), (5, 'A'), (5, 'A'),
(6, 'C'),
(7, 'B'), (7, 'A'),
(8, 'A'), (8, 'A'), (8, 'A'), (8, 'A'),
(9, 'B'), (9, 'B'), (9, 'A'),
(10, 'C'), (10, 'B'),
(11, 'A'), (11, 'A'), (11, 'A'), (11, 'A'),
(12, 'B'),
(13, 'C'), (13, 'B'), (13, 'A'),
(14, 'C'), (14, 'B'),
(15, 'B'),
(16, 'A'), (16, 'A'), (16, 'A'), (16, 'A'),
(17, 'B'), (17, 'A'), (17, 'A'),
(18, 'C'), (18, 'B'),
(19, 'A'), (19, 'A'), (19, 'A'), (19, 'A'),
(20, 'B');

```

-- Opening_Times Table

```

INSERT INTO Opening_Times (department_id, day, opening_time,
closing_time) VALUES

```

```

(1, 'Monday', '07:00:00', '20:00:00'),
(2, 'Tuesday', '08:30:00', '17:30:00'),
(3, 'Tuesday', '09:00:00', '19:00:00'),
(4, 'Wednesday', '08:00:00', '19:00:00'),
(5, 'Wednesday', '08:00:00', '18:00:00'),
(6, 'Thursday', '09:00:00', '18:00:00'),
(7, 'Thursday', '09:00:00', '17:00:00'),
(8, 'Friday', '10:00:00', '17:00:00'),
(9, 'Saturday', '10:00:00', '14:00:00'),
(10, 'Sunday', '12:00:00', '14:00:00');

```

-- Staff Table

```

INSERT INTO Staff (name, job_title, department_id,
years_at_university, days_off) VALUES
('Prof. Nobody', 'Dean', 1, 37, 0),

```

