University Of H-Town

[Student Database]

[Joseph Shoboiki, Esteban Mejia]

* **Abstract:**

University of H-Town is a university we have created. We also have created a database that will assist staff and students through the process of managing classes, managing grades, and managing schedules. For students, they will only be able to search through classes, add/drop classes and check current grades. The professors can search through students and change the student’s grades. The administrator can access and edit all the data including grades, addresses, credits, etc...

* **Mission Statement:**

The purpose of this database system is to maintain the data that is created and used by the university, to support the whole university system that focuses on students, classes, professors, advisors, credits and locations. To create ease of access for both students and faculty.

* **Mission Objectives:**

To maintain (enter, update, and delete) data on students.

To maintain (enter, update, and delete) data on professors.

To maintain (enter, update, and delete) data on courses.

To maintain (enter, update, and delete) data on credits.

To maintain (enter, update, and delete) data on advisors.

To maintain (enter, update, and delete) data on classrooms.

To maintain (enter, update, and delete) data on departments.

To perform searches on students.

To perform searches on professors.

To perform searches on courses.

To perform searches on classes.

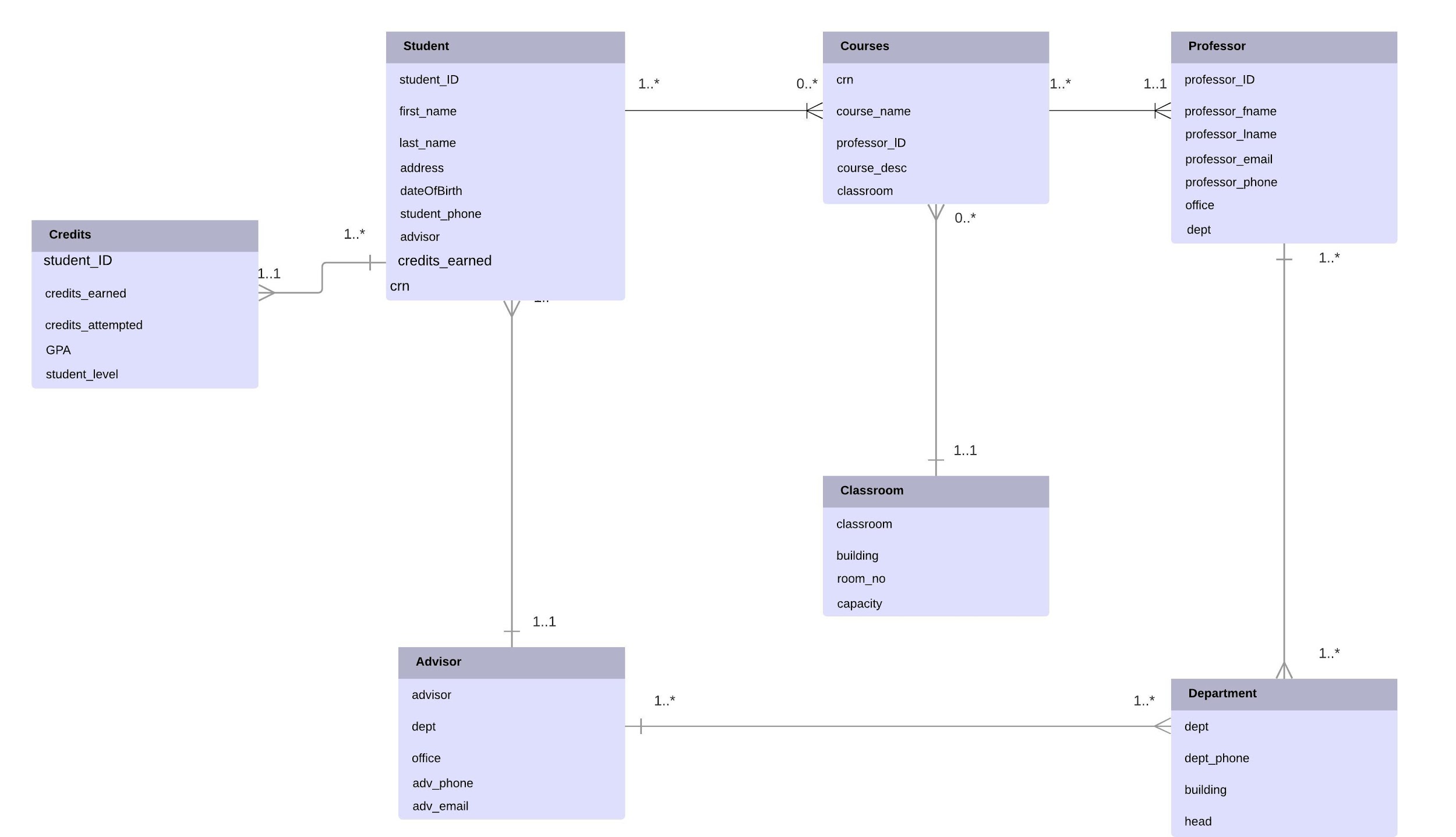
To perform searches on building.

