**CS 6700 Introduction to Database Management Systems Project**

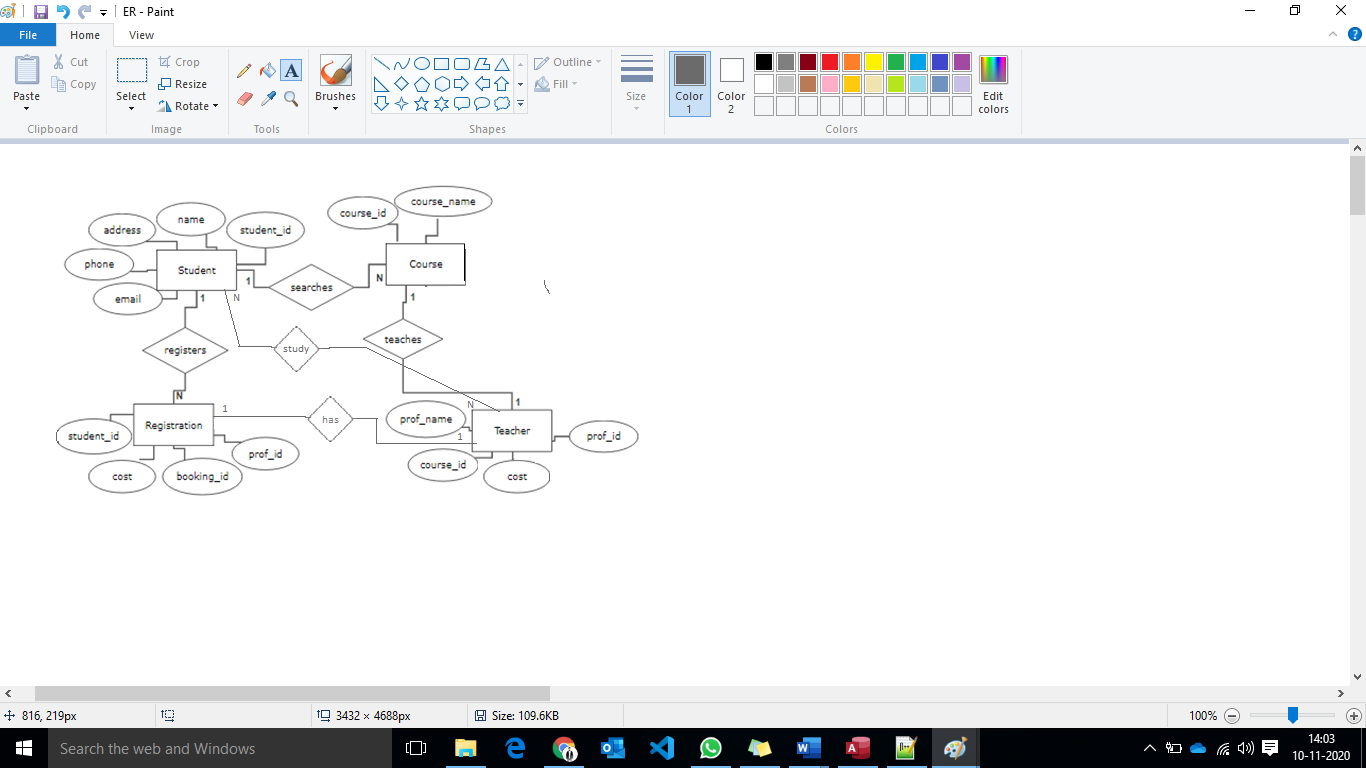
**Fall 2020**

(Nikhil Chavan, U00969241, chavan.10@wright.edu)

**Project Idea:** Online Education Course Enrolment.

**Tools:** Microsoft Access Database

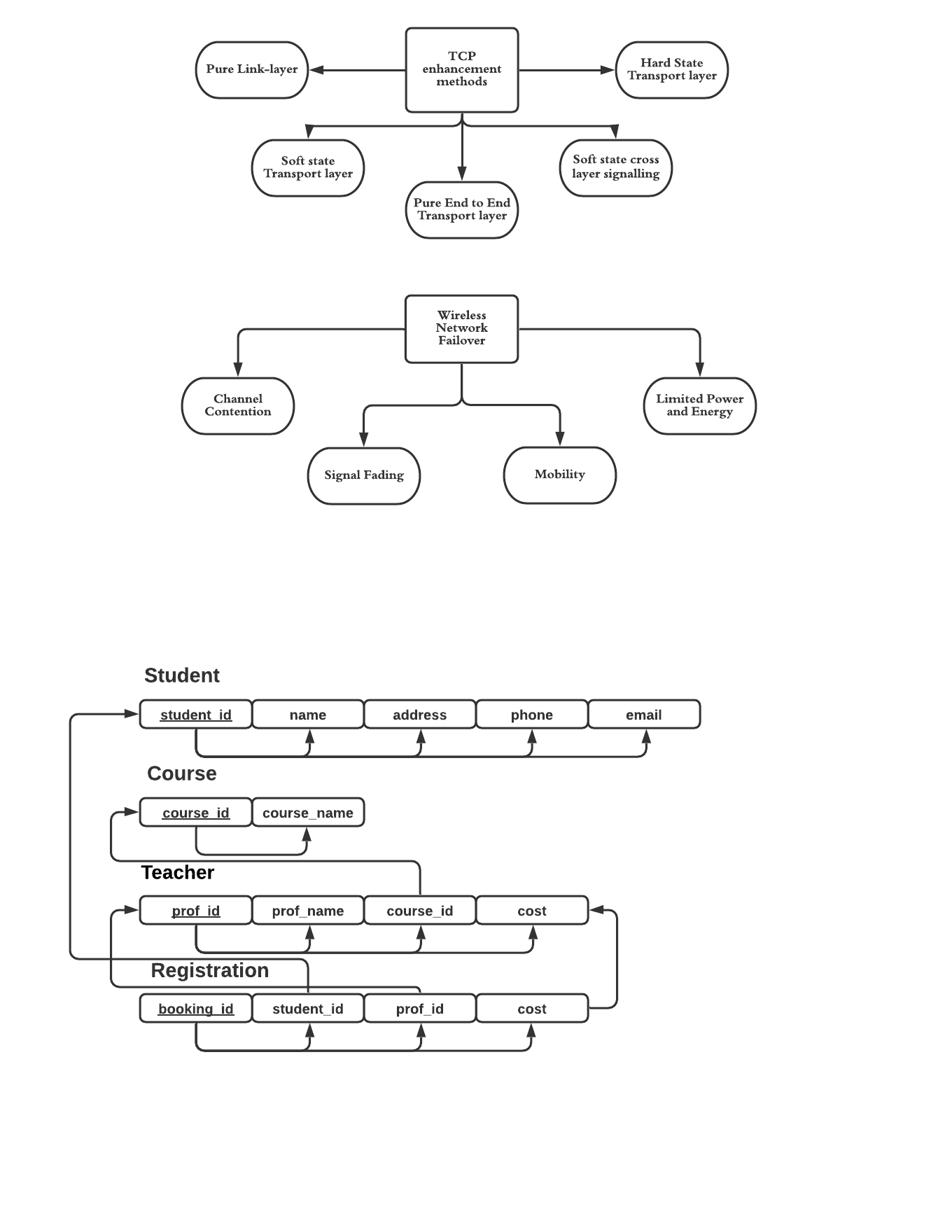
**Entity Relationship Diagram(ERD):**



**Assumptions for Relationships between the Tables:**

* **Student to Course (1 to N):** A student could search and enrol in different courses available in the e-learning website.
* **Course to Teacher(1 to 1):** A teacher or professor could only teach one subject, different teacher could also teach same course as other but at different timing.
* **Student to Registration (1 to N):** A student could enrol in how many courses he wanted to register.
* **Registration to Teacher (1 to 1):** A single registration record could have a single professor associated with the student registration.
* **Student to Teacher (N to N):** A Student could study under different professors whereas a teacher could teach different students