```

('Dr. Amy Watson', 'Lecturer', 1, 5, 12),
('Mr. Mark Johnson', 'Administrator', 2, 10, 5),
('Ms. Sarah Brown', 'Assistant', 2, 2, 8),
('Dr. Michael Lee', 'Researcher', 3, 7, 10),
('Ms. Olivia Davis', 'Lecturer', 3, 3, 9),
('Mr. James Harris', 'Technician', 4, 6, 15),
('Dr. Linda Martinez', 'Researcher', 4, 9, 6),
('Ms. Emily Clark', 'Office Worker', 5, 1, 5),
('Mr. Ryan King', 'Lecturer', 5, 4, 11),
('Dr. Angela Wright', 'Lecturer', 6, 12, 18),
('Mr. Daniel Evans', 'Technician', 6, 8, 14),
('Ms. Rebecca Adams', 'Office Worker', 7, 3, 7),
('Prof. Christopher Scott', 'Dean', 7, 20, 22),
('Mr. Kevin Carter', 'Administrator', 8, 5, 9),
('Dr. Isabella Young', 'Researcher', 8, 6, 10),
('Ms. Mia Hill', 'Assistant', 9, 4, 8),
('Dr. Liam Walker', 'Lecturer', 9, 11, 16),
('Mr. William Hall', 'Technician', 10, 9, 12),
('Ms. Sophia Allen', 'Office Worker', 10, 2, 4);

```

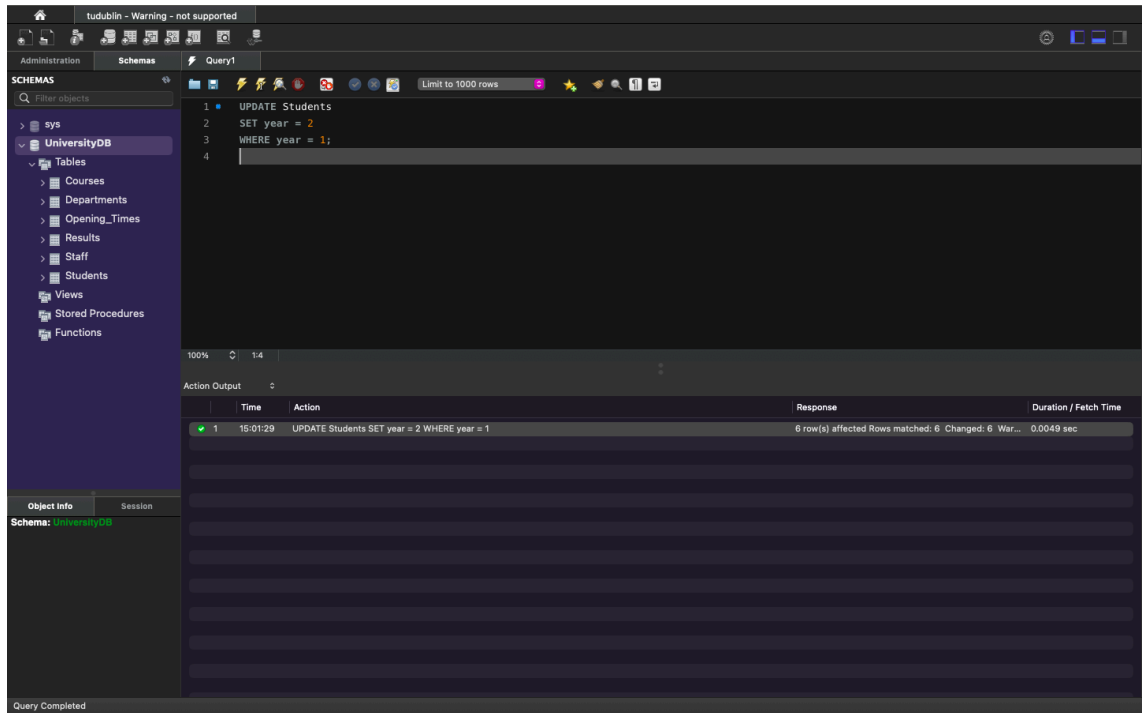
Screenshot

The screenshot shows a database management tool interface with a dark theme. The left sidebar displays a tree view of the database schema, including tables like Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions. The main area shows a list of SQL statements being executed. The bottom section displays the 'Action Output' table, which provides details on the execution of each statement, including the time taken, the action performed, the response, and the duration/fetch time.

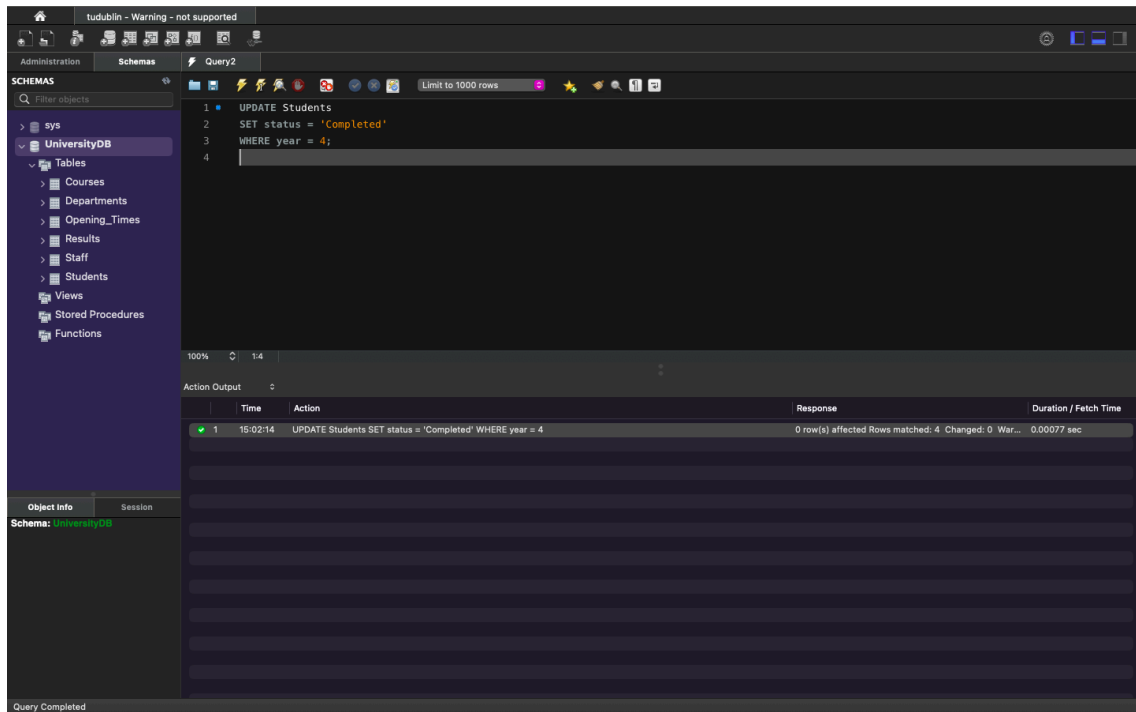
Time	Action	Response	Duration / Fetch Time
03:50:01	INSERT INTO Departments (name, school, phone_number) VALUES ('School of Computing', 'Computing', '123456789')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.0048 sec
03:50:01	INSERT INTO Courses (title, academic_level, department_id, delivery_method, semesters, participants) VALUES ('Cyber Security', 8, 1, 'On-Campus', 8, 30)	20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0	0.0026 sec
03:50:01	INSERT INTO Students (first_name, last_name, date_of_birth, address, email, course_id, outstanding_fees_due, year, ...)	20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0	0.0032 sec
03:50:01	INSERT INTO Results (student_id, grade) VALUES (1, 'A'), (1, 'A'), (2, 'B'), (3, 'C'), (3, 'B'), (4, 'B'), (5, 'A'), (5, 'A'), (5, 'A'), ...	46 row(s) affected Records: 46 Duplicates: 0 Warnings: 0	0.0025 sec
03:50:01	INSERT INTO Opening_Times (department_id, day, opening_time, closing_time) VALUES (1, 'Monday', '07:00:00', '20:00:00')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.0028 sec
03:50:01	INSERT INTO Staff (name, job_title, department_id, years_at_university, days_off) VALUES ('Prof. Nobody', 'Dean', 1, 3, ...)	20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0	0.0039 sec

Task Three – Query The Dataset

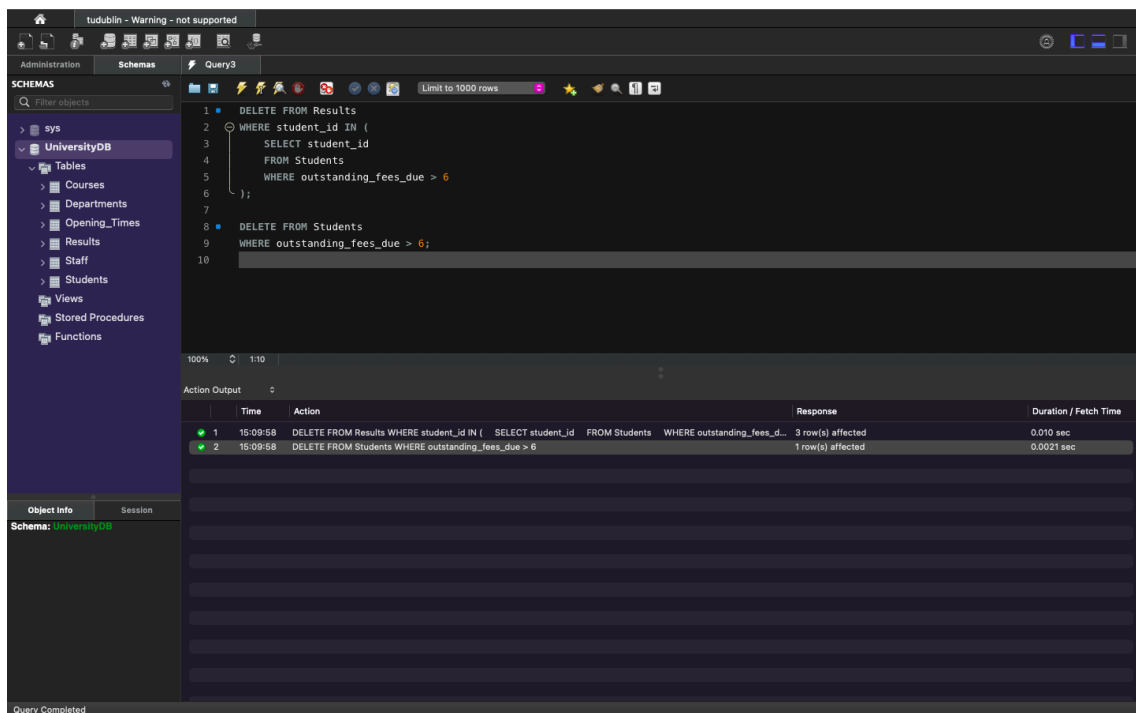
1. Change all students in year one of any course to year two.



2. Modify the status of any student in year 4 of all courses to indicate that they have now completed the course.



3. Delete all students who have outstanding fees which have not been paid for more than 6 months.



4. Display the first name surname and grade of all students sorting the results so the highest grades are first.

The screenshot shows a database management tool interface. On the left, a sidebar displays the 'UniversityDB' schema with tables like Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions. The main area shows a SQL query (Query4) that selects the first name, last name, and grade of all students, sorted by grade in descending order. The query is as follows:

```
1 SELECT s.first_name AS "First Name", s.last_name AS "Last Name", r.grade AS "Last Grade"
2 FROM Students s
3 JOIN Results r ON s.student_id = r.student_id
4 WHERE r.result_id IN (SELECT MAX(result_id) FROM Results GROUP BY student_id)
5 ORDER BY r.grade ASC;
```

The results are displayed in a table with columns 'First Name', 'Last Name', and 'Last Grade'. The results are sorted by grade in descending order, showing students like Darcy Lowerty with grade A, Bob White with grade A, David Black with grade A, Emma Gray with grade A, Frank Blue with grade A, Hannah Purple with grade A, Maria Silver with grade A, Nathan Gold with grade A, Paul Green with grade A, John Doe with grade B, Jane Smith with grade B, Alice Brown with grade B, Grace Yellow with grade B, Ivy Orange with grade B, Karen Pink with grade B, Leo Brown with grade B, Olivia Red with grade B, Quinn Blue with grade B, and Charlie Green with grade C.

5. Add one new record to each table.

The screenshot shows a database management tool interface. On the left, a sidebar displays the 'UniversityDB' schema with tables like Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions. The main area shows a SQL script (Query5) that adds one new record to each table. The script is as follows:

```
1 -- Add to Departments Table
2 INSERT INTO Departments (name, school, phone_number)
3 VALUES ('School of AI', 'Artificial Intelligence', '01-9876543');
4
5 -- Add to Courses Table
6 INSERT INTO Courses (title, academic_level, department_id, delivery_method, semesters, participants)
7 VALUES ('Deep Learning', 9, (SELECT department_id FROM Departments WHERE name = 'School of AI'), 'Online', 6, 40);
8
9 -- Add to Students Table
10 INSERT INTO Students (first_name, last_name, date_of_birth, address, email, course_id, outstanding_fees_due, year, status)
11 VALUES ('Test', 'Student', '2003-01-01', 'Test Address', 'test.student@example.com', (SELECT course_id FROM Courses WHERE title = 'Deep Learning'),
12
13 -- Add to Results Table
14 INSERT INTO Results (student_id, grade)
15 VALUES ((SELECT student_id FROM Students WHERE first_name = 'Test' AND last_name = 'Student'), 'A');
16
17 -- Add to Opening_Times Table
18 INSERT INTO Opening_Times (department_id, day, opening_time, closing_time)
19 VALUES ((SELECT department_id FROM Departments WHERE name = 'School of AI'), 'Saturday', '09:00:00', '17:00:00');
20
21 -- Add to Staff Table
22 INSERT INTO Staff (name, job_title, department_id, years_at_university, days_off)
23 VALUES ('Test Staff', 'Lecturer', (SELECT department_id FROM Departments WHERE name = 'School of AI'), 1, 2);
24
```

The script is executed, and the results are displayed in a table with columns 'Time', 'Action', 'Response', and 'Duration / Fetch Time'. The results show that the script successfully added one new record to each table.

6. Delete one record from all tables.

The screenshot shows a database management tool interface with a SQL query editor. The query is designed to delete one record from each table in the UniversityDB schema. The tables targeted are Staff, Opening_Times, Results, Students, Courses, and Departments. The query uses subqueries to identify specific records to delete based on unique identifiers like name, department_id, student_id, first_name, last_name, and title.