To perform searches on advisors.

* **Major User Views:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Access Type | Students | Staff(Professor) | Administrator |
| Students | Manages  Queries | X | X | X |
| Professors | Manages  Queries | X | X | X |
| Courses | Manages  Queries | X | X | X |
| Credits | Manages  Queries | X | X | X |
| Classrooms | Manages  Queries | X | X | X |
| Building | Manages  Queries | X | X | X |
| Advisor | Manages  Queries | X | X | X |

* **ER Diagram:**



* **Relational Model:**

Student table is related to the advisor table due to the foreign Key (advisor):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| students\_ID (PK) | first\_name | last\_name | address | dateOfBirth | student\_phone | advisor | credits\_earned | crn |
| 123456 | Esteban | Mejia | 123 Fake St | 07/26/1994 | 123456789 | Jimbo | 122 | 2323 |
| 181818 | Richard | Richy | 4567 Bluebonnet St | 04/02/1996 | 986532121 | Jane | 22 | 3333 |
| 523698 | Michael | Scott | 2211 Dunder St. | 02/04/1988 | 369263632 | Jane | 45 | 3333 |
| 546783 | Jane | Benitez | 100 Main St. | 05/26/1984 | 789563211 | Jimbo | 120 | 7931 |
| 556643 | Bert | Osborne | 5659 Evergreen St | 12/25/2001 | 212178521 | Joe | 66 | 9898 |
| 568921 | Kenneth | McDonald | 3659 Tidwell St | 05/30/1999 |  | James | 100 | 2323 |
| 786512 | Andy | Carter | 7656 Sherman St. | 01/01/1991 | 562332698 | James | 0 | 3333 |
| 788552 | Jamie | Lanister | 123 Casterly Rd | 02/03/1975 | 232336665 | Joe | 66 | 7931 |
| 789887 | Joseph | Shoboiki | 456 Red Ln | 08/12/1993 | 987654321 | James | 126 | 9898 |
| 898989 | Claudia | Mejia | 54678 Green Tree Rd. | 07/08/1994 | 333666998 | Jimbo | 120 | 9898 |
| 999999 | Aubrey | James | 7899 Coyle St. | 01/21/1988 |  | Jane | 120 | 2323 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| advisor | dept | office | adv\_phone | adv\_email |
| James | CJ | NM456 | 986325622 | james@uni.edu |
| Jane | CS | A789 | 787894939 | jane@uni.edu |
| Jimbo | English | S989 | 885566331 | jimbo@uni.edu |
| Joe | CS | N999 | 865326965 | joe@uni.edu |
| Ron | CS | A777 | 7778889999 | ron@uni.edu |

Advisor table is related to Department table due to the foreign key dept:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| advisor | dept | office | adv\_phone | adv\_email |
| James | CJ | NM456 | 986325622 | james@uni.edu |
| Jane | CS | A789 | 787894939 | jane@uni.edu |
| Jimbo | English | S989 | 885566331 | jimbo@uni.edu |
| Joe | CS | N999 | 865326965 | joe@uni.edu |
| Ron | CS | A777 | 7778889999 | ron@uni.edu |

|  |  |  |  |
| --- | --- | --- | --- |
| dept | dept\_phone | building | head |
| Accounting | 9890003456 | North | Red |
| CJ | 895691193 | East | Jones |
| CS | 1236544654 | ACAD | Lin |
| English | 986632123 | South | Smith |
| Math | 562398655 | North | Wilder |