**Relational Schema:**

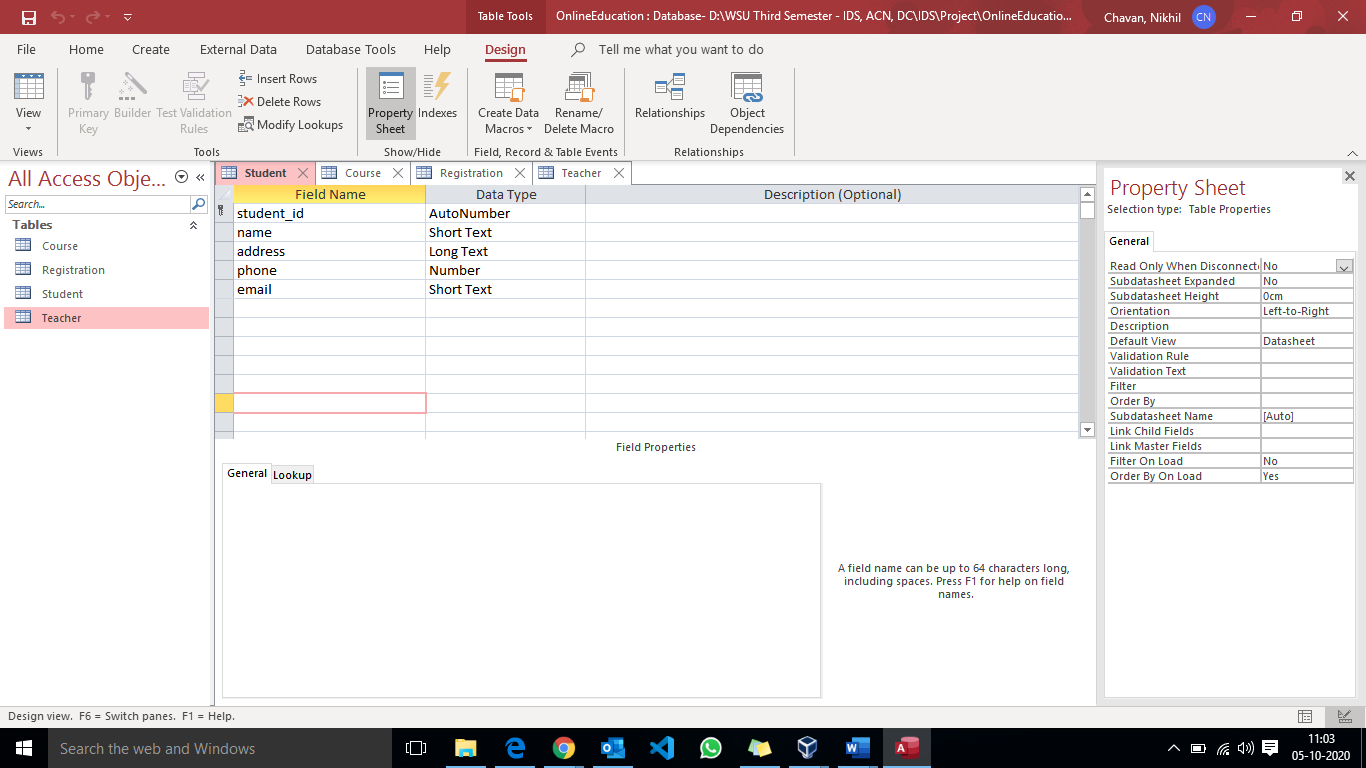
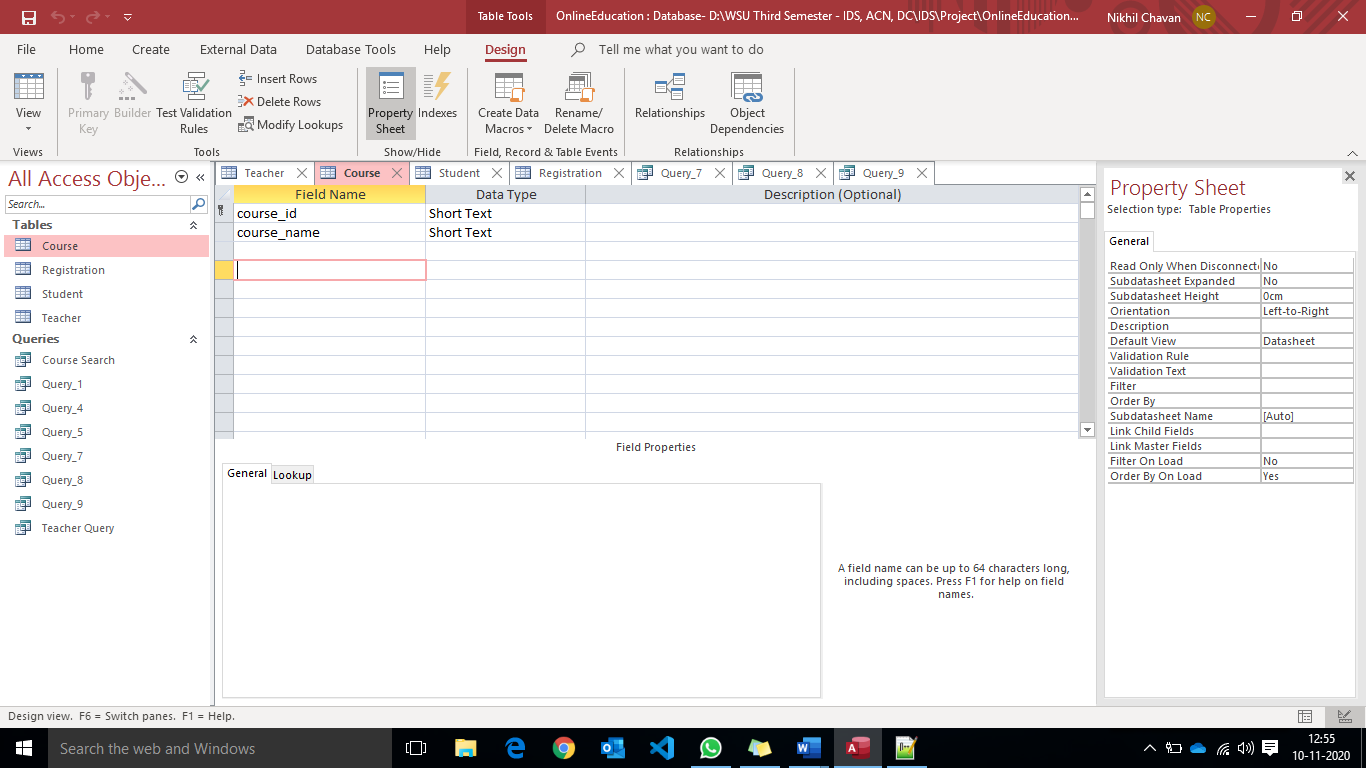