```
1 -- Delete from Staff Table
2 DELETE FROM Staff
3 WHERE name = 'Test Staff';
4
5 -- Delete from Opening_Times Table
6 DELETE FROM Opening_Times
7 WHERE department_id = (SELECT department_id FROM Departments WHERE name = 'School of AI');
8
9 -- Delete from Results Table
10 DELETE FROM Results
11 WHERE student_id = (SELECT student_id FROM Students WHERE first_name = 'Test' AND last_name = 'Student');
12
13 -- Delete from Students Table
14 DELETE FROM Students
15 WHERE first_name = 'Test' AND last_name = 'Student';
16
17 -- Delete from Courses Table
18 DELETE FROM Courses
19 WHERE title = 'Deep Learning';
20
21 -- Delete from Departments Table
22 DELETE FROM Departments
23 WHERE name = 'School of AI';
24
```

The Action Output section shows the execution results for each table:

Time	Action	Response	Duration / Fetch Time
15:31:57	DELETE FROM Staff WHERE name = 'Test Staff'	1 row(s) affected	0.0051 sec
15:31:57	DELETE FROM Opening_Times WHERE department_id = (SELECT department_id FROM Departments WHERE name = ...	1 row(s) affected	0.0024 sec
15:31:57	DELETE FROM Results WHERE student_id = (SELECT student_id FROM Students WHERE first_name = 'Test' AND last...	1 row(s) affected	0.0064 sec
15:31:57	DELETE FROM Students WHERE first_name = 'Test' AND last_name = 'Student'	1 row(s) affected	0.0020 sec
15:31:57	DELETE FROM Courses WHERE title = 'Deep Learning'	1 row(s) affected	0.0017 sec
15:31:57	DELETE FROM Departments WHERE name = 'School of AI'	1 row(s) affected	0.0044 sec

Query Completed

7. Find the total number of days off for all staff, order this by least days off.

The screenshot shows a database management tool interface with a SQL query editor. The query is designed to find the total number of days off for all staff, ordered by least days off.

```
1 SELECT name AS "Staff Name", days_off AS "Days Off"
2 FROM Staff
3 ORDER BY days_off ASC;
4
```

The Result Grid shows the execution results:

Staff Name	Days Off
Prof. Nobody	0
Mrs. Sophia Allen	4
Mr. Mark Johnson	5
Mrs. Emily Clark	6
Dr. Linda Martinez	6
Mrs. Rebecca Adams	7
Mr. Sarah Brown	8
Mrs. Mia Hill	8
Mrs. Olivia Davis	9
Mr. Kevin Carter	9
Dr. Michael Lee	10
Dr. Isabelle Young	10
Mr. Ryan King	11
Dr. Amy Watson	12
Mr. William Hall	12
Mr. Daniel Evans	14
Mr. James Harris	15
Dr. Liam Walker	16
Dr. Angela Wright	16
Prof. Christopher S...	22

Staff 11

Query Completed

8. Count how many students are doing a business course.

The screenshot shows a database query tool interface. On the left, a sidebar lists the database schema 'UniversityDB' with tables: Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions. The main query editor displays the following SQL code:

```
1 SELECT COUNT(*) AS "Student Count"
2 FROM Students s
3 JOIN Courses c ON s.course_id = c.course_id
4 JOIN Departments d ON c.department_id = d.department_id
5 WHERE d.name = 'School of Business';
6
```

Below the query editor, the 'Result Grid' shows a single row with the value '2' under the column 'Student Count'. The status bar at the bottom indicates 'Query Completed'.

9. Change the roll of all staff who have a job title Office worker to Administrator.

The screenshot shows the same database query tool interface. The main query editor displays the following SQL code:

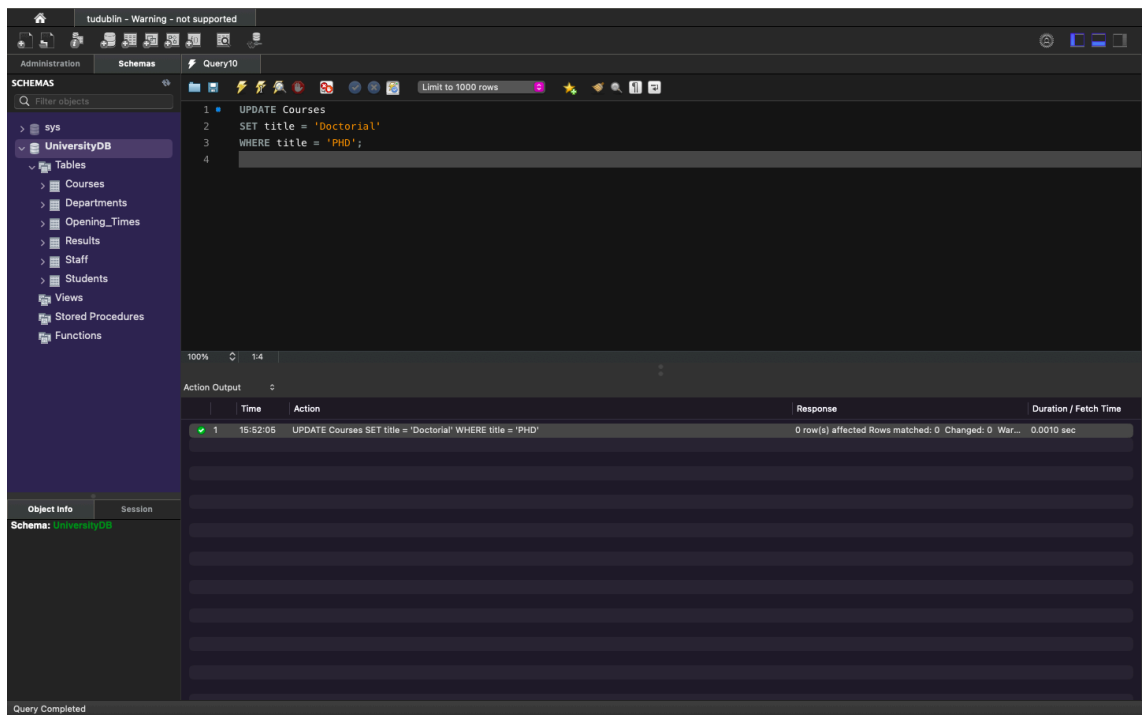
```
1 UPDATE Staff
2 SET job_title = 'Administrator'
3 WHERE job_title = 'Office Worker';
4
```

Below the query editor, the 'Action Output' table shows the execution results:

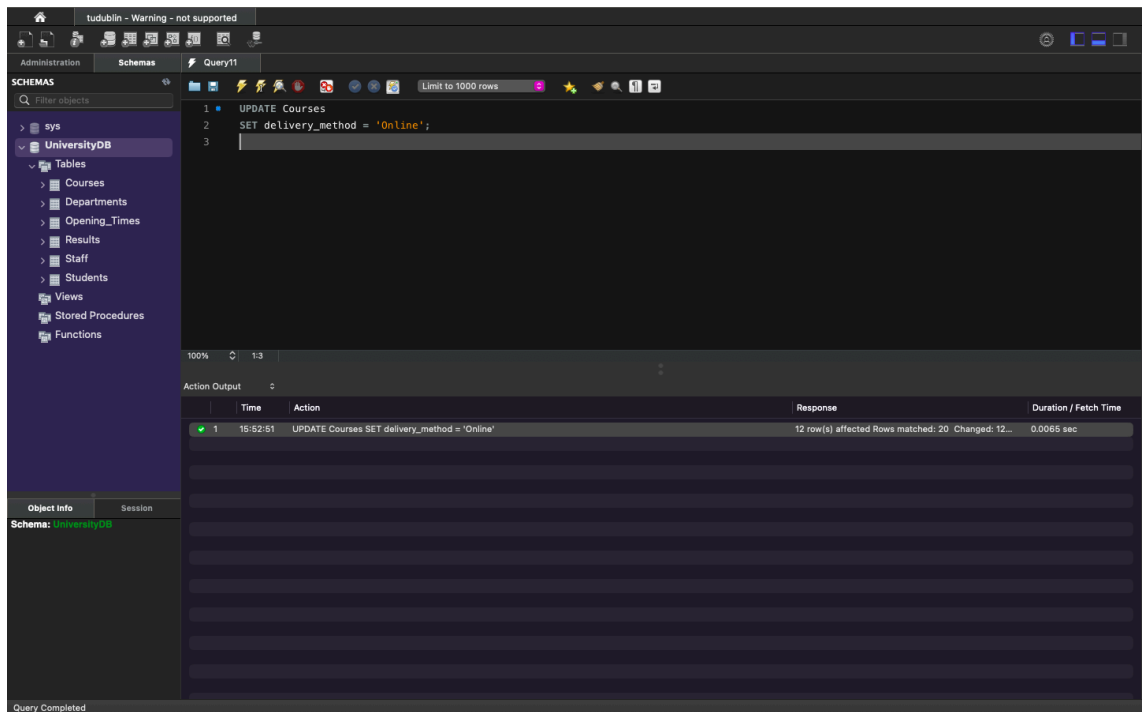
	Time	Action	Response	Duration / Fetch Time
1	15:51:18	UPDATE Staff SET job_title = 'Administrator' WHERE job_title = 'Office Worker'	3 row(s) affected Rows matched: 3 Changed: 3 War...	0.0053 sec

The status bar at the bottom indicates 'Query Completed'.

10. Change all courses entitled PHD to Doctorial.



11. Set the delivery method of all courses to online.



12. Update the opening times of the university to say closed to visitors.

The screenshot shows a SQL Studio interface with a query window titled 'Query12'. The query contains the following SQL statements:

```
1 ALTER TABLE Opening_Times
2 MODIFY opening_time VARCHAR(50),
3 MODIFY closing_time VARCHAR(50);
4
5 UPDATE Opening_Times
6 SET opening_time = 'Closed to Visitors', closing_time = 'Closed to Visitors';
7
```

The 'Action Output' pane shows the execution results:

	Time	Action	Response	Duration / Fetch Time
✓	16:02:47	ALTER TABLE Opening_Times MODIFY opening_time VARCHAR(50), MODIFY closing_time VARCHAR(50)	10 row(s) affected Records: 10 Duplicates: 0 Warnings...	0.066 sec
✓	16:02:47	UPDATE Opening_Times SET opening_time = 'Closed to Visitors', closing_time = 'Closed to Visitors'	10 row(s) affected Rows matched: 10 Changed: 10 Warnings...	0.0026 sec

The 'Object Info' pane shows the schema 'UniversityDB' with a list of tables including Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions.

13. Drop all information contained in the courses relation.

The screenshot shows a SQL Studio interface with a query window titled 'Query13'. The query contains the following SQL statements:

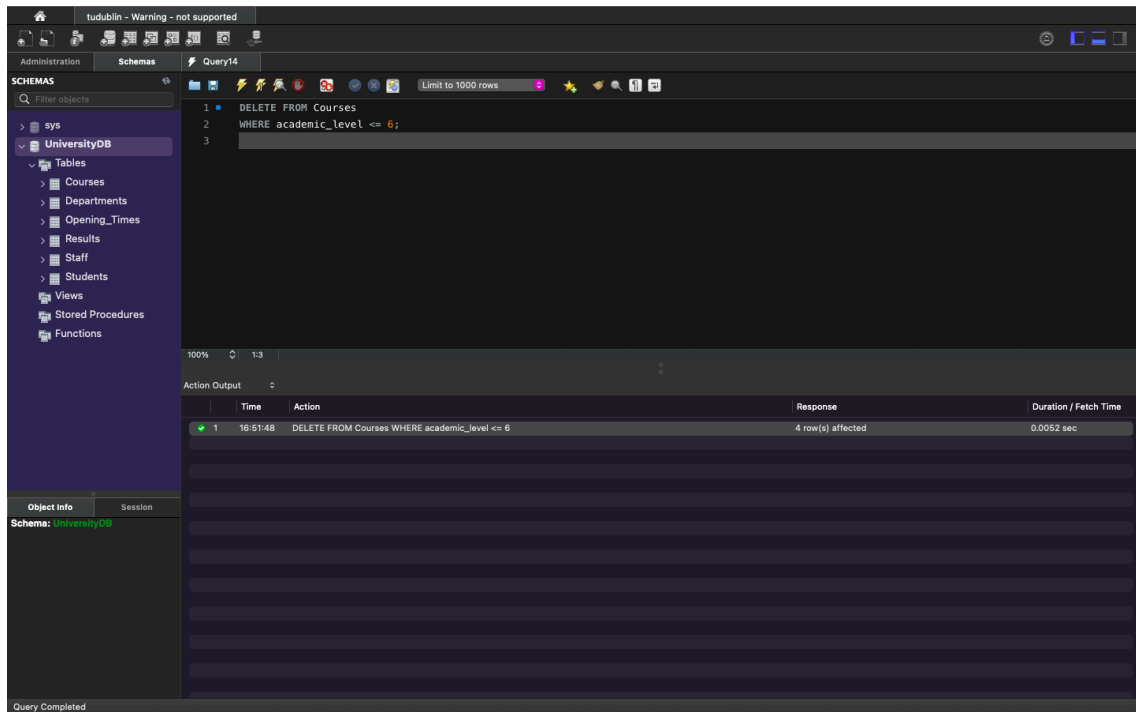
```
1 ALTER TABLE Students
2 DROP FOREIGN KEY students_ibfk_1;
3
```