Student Table is related to the Course table because crn is a foreign key in the student table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| students\_ID (PK) | first\_name | last\_name | address | dateOfBirth | student\_phone | advisor | credits\_earned | crn |
| 123456 | Esteban | Mejia | 123 Fake St | 07/26/1994 | 123456789 | Jimbo | 122 | 2323 |
| 181818 | Richard | Richy | 4567 Bluebonnet St | 04/02/1996 | 986532121 | Jane | 22 | 3333 |
| 523698 | Michael | Scott | 2211 Dunder St. | 02/04/1988 | 369263632 | Jane | 45 | 3333 |
| 546783 | Jane | Benitez | 100 Main St. | 05/26/1984 | 789563211 | Jimbo | 120 | 7931 |
| 556643 | Bert | Osborne | 5659 Evergreen St | 12/25/2001 | 212178521 | Joe | 66 | 9898 |
| 568921 | Kenneth | McDonald | 3659 Tidwell St | 05/30/1999 |  | James | 100 | 2323 |
| 786512 | Andy | Carter | 7656 Sherman St. | 01/01/1991 | 562332698 | James | 0 | 3333 |
| 788552 | Jamie | Lanister | 123 Casterly Rd | 02/03/1975 | 232336665 | Joe | 66 | 7931 |
| 789887 | Joseph | Shoboiki | 456 Red Ln | 08/12/1993 | 987654321 | James | 126 | 9898 |
| 898989 | Claudia | Mejia | 54678 Green Tree Rd. | 07/08/1994 | 333666998 | Jimbo | 120 | 9898 |
| 999999 | Aubrey | James | 7899 Coyle St. | 01/21/1988 |  | Jane | 120 | 2323 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| crn | course\_name | professor\_ID | course\_desc | classroom |
| 2323 | CJ 1301 | 9119 | Law I | E653 |
| 3333 | MATH 1301 | 23232 | Algebra II | N353 |
| 5555 | CS 1300 | 12313 | Visual Basic | A323 |
| 7931 | ENG 2301 | 63956 | English III | S786 |
| 9898 | CS 3319 | 12313 | Intro to CS basics | A323 |

Course table is related to the professor table because professor id is a foreign key in the course table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| crn | course\_name | professor\_ID | course\_desc | classroom |
| 2323 | CJ 1301 | 9119 | Law I | E653 |
| 3333 | MATH 1301 | 23232 | Algebra II | N353 |
| 5555 | CS 1300 | 12313 | Visual Basic | A323 |
| 7931 | ENG 2301 | 63956 | English III | S786 |
| 9898 | CS 3319 | 12313 | Intro to CS basics | A323 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| professor\_ID | professor\_fname | professor\_lname | professor\_email | Professor\_phone | office | dept |
| 12313 | Daniel | Ocean | daniel@uni.edu | 8956232 | A111 | CS |
| 23232 | Ricky | Road | rick@uni.edu | 78965884 | N765 | Math |
| 63956 | Justin | Watson | watson@uni.edu | 36364456 | S222 | English |
| 67898 | Reed | Brandon | reed@uni.edu | 9898887898 | S345 | CS |
| 9119 | Brian | Garcia | brian@uni.edu | 9119119 | E911 | CJ |

Professor table is related to the department table due to the foreign key dept:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| professor\_ID | professor\_fname | professor\_lname | professor\_email | Professor\_phone | office | dept |
| 12313 | Daniel | Ocean | daniel@uni.edu | 8956232 | A111 | CS |
| 23232 | Ricky | Road | rick@uni.edu | 78965884 | N765 | Math |
| 63956 | Justin | Watson | watson@uni.edu | 36364456 | S222 | English |
| 67898 | Reed | Brandon | reed@uni.edu | 9898887898 | S345 | CS |
| 9119 | Brian | Garcia | brian@uni.edu | 9119119 | E911 | CJ |

|  |  |  |  |
| --- | --- | --- | --- |
| dept | dept\_phone | building | head |
| Accounting | 9890003456 | North | Red |
| CJ | 895691193 | East | Jones |
| CS | 1236544654 | ACAD | Lin |
| English | 986632123 | South | Smith |
| Math | 562398655 | North | Wilder |