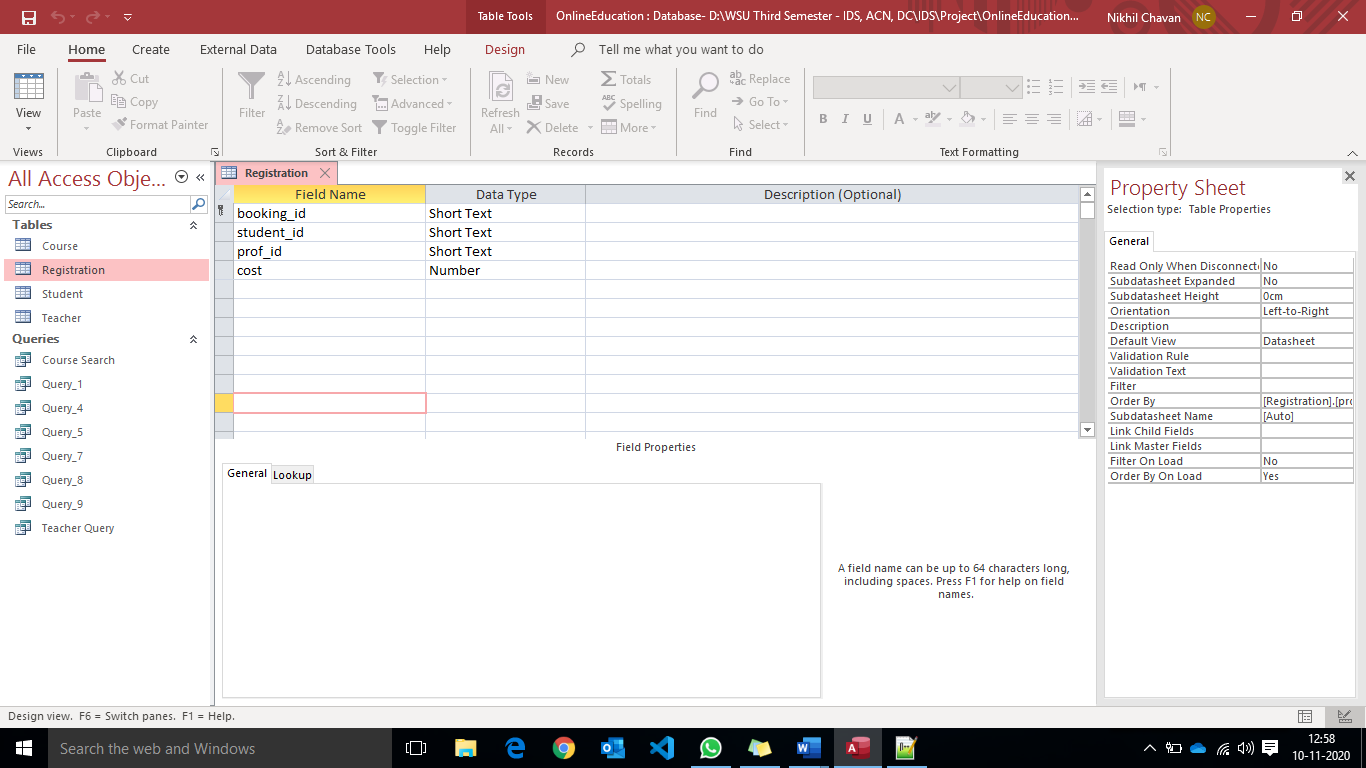
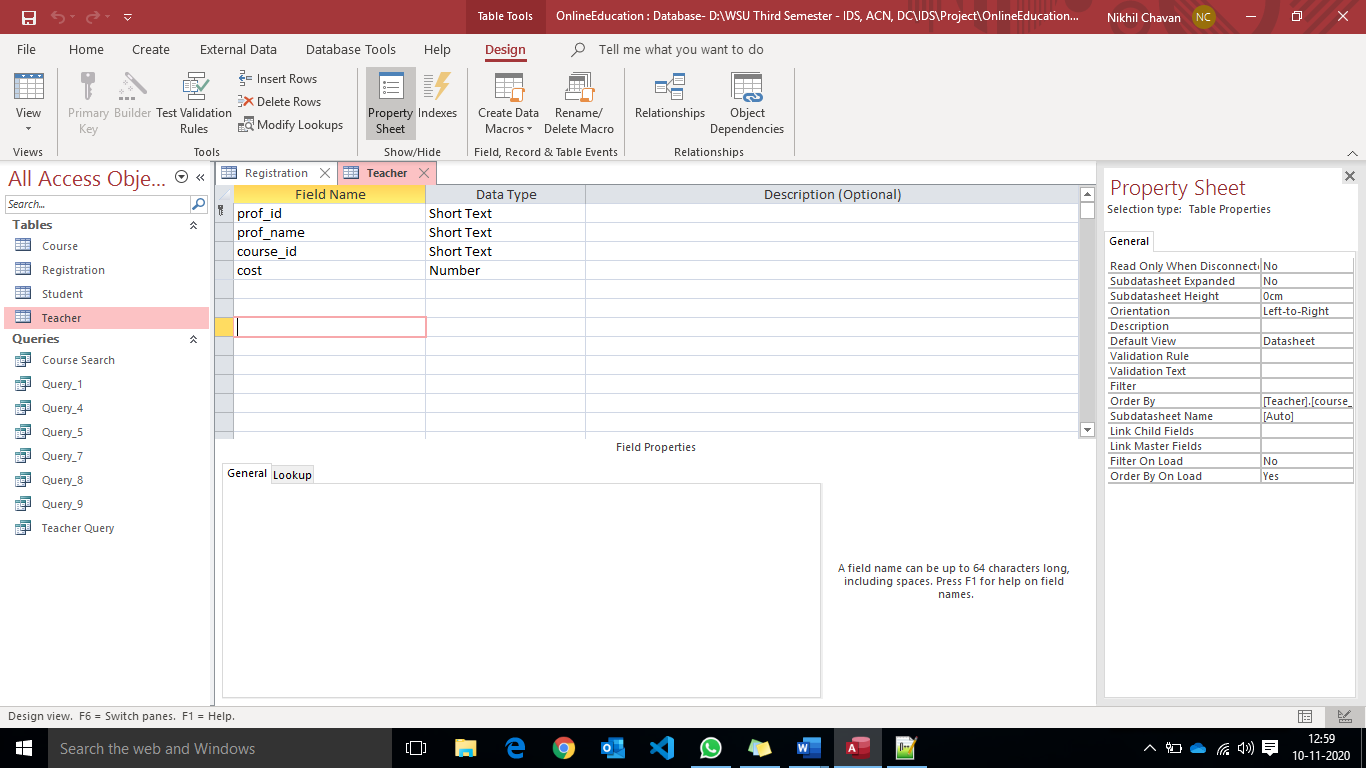
**Database tables:** Student, Course, Teacher, Registration.

**Table Schema:**

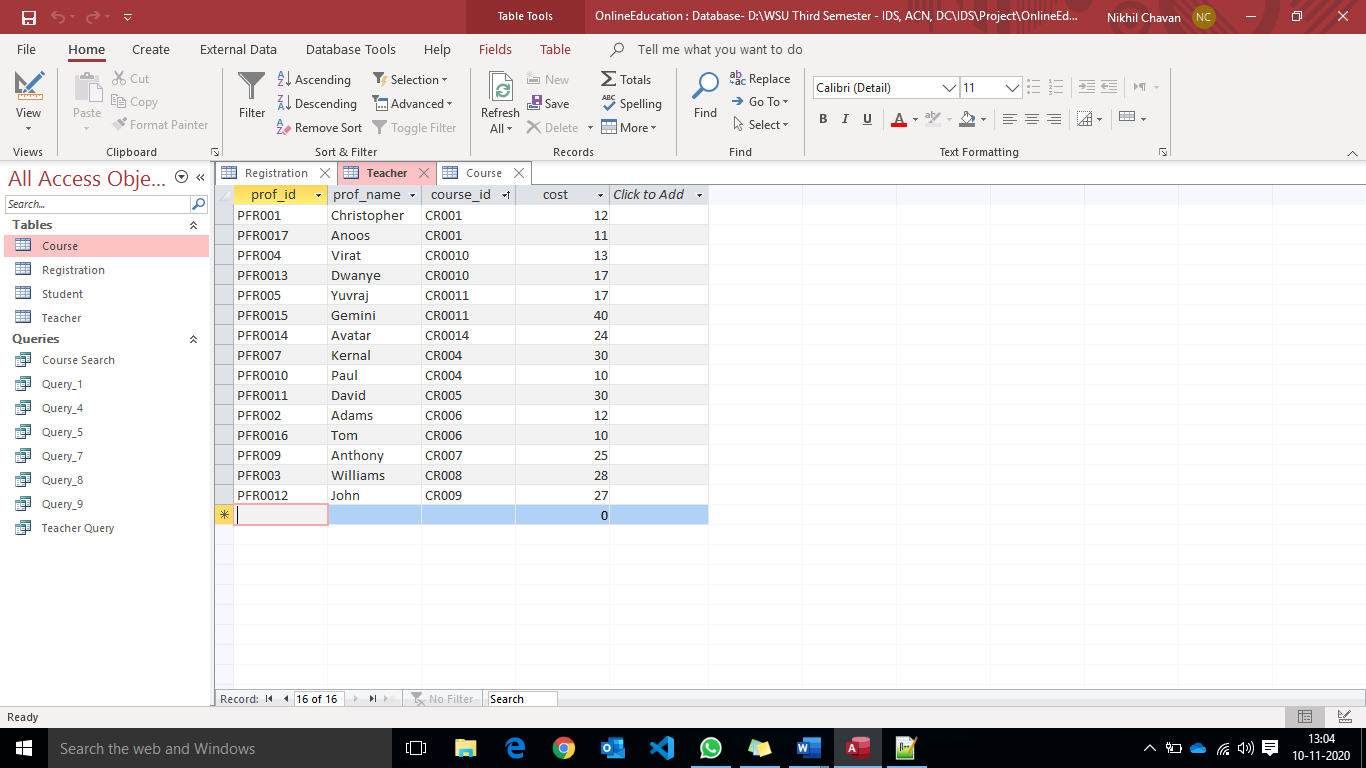
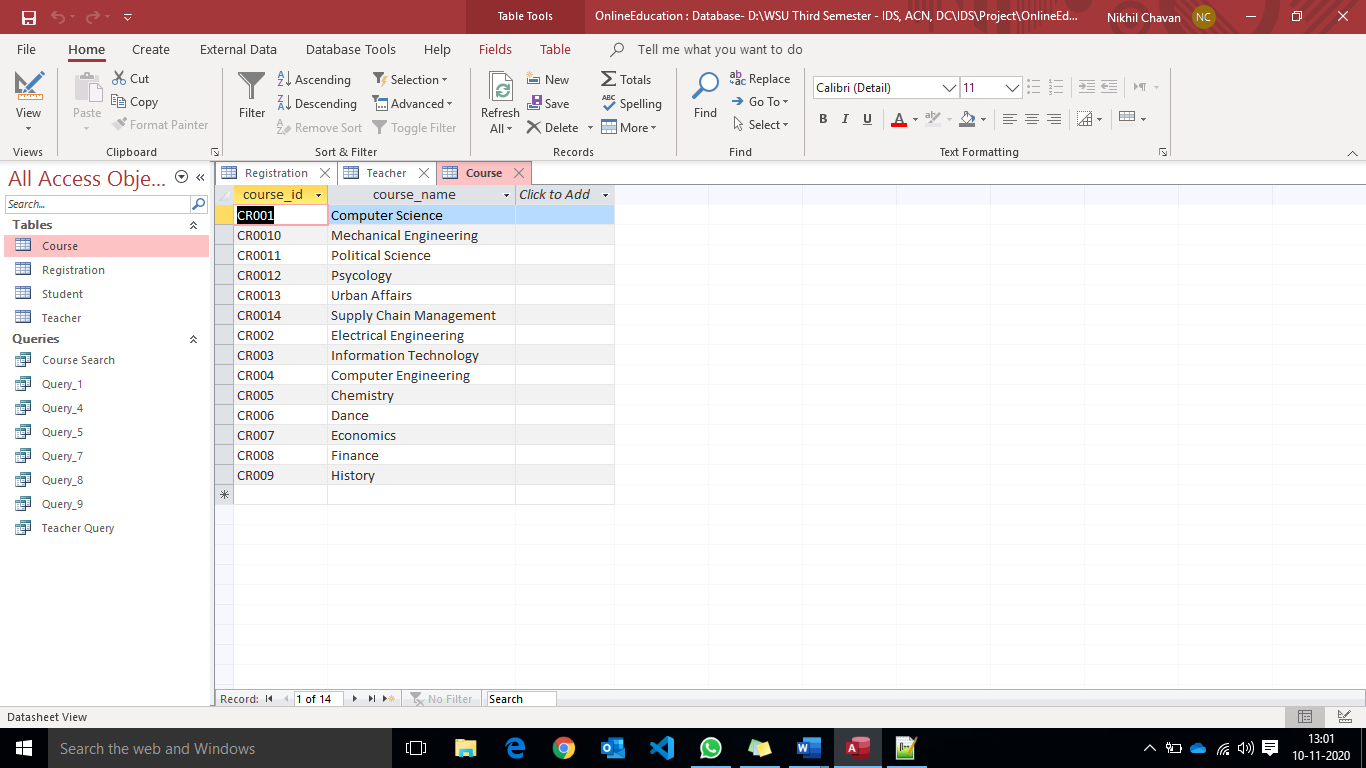
* **Student:** student\_id, name, address, phone, email.
* **Course:** course\_id, course\_name.
* **Teacher:** prof\_id, prof\_name, couse\_id, cost.
* **Registration:** booking\_id, student\_id, prof\_id, cost.