The 'Action Output' pane shows the execution results:

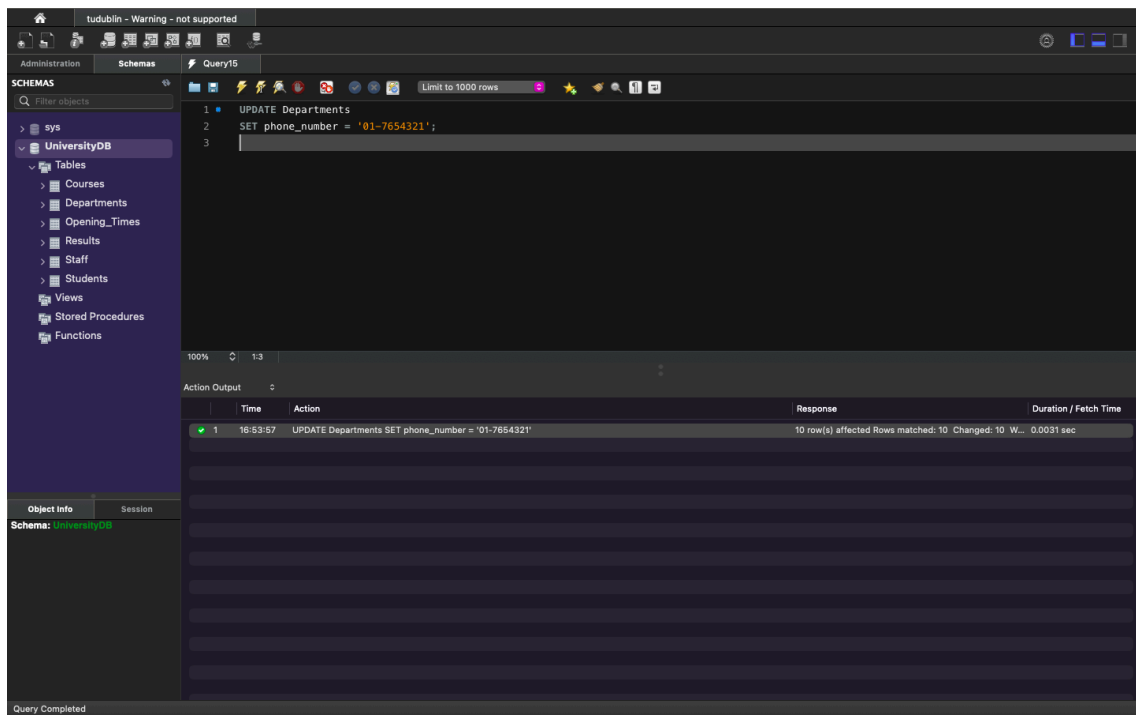
	Time	Action	Response	Duration / Fetch Time
✓	16:50:28	ALTER TABLE Students DROP FOREIGN KEY students_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.020 sec

The 'Object Info' pane shows the schema 'UniversityDB' with a list of tables including Courses, Departments, Opening_Times, Results, Staff, Students, Views, Stored Procedures, and Functions.

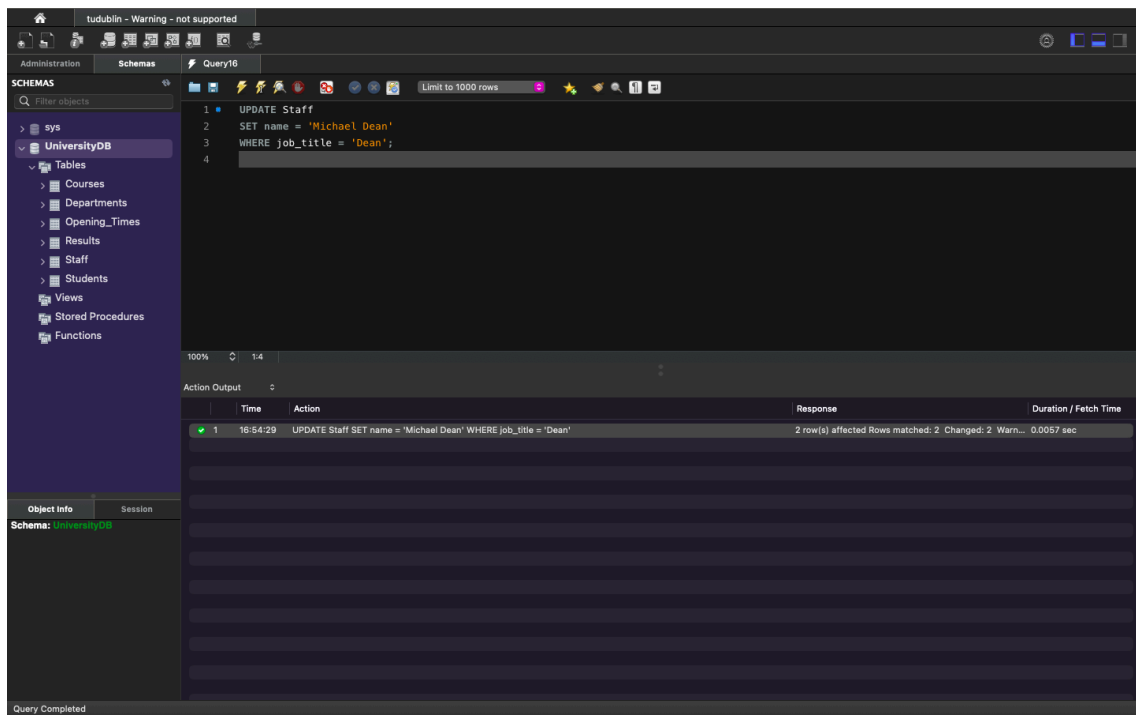
14. Delete all courses from the database with an academic level of 6 or less.



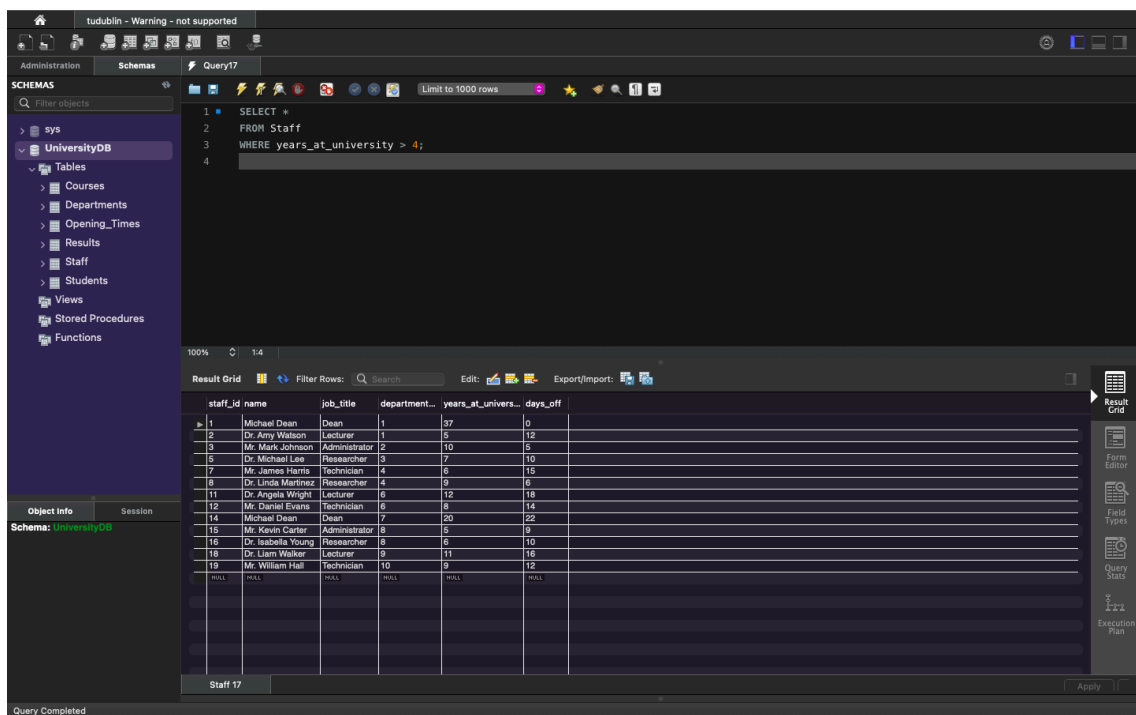
15. Set the university phone number to be 01-7654321.



16. Set the dean of the university to be called Michael Dean.



17. Show all staff members who have been working there for longer than 4 years.



18. Show all courses that the university offers which run over 3 semesters and have a minimum of 20 participants per class.

The screenshot shows a database management tool interface. The left sidebar displays the 'UniversityDB' schema with various tables. The main query editor contains the following SQL code:

```
1 SELECT *
2 FROM Courses
3 WHERE semesters > 3 AND participants >= 20;
```

The 'Result Grid' shows the following data:

course_id	title	academic_level	department	delivery_meth...	semesters	participan...
1	Cyber Security	8	1	Online	8	30
2	Computer Science	7	1	Online	6	50
9	Physics	9	5	Online	8	30
13	Pharmacy	9	7	Online	8	30
14	Medical Sciences	10	7	Online	8	35

19. Identify how many students have the word road in their address.

The screenshot shows the same database management tool interface. The query editor contains the following SQL code:

```
1 SELECT COUNT(*) AS "Students with Road in Address"
2 FROM Students
3 WHERE address LIKE '%Road%';
```

The 'Result Grid' shows the following data:

Students with Road in Addr...
2

At the bottom of the window, a status bar indicates: "SQL script saved to 'C:\Users\dany\Desktop\Database Fundamental Project\Query The Dataset\Query19.sql'" and a "Read Only" button is visible.

20. Create a view that will show the result of a query drawing information from three tables at once.

The screenshot shows a database management interface with a dark theme. On the left, a sidebar displays the 'UniversityDB' schema with various tables and views. The main area shows a SQL query in a text editor, and below it, a 'Result Grid' displaying the output of the query. The query creates a view named 'Student_Course_Grades' and then selects from it. The result grid shows a table with four columns: 'first_name', 'last_name', 'course_title', and 'grade'. The data includes students like Dany, John, Jane, Bob, David, Emma, Hannah, and Ivy, each associated with a specific course and grade.

```
1 CREATE VIEW Student_Course_Grades AS
2 SELECT s.first_name, s.last_name, c.title AS course_title, r.grade
3 FROM Students s
4 JOIN Courses c ON s.course_id = c.course_id
5 JOIN Results r ON s.student_id = r.student_id;
6
7 SELECT * FROM Student_Course_Grades;
8
```

first_name	last_name	course_title	grade
Dany	qewerty	Cyber Security	A
Dany	qewerty	Cyber Security	A
John	Doe	Cyber Security	B
Jane	Smith	Computer Science	C
Jane	Smith	Computer Science	B
Bob	White	Accounting Disappearance	A
Bob	White	Accounting Disappearance	A
Bob	White	Accounting Disappearance	A
Charlie	Green	Houdini Tricks	C
David	Black	Advanced Potions	B
David	Black	Advanced Potions	A
Emma	Gray	Fine Arts	A
Emma	Gray	Fine Arts	A
Emma	Gray	Fine Arts	A
Emma	Gray	Fine Arts	A
Grace	Yellow	Physics	C
Grace	Yellow	Physics	B
Hannah	Purple	Astrophysics	A
Hannah	Purple	Astrophysics	A
Hannah	Purple	Astrophysics	A
Hannah	Purple	Astrophysics	A
Ivy	Orange	Batman's Law	B

Student_Course_Grades 21

Query Completed