The course table is related to the classroom table because of the foreign key classroom:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| crn | course\_name | professor\_ID | course\_desc | classroom |
| 2323 | CJ 1301 | 9119 | Law I | E653 |
| 3333 | MATH 1301 | 23232 | Algebra II | N353 |
| 5555 | CS 1300 | 12313 | Visual Basic | A323 |
| 7931 | ENG 2301 | 63956 | English III | S786 |
| 9898 | CS 3319 | 12313 | Intro to CS basics | A323 |

|  |  |  |
| --- | --- | --- |
| classroom | building | capacity |
| A323 | ACAD | 60 |
| A999 | ACAD | 100 |
| E653 | East | 35 |
| N353 | North | 25 |
| S786 | South | 25 |

Student table is related to the Credits table due to the student\_ID foreign key in the credits table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| students\_ID (PK) | first\_name | last\_name | address | dateOfBirth | student\_phone | advisor | credits\_earned | crn |
| 123456 | Esteban | Mejia | 123 Fake St | 07/26/1994 | 123456789 | Jimbo | 122 | 2323 |
| 181818 | Richard | Richy | 4567 Bluebonnet St | 04/02/1996 | 986532121 | Jane | 22 | 3333 |
| 523698 | Michael | Scott | 2211 Dunder St. | 02/04/1988 | 369263632 | Jane | 45 | 3333 |
| 546783 | Jane | Benitez | 100 Main St. | 05/26/1984 | 789563211 | Jimbo | 120 | 7931 |
| 556643 | Bert | Osborne | 5659 Evergreen St | 12/25/2001 | 212178521 | Joe | 66 | 9898 |
| 568921 | Kenneth | McDonald | 3659 Tidwell St | 05/30/1999 |  | James | 100 | 2323 |
| 786512 | Andy | Carter | 7656 Sherman St. | 01/01/1991 | 562332698 | James | 0 | 3333 |
| 788552 | Jamie | Lanister | 123 Casterly Rd | 02/03/1975 | 232336665 | Joe | 66 | 7931 |
| 789887 | Joseph | Shoboiki | 456 Red Ln | 08/12/1993 | 987654321 | James | 126 | 9898 |
| 898989 | Claudia | Mejia | 54678 Green Tree Rd. | 07/08/1994 | 333666998 | Jimbo | 120 | 9898 |
| 999999 | Aubrey | James | 7899 Coyle St. | 01/21/1988 |  | Jane | 120 | 2323 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| credits\_earned | credits\_attempted | GPA | student\_level | student\_ID |
| 122 | 122 | 3.02 | Senior | 123456 |
| 22 | 22 | 3.8 | Freshman | 181818 |
| 45 | 47 | 2.6 | Sophomore | 523698 |
| 120 | 122 | 3.0 | Senior | 546783 |
| 66 | 66 | 4.0 | Junior | 556643 |
| 100 | 100 | 2.8 | Senior | 568921 |
| 0 | 0 | 0 | Freshman | 786512 |
| 66 | 66 | 3.8 | Junior | 788552 |
| 126 | 126 | 2.4 | Senior | 789887 |
| 120 | 120 | 4.0 | Senior | 898989 |

* ***Functional Dependency:***

Dependency comes in two varieties: Functional and transitive dependencies. We use these dependencies mainly in the normalization process. They aid in various steps of normalization throughout the data preprocessing stage of our project.

***Functional Dependency:*** is used to describe and model relationships between attributes which means that an occurrence within a certain attribute determines an occurrence in another attribute without ambiguity. This is split between the determinant and the dependent relationship which becomes important in determining which attribute serves best as primary key among other determinations. In figure 5.1 shows our instructor table with seven attributes to its name.

I like to start by evaluating the data sets for candidate keys, super keys and foreign keys. This help with dependencies understanding dependencies and checking for your primary key. From this step you map out dependencies in your attributes which allows you to confirm your keys and vice versa. Functional dependencies are as follows:

Instructor ID First name + last name

Instructor ID Email

Instructor ID office

Instructor ID Department

First name + Last name Email

First name + Last name office

First name + last name instructor ID

First name + last name phone no

First name + last name Department

Department First name + last name

Department Email

Department office

Department phone no

After Ascertaining these dependencies we can begin the process of normalization but we also need to ascertain transitive dependency. A transient in our tables are as follows

Instructor ID First name + last name Email

Instructor ID First name + last name phone no

* ***Normalization:***

Taking the functional dependency data from the previous section we begin the process of normalization. This process reduces data redundancy, create and ensure relational dependency of data so update anomalies can be prevented. We begin by getting our un-normalized form(UNF) data into first normal form(1NF). We do this by removing repeating groups and ensuring that each intersection of rows and columns contains one and only one value.