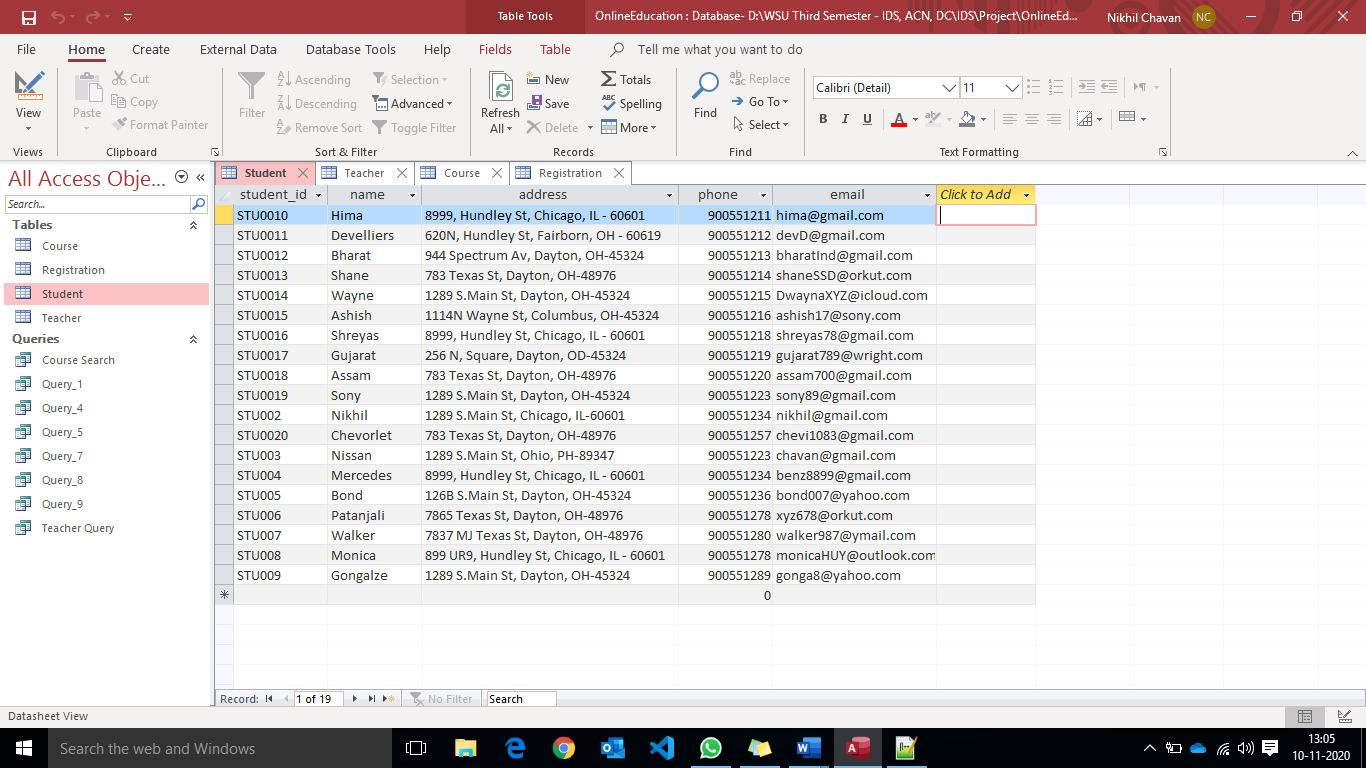
Below screenshots are the schema for the tables used in the project development.

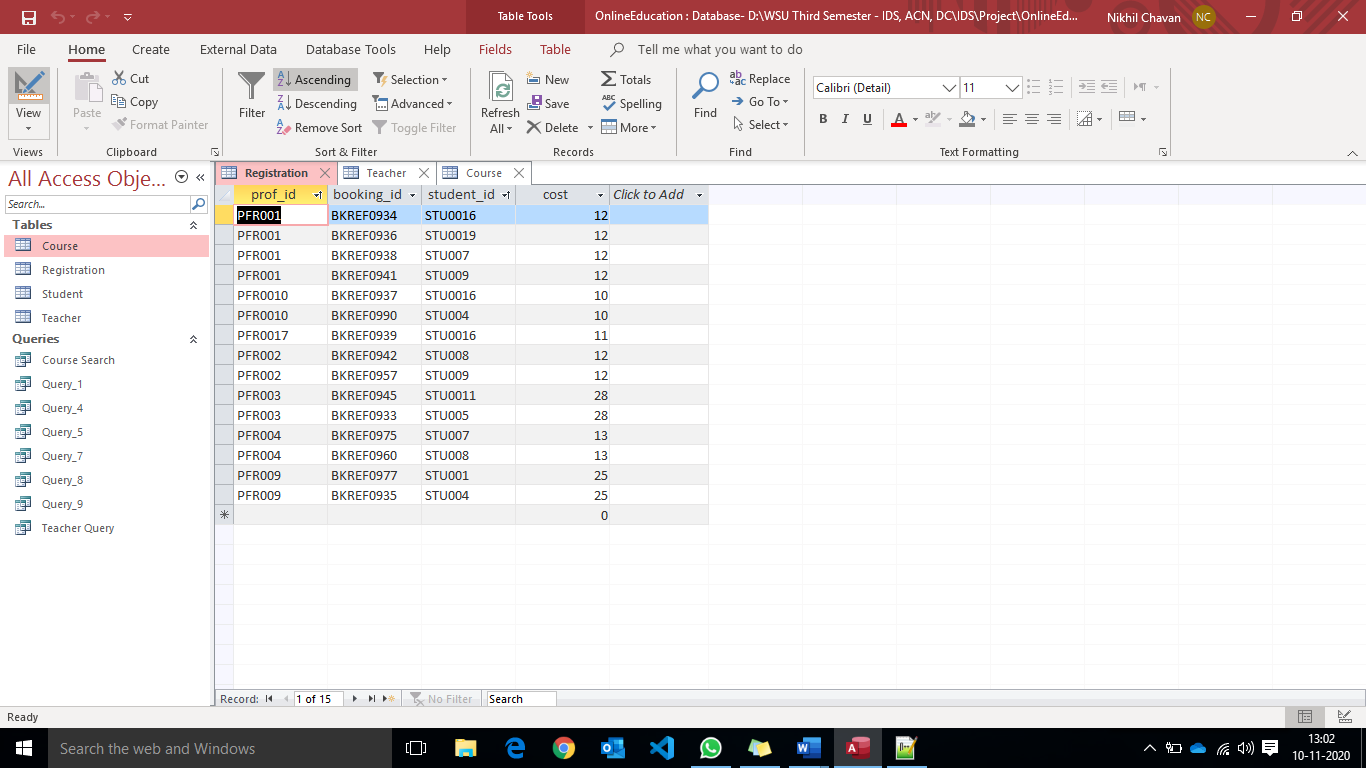
 

**Contents in the Tables: (**Table names are highlighted in **RED** in all the below screenshots**)**





**Project Transaction Description:**

* A Student could search for a course according to the course, cost, and professor. If a student finds the desired course, he proceeds with the enrolling of the course which includes providing the personal information like name, address, location, and phone number. Later the student pays for the enrolled course and the same information is updated in the database.
* A student could enrol in different courses at different professors, no two courses enrolled by a student will conflict for a successful booking.
* A course could be available all the time but under different professors.
* Different professors could not teach the same course at the same time.

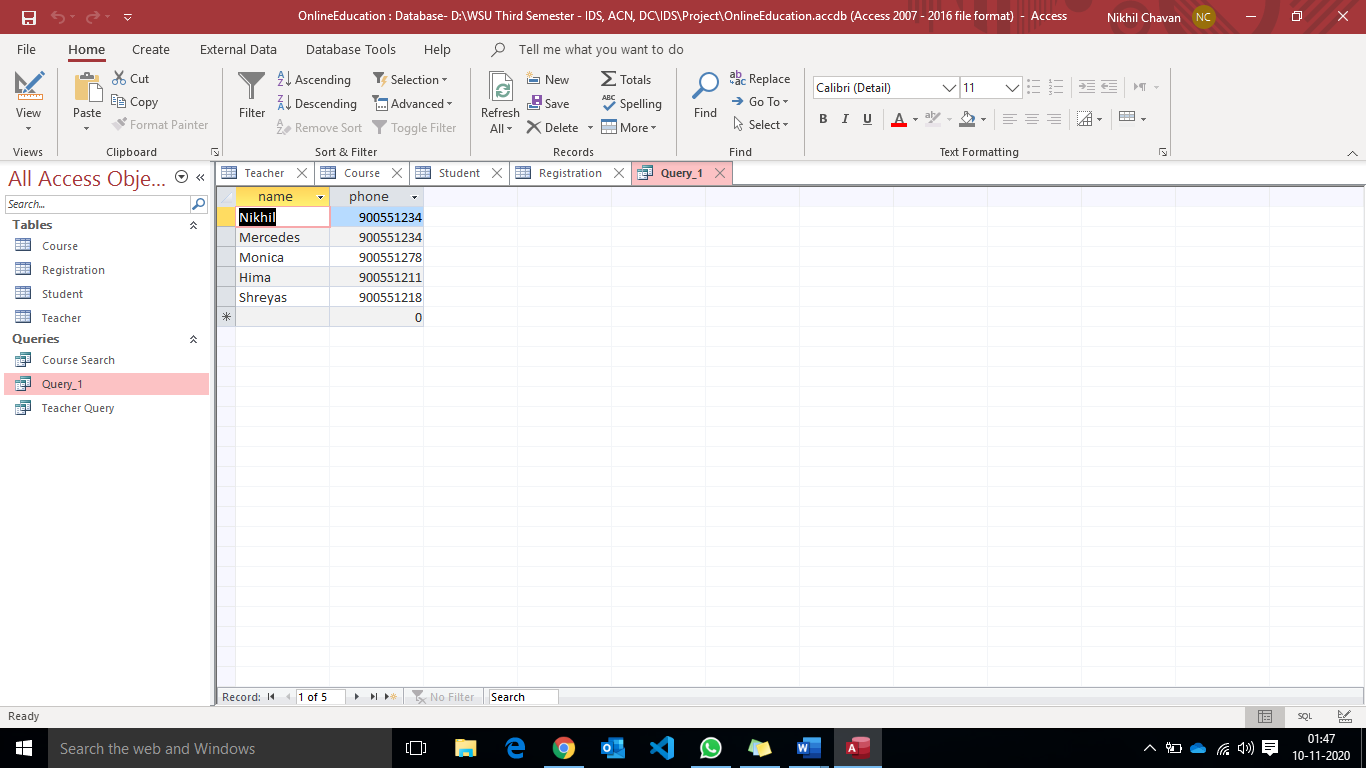
**Screenshots of the MS Access SQL Queries according to the Functionalities:**

1. **Retrieve Students details who live in Chicago**

SELECT name, phone

FROM Student

WHERE address like '\*Chicago\*';



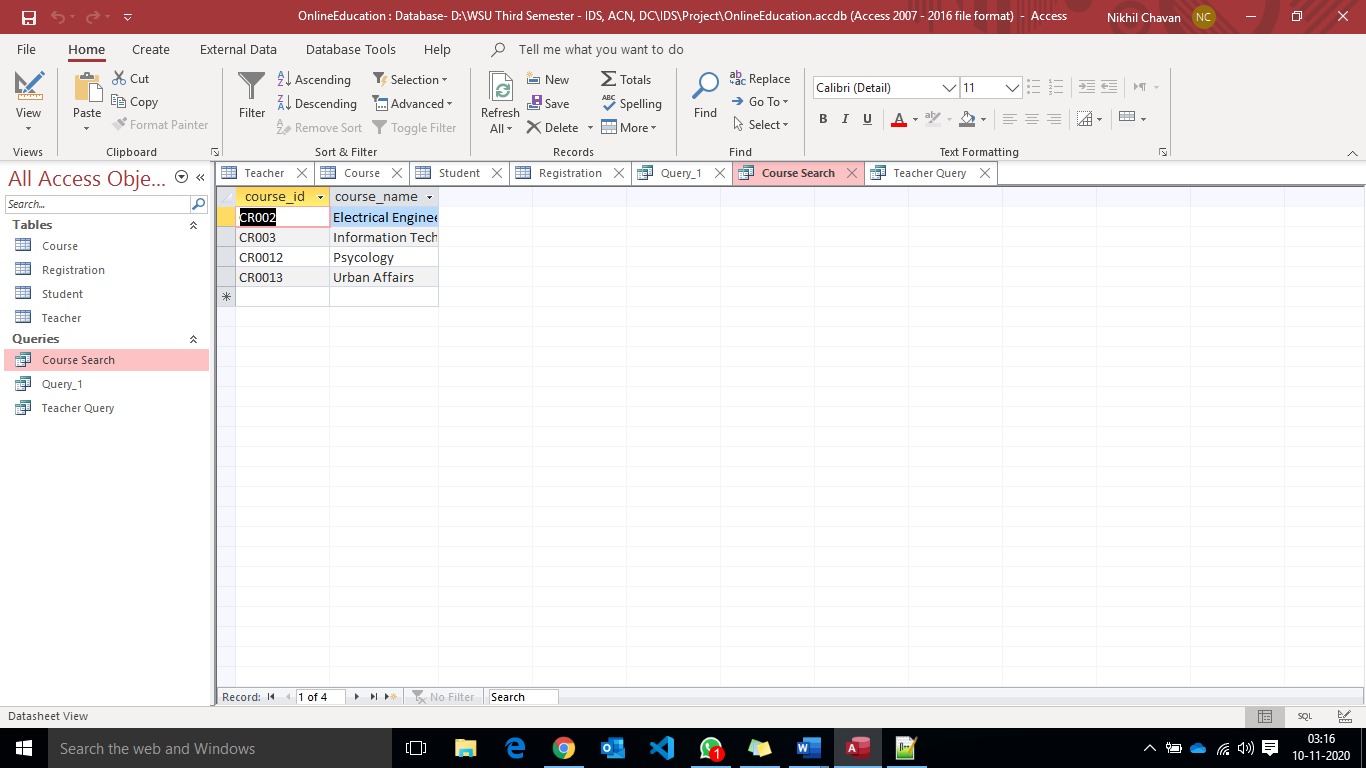
1. **List courses which are not linked to any professors**