After this process is done we begin the process of transforming the data to the second normal form. We do this by removing partial dependencies this is where functional dependencies come into play. Firstly, we identify the primary key, candidate keys, and functional dependencies. The primary key in our table is the Instructor ID. It is unique to the data set and can be used as an identifying attribute to pull out information on the instructor table guarantying good functional dependency in table. The candidate keys in our Table is plentiful in our table which is good. We have the unique combination of first and last names, email address. Then we look to the functional dependencies covered in previous section to guide us in removing partial dependencies the table has none, so we ascertain that this is our second normal form.

The process of transforming data from second normal form to third normal form depends on transitive dependency. To bring the table to the third normal all transitive dependencies that are removed. Hence, we ascertain that our table is in the third normal form.

Lastly, we bring our Table to the Boyce-Codd Normal Form (3.5NF / BCNF). We do this by ensuring that there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. This removes anomalies due to functional dependencies ensuring data redundancy problem is eliminated in table.

Fig 5.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instructor Table | LName | Email | Office(Location) | Instructor ID | Phone No | Department |
| Ping | Xun | Xun\_P@gmail.com | A607, 0MB | 100195 | 281-666-2278 | Computer Science |
| James | Marsden | Mardsden\_BigD@gmail.com | A805, 0MB | 100196 | 281-666-2381 | Electronics Engineering |
| Biggie | Smalls | Small\_BigD@live.com | C305, BB | 100198 | 281-666-2175 | Finance |
| Wiz | Khalifa | khalifa420@gmail.com | A807, 0MB | 100199 | 281-666-2398 | Electronics Engineering |
| Snoop | Dogg | Snoopdoggydog@gmail.com | A605, 0MB | 100102 | 281-666-2251 | Computer Science |
| Nate | Dogg | Nate\_dog@gmail.com | C405, BB | 100105 | 281-666-2799 | Accounting |
| Ice | Cube | Cube\_d\_boss@gmail.com | N405, 0MB | 100110 | 281-666-2066 | English |
| Logic | Henderson | Logic\_god@gmail.com | S607, 0MB | 100295 | 281-666-2676 | Mathematics |
| Bori | Thorson | Bori\_Thor@gmail.com | S405, 0MB | 100395 | 281-666-2912 | History |

Primary Key = Instructor ID

Candidate keys = Email

First name +last name

Phone no

Office

Fig 5.2

Instructor Table

BCNF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instructor ID | fName | LName | Email | Office(Location) | Phone No | Department |
| 100195 | Ping | Xun | [Xun\_P@gmail.com](mailto:Xun_P@gmail.com) | A607, 0MB | 281-666-2278 | Computer Science |
| 100196 | James | Marsden | [Mardsden\_Big@gmail.com](mailto:Mardsden_BigD@gmail.com) | A805, 0MB | 281-666-2381 | Electronics Engineering |
| 100198 | Biggie | Smalls | [Small\_Big@live.com](mailto:Small_BigD@live.com) | C305, BB | 281-666-2175 | Finance |
| 100199 | Wiz | Khalifa | [khalifa420@gmail.com](mailto:khalifa420@gmail.com) | A807, 0MB | 281-666-2398 | Electronics Engineering |
| 100102 | Snoop | Dogg | [Snoopdoggydog@gmail.com](mailto:Snoopdoggydog@gmail.com) | A605, 0MB | 281-666-2251 | Computer Science |
| 100105 | Nate | Dogg | [Nate\_dog@gmail.com](mailto:Nate_dog@gmail.com) | C405, BB | 281-666-2799 | Accounting |
| 100110 | Ice | Cube | [Cube\_d\_boss@gmail.com](mailto:Cube_d_boss@gmail.com) | N405, 0MB | 281-666-2066 | English |
| 100295 | Logic | Henderson | [Logic\_god@gmail.com](mailto:Logic_god@gmail.com) | S607, 0MB | 281-666-2676 | Mathematics |
| 100395 | Bori | Thorson | [Bori\_Thor@gmail.com](mailto:Bori_Thor@gmail.com) | S405, 0MB | 281-666-2912 | History |