SELECT C.course\_id,C.course\_name

FROM Course AS C

WHERE C.course\_id NOT IN

(SELECT DISTINCT(T.course\_id) from Teacher AS T)

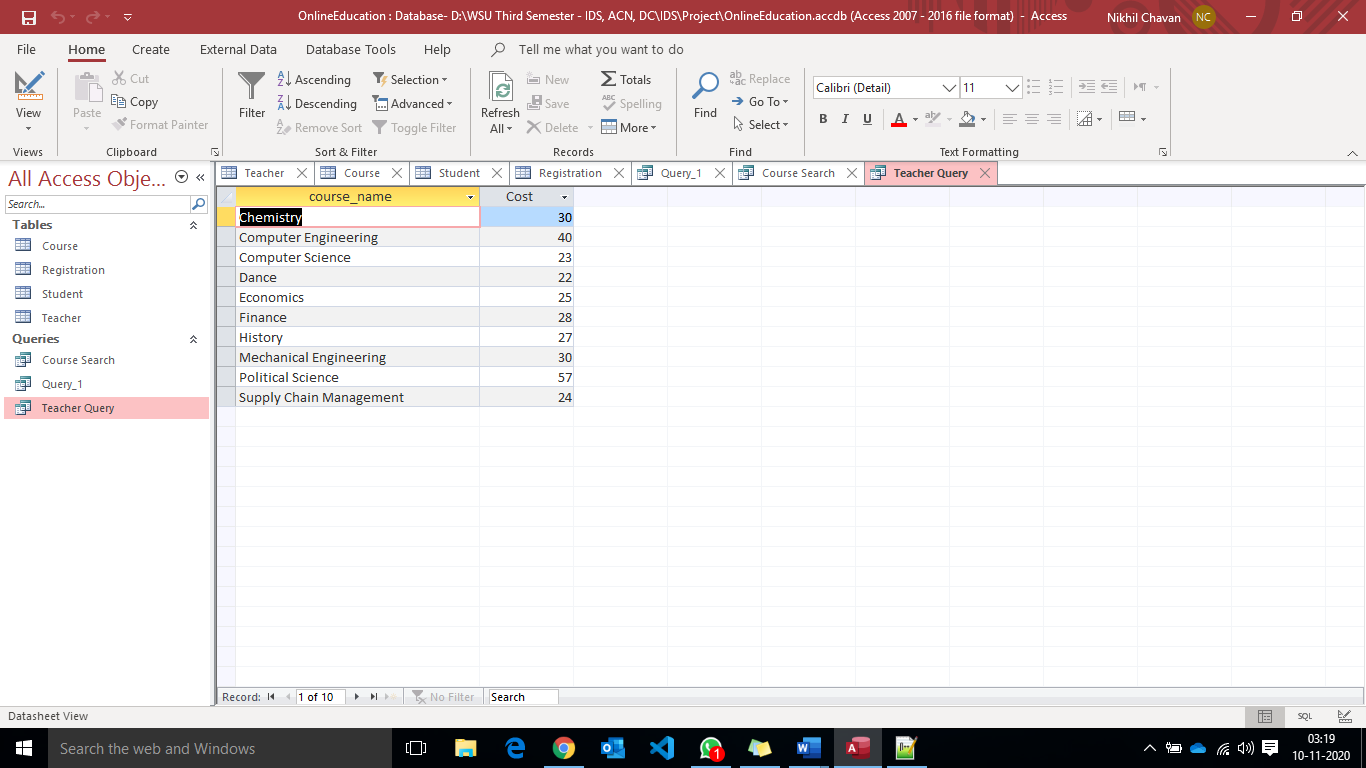


1. **List of professors group by course name which are present in the course table**

SELECT C.course\_name, Sum(T.cost) AS Cost

FROM Course AS C INNER JOIN Teacher AS T ON C.course\_id = T.course\_id

GROUP BY C.course\_name;



In order to obtain the above results, the aggregate function SUM is used to retrieve the course name present in the Course table by joining the Teacher table in order to get the course name associated with proferssor.

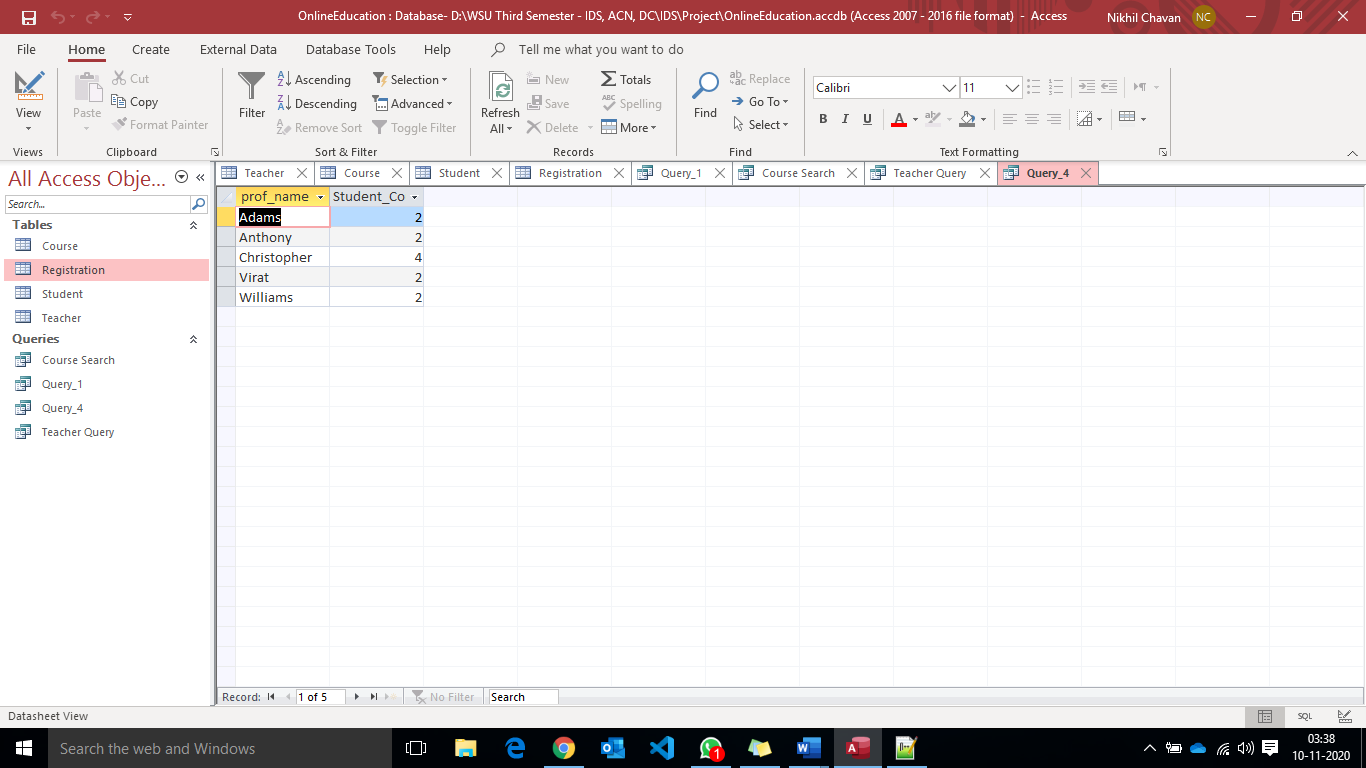
1. **Number of students count under the professor greater than 2**

SELECT T.prof\_name, count(R.student\_id) AS Student\_Count

FROM Registration AS R INNER JOIN Teacher AS T ON R.prof\_id = T.prof\_id

GROUP BY T.prof\_name

HAVING count(R.student\_id) > 1;