Fig 5.3

BCNF Student Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FName | LName | STD ID | Address | DOB | Major | Phone No | Advisor | Credits Earned |
| John | Sho | 200101 | 2740 Johnny Avenue | 10/4/1998 | Computer Science | 281-905-8500 | James Coney | 75 |
| Jay | Ray | 200102 | 1908 foot Dr. | 9/10/1997 | Electronics Engineering | 281-455-2675 | Veruca Joe | 57 |
| Carl | Scolera | 200104 | 1234 JohnFitz Dr. | 4/21/1996 | Accounting | 281-465-2178 | Jane Vargas | 75 |
| Drake | Juggz | 200199 | 2124 Maclovoin Avenue | 3/21/1997 | Finance | 281-407-2748 | Jimbo Smalls | 75 |
| Jack | Dickemz | 200190 | 1222 Badassery Dr. | 4/21/1999 | Computer Science | 335-251-7521 | James Coney | 25 |
| Dick | Tracy | 200180 | 3333 Satans Dr. | 3/21/1998 | Electronics Engineering | 555-281-5455 | Veruca Joe | 35 |
| Joe | Logan | 200167 | 6666 Perdition Dr | 2/22/1998 | Accounting | 555-541-5247 | Jane Vargas | 100 |
| Carl | Oslo | 200170 | 7568 JackHammer Dr. | 2/27/1994 | Computer Science | 281-407-0547 | James Coney | 105 |
| Jenny | Slater | 200165 | 969 Mission Dr. | 5/27/1995 | Electronics Engineering | 281-465-7548 | Veruca Joe | 25 |
| Naomi | Watt | 200111 | 989 Mission Dr. | 7/12/1996 | Accounting | 832-282-5547 | Jane Vargas | 45 |
| Nami | Watt | 200311 | 1006 foot Dr. | 8/21/1995 | Finance | 281-475-5555 | Jimbo Smalls | 65 |
| Crystal | Hernandez | 200411 | 500 Westheimer rd. | 5/13/1996 | Electronics Engineering | 281-475-5666 | Veruca Joe | 85 |
| Fantasia | Cosmo | 200176 | 11000 Westheimer rd. | 6/14/1995 | Computer Science | 281-666-4754 | James Coney | 100 |
| Aqua | Fina | 200179 | 11599 Jules Vern Avenue | 6/15/1995 | Computer Science | 281-656-4078 | James Coney | 57 |
| Anki | Xander | 200435 | 43 Perdition dr. | 6/15/1985 | Finance | 281-541-4755 | Jimbo Smalls | 35 |
| Pope | Jer | 200711 | 25 Jackson rd. | 7/21/1996 | Finance | 555-254-6666 | Jimbo Smalls | 85 |
| Pedo | Files | 200656 | 27 Hellroad av | 5/17/1990 | Accounting | 666-584-0022 | Jane Vargas | 54 |
| Nami | Korosawa | 200929 | 198 Trapalot Dr. | 4/5/1992 | Accounting | 666-284-0063 | Jane Vargas | 94 |
| Tupac | Shakur | 200911 | 200 Westside Avenue | 1/21/1996 | Electronics Engineering | 666-284-6500 | Veruca Joe | 95 |