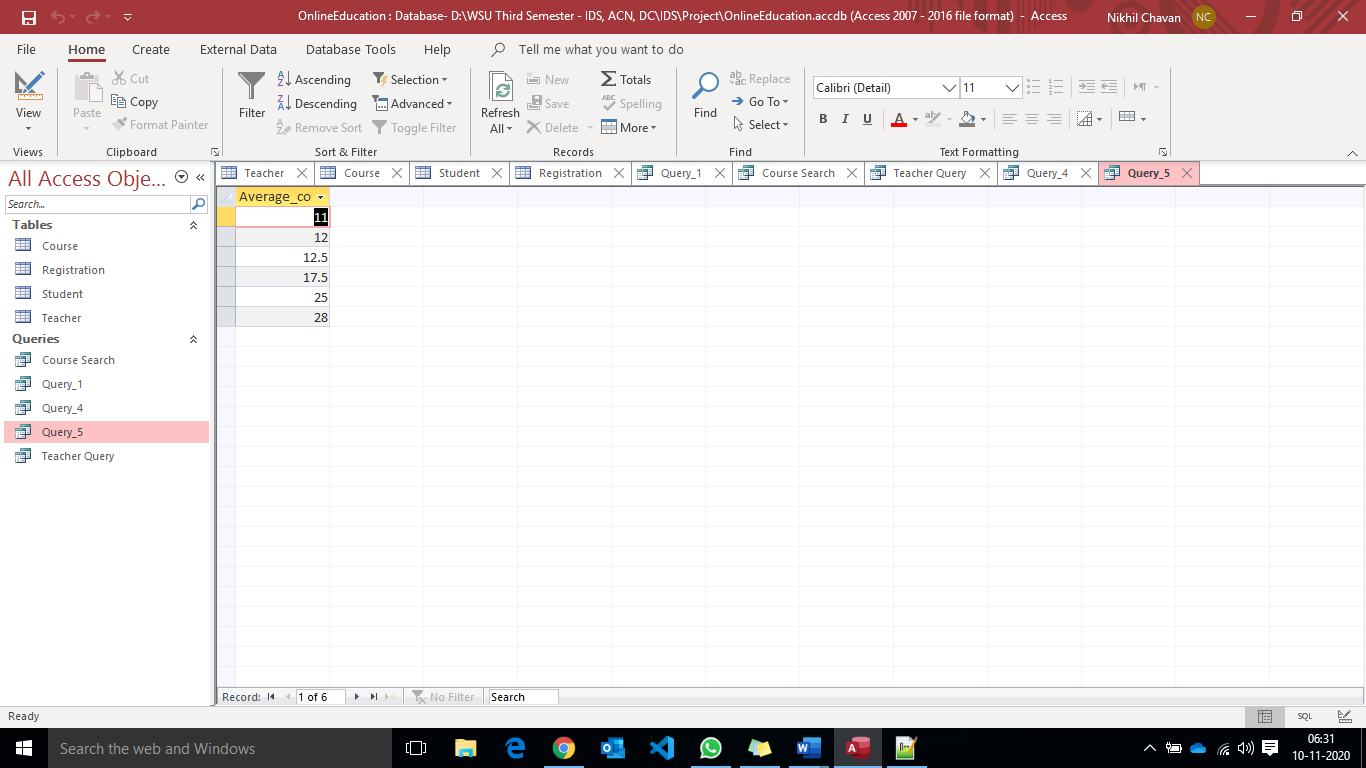
To retrieve the professors with the students under him/her are greater than 2 we need to combine Registration table and the Teacher table and apply HAVING clause to provide the count condition based on the students registered under a professor.

1. **Retrieve Average cost of the course from the professor table for specific course.**

SELECT DISTINCT AVG(R.cost) AS Average\_cost

FROM Registration AS R

GROUP BY R.student\_id;



The AVG aggregate function is used to produce the average cost of the student who registered for the different courses and DISTINCT function to remove the duplicated values over the averages for further analysis.

1. **List all the professors whose cost is greater than all the professors in “Mechanical Engineering”**

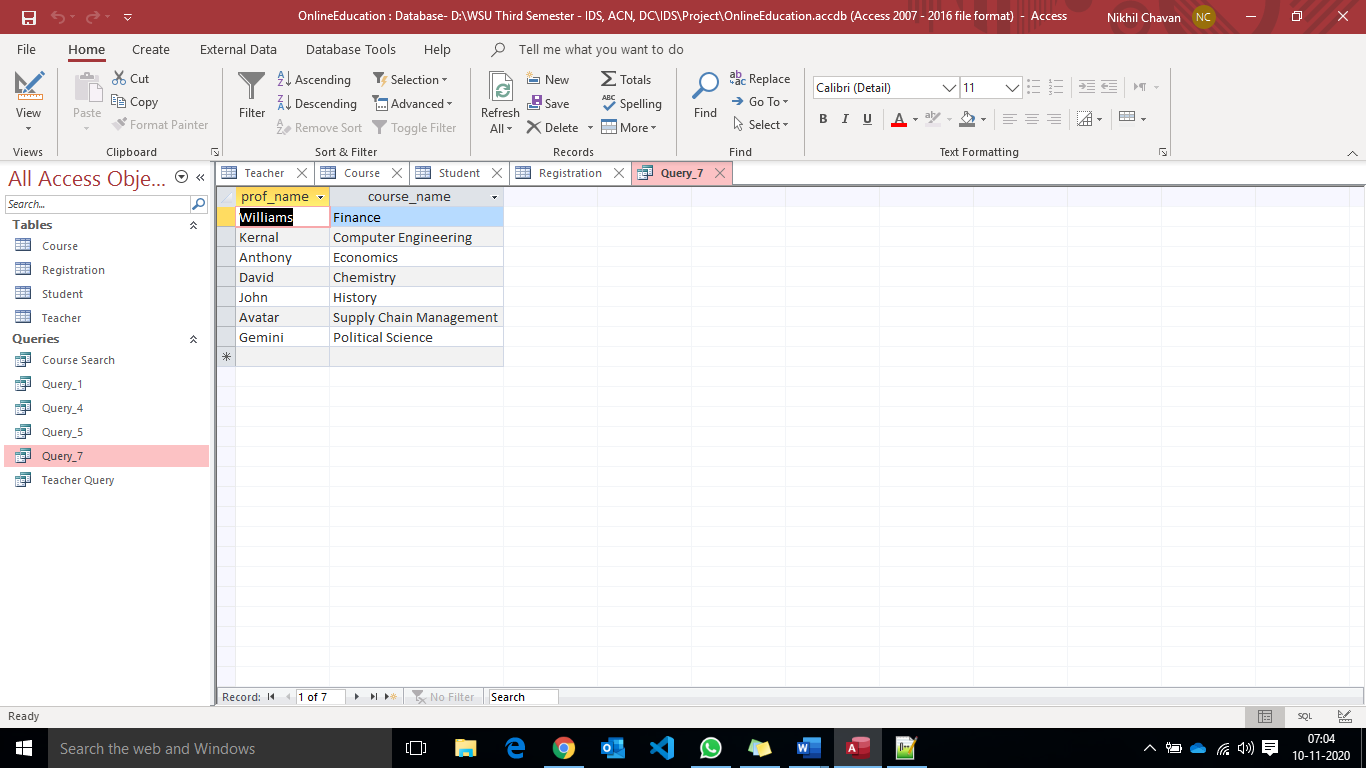
SELECT T.prof\_name, C.course\_name

FROM Teacher AS T INNER JOIN Course AS C ON T.course\_id = C.course\_id WHERE cost >ALL (

SELECT T.cost

FROM Teacher AS T INNER JOIN Course AS C ON T.course\_id = C.course\_id

WHERE C.course\_name = 'Mechanical Engineering');



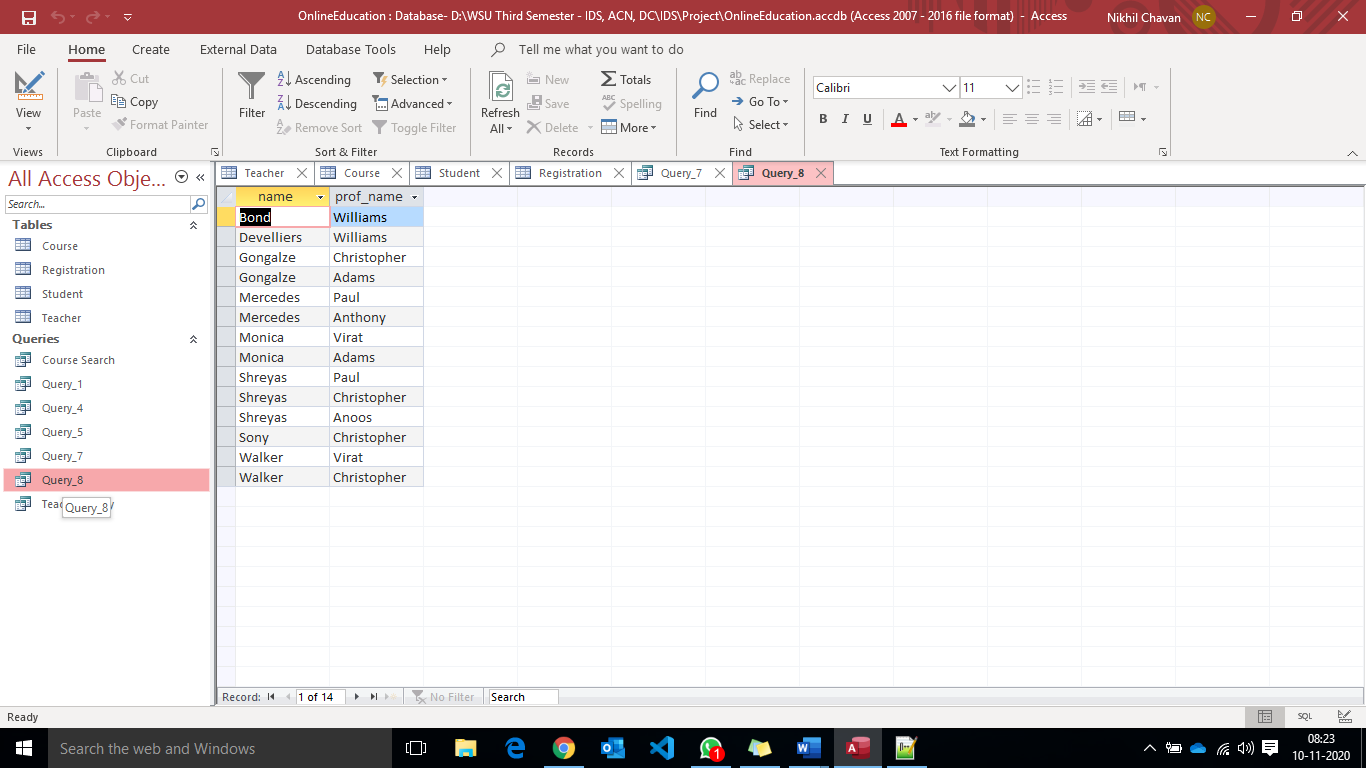
1. **List of the students enrolled in the course in ascending order and their respective professors in the descending order.**

SELECT S.name, T.prof\_name

FROM Student S, Teacher T, Registration R

WHERE R.student\_id = S.student\_id AND R.prof\_id = T.prof\_id

ORDER BY S.name, T.prof\_name DESC;



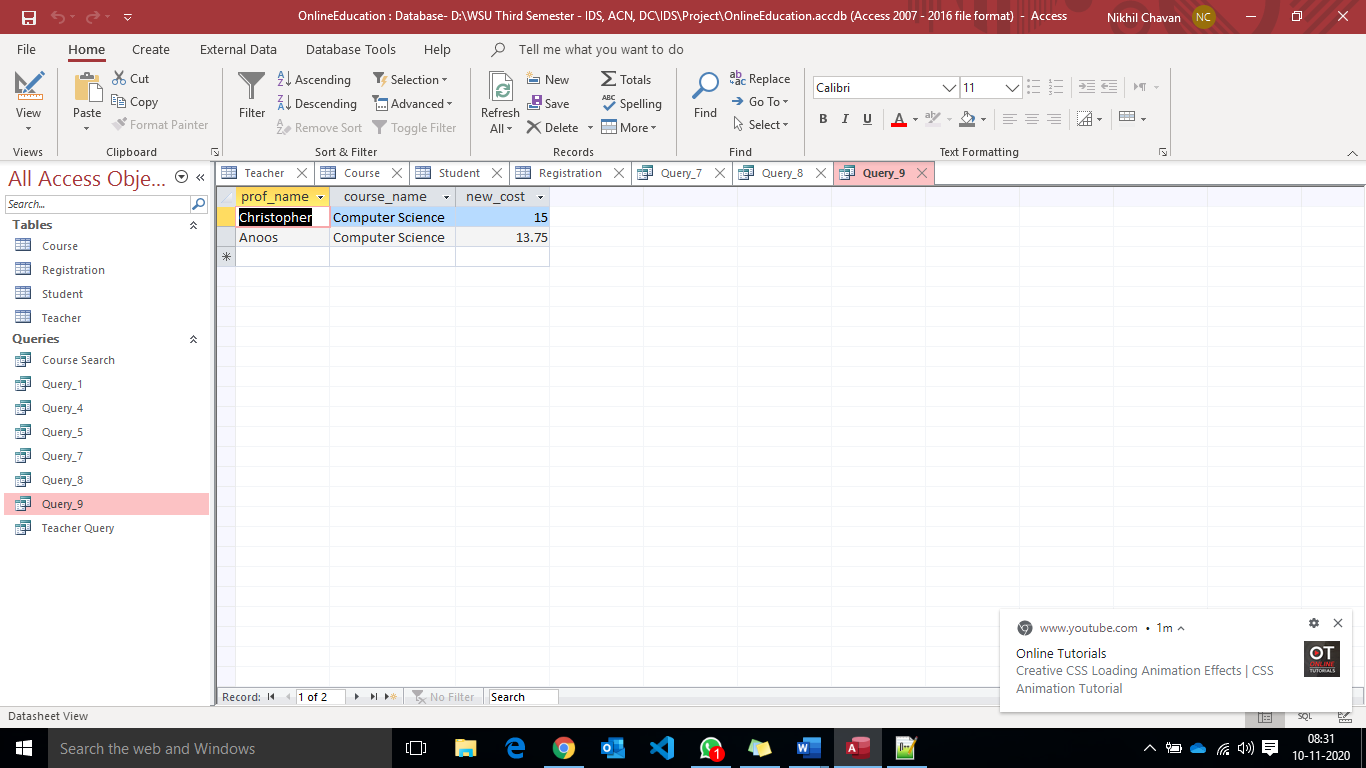
The ORDER BY keyword is used to sort the Student names in ascending order thereby DESC keyword to reverse the order the professors name alphabetically with the associated students registered under them.

1. **New hike for the professor in “Computer Science” course, cost by 25%**

SELECT T.prof\_name, C.course\_name, 1.25\*T.cost AS new\_cost

FROM Teacher AS T INNER JOIN Course AS C ON T.course\_id = C.course\_id

WHERE C.course\_name = 'Computer Science';



The cost refers to the professor cost for the course. To increase a cost of the Computer science professors by 25% we simply do cost ( 1 + (25/100) ) = 1.25\*cost

**Discussion:**

The project made me understand the task of identifying the requirements was the most basic and high priority task for any project development, once the requirements are locked coming up with the database design was another important task because the isolation, atomicity, consistent read and writes and Durability features should be taken care. In this project the use of aggregate functions, getting results by joining the tables for scenarios, identify the null case scenarios, the proper use of the Boolean values are key features that were taken care of always. I’ve also gained knowledge on using the Nested queries, EXISTS and IN operators, the difference between having and where which are indeed used in the project. The initial idea had changed just to address a few edge cases because of which the CRUD operations of the database could be efficient. The limitations of Int, Long Int, Number, the automatic id generator is well understood

There might be few scenarios which could make this project even better like the login functionality, dealing with the payment options, adding the different professor's attributes, querying on professors attributes, allowing the student to access the course in offline mode, creating a live discussion forum to address the doubts by the student in either offline and online mode, notification or the class material option for the students which requires different tables, redirecting the students all the related content different roles based on the authentication for different user scenarios, implementing the security features, checking for database leaks, identifying the proper database to store the student activity while studying the course online i.e. either to store it in SQL and NoSQL depending upon the higher reads or higher writes. Additionally with the use of Database tables when used with the server and client-side language could result in a live web page which could be deployed and used by the general public however the field of expansion always increases with the usage of the web applications based on which database updates must be done.

**Source Code:**