Primary Key = Student ID

Candidate keys = Address

First name +last name

Phone no

Fig 5.4

Course Table BCNF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CRN | Course Name | Professor ID | Course Description | Classroom |
| 50591 | Math 2301 | 100295 | Calculus I | A635 |
| 50592 | Math 2302 | 100295 | Calculus II | A635 |
| 50597 | Math 2507 | 100295 | Discrete Math | A705 |
| 50587 | Math3305 | 100295 | Probability & Stat | A705 |
| 35053 | Engl 1305 | 100110 | Composition I | A635 |
| 35057 | Engl 1306 | 100110 | Composition II | A635 |
| 35089 | Engl 2307 | 100110 | Technical Writing | N950 |
| 30025 | Hist 2301 | 100395 | History to 1847 | N500 |
| 30027 | Hist 2302 | 100395 | History after 1847 | N700 |
| 30047 | Govt 2307 | 100395 | US Government | A635 |
| 60058 | CS 1408 | 100195 | Introduction to vBasics | S800 |
| 60123 | CS 2405 | 100195 | Advanced C++ | A635 |
| 60066 | CS 2401 | 100195 | Introduction to C++ | S600 |
| 60152 | CS 3305 | 100195 | Theory of Robotics | S201 |
| 60661 | CS 4305 | 100102 | Operating Systems | S625 |
| 60662 | CS 4321 | 100102 | Game Development | S600 |
| 60772 | CS 4307 | 100102 | Artificial Intelligence | S600 |
| 80231 | Engr 1301 | 100196 | Engineering Fundamentals I | N805 |
| 80235 | Engr 1302 | 100196 | Engineering Fundamentals II | N805 |
| 80251 | Engr 2401 | 100196 | CADD | N800 |
| 80952 | Engr 2402 | 100199 | Circuit Analysis | N800 |
| 80953 | Engr 3302 | 100199 | Digital Hardware | N800 |
| 80456 | Engr 3305 | 100199 | Digital Software | N800 |
| 20599 | Acc 1401 | 100105 | Introduction to accounting | A200 |
| 20559 | Acc 1408 | 100105 | Financial Accounting | A200 |
| 20659 | Acc 2401 | 100105 | Managerial Accounting | A200 |
| 20669 | Acc 3501 | 100105 | Tax Accounting | A200 |
| 20455 | Fin 3501 | 100198 | Business Finance | F200 |
| 20859 | Fin 2501 | 100198 | Small Business Finance | N200 |
| 20844 | Fin 4201 | 100198 | Corporate Finance | F200 |
| 20889 | Fin 4105 | 100198 | Real Estate Management | N200 |

Primary Key = CRN

Candidate keys = Course Name

Course Description

Classroom

* **Use Cases:**

Students

* Insert Query
  + Actor: Administrator
  + Steps:
    - User selects the “New Student” option
    - A new student id is generated and displayed
    - Prompt user to enter the student’s name, address, phone number and date of birth.
    - All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Students (student, first\_name, last\_name, address, dateOfBirth, student\_phone)

VALUES(969696, “John”, “Gomez”, “4824 Werner St.”, 09/11/1991, 7136457898);

* Delete Query
  + Actor: Administrator
  + Steps:
    - User selects the “Delete Student” option.
    - Prompt the user to enter the student id that they want to delete
    - All of the student’s information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Students

WHERE students\_id = 969696;

* Update Query
  + Actor: Administrator
  + Steps:
    - User selects the “Update Student’s information” option
    - Prompt the user to enter the student id that they would like to update
    - The students information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Students

SET address = “54678 Green Tree Rd.”

WHERE student\_ID = 898989;

* Aggregate Query
  + Actor: Administrator
  + Steps:
    - User selects “ Total amount of Students” option
    - Number of currently enrolled students is then displayed
  + SQL Code:

SELECT COUNT (student\_ID) from Students;

Professor:

* Insert Query
  + Actor: Administrator
  + Steps:
    - User selects the “New Professor” option
    - A new professor id is generated and displayed
    - Prompt user to enter the professor’s name, address, phone number, email, office and department.
    - All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Professor (professor\_ID, professor\_fname, professor\_lname, professor\_email, professor\_phone, office, dept)

VALUES(67898, “Reed”, “Brandon”, “reed@uni.edu”, 9898887898, S345, CS);

* Delete Query
  + Actor: Administrator
  + Steps:
    - User selects the “Delete Professor” option.
    - Prompt the user to enter the professor id that they want to delete
    - All of the professor’s information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Professor

WHERE professor\_id = 9119;

* Update Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the “Update Professor’s information” option
    - Prompt the user to enter the professor id that they would like to update
    - The professor information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Professor

SET office = N765

WHERE professor\_ID = 23232;

* Aggregate Query
  + Actor: Administrator
  + Steps:
    - User selects “ Total amount of Professor” option
    - Number of currently employed professors is then displayed
  + SQL Code:

SELECT COUNT (professor\_ID) from Professor;

Department:

* Insert Query
  + Actor: Administrator
  + Steps:
    - User selects the “New Department” option
    - A blank field appears, user inputs new department name
    - Prompt user to enter the departments phone number, building and the head of the department.
    - All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Department (dept, dept\_phone, building, head)

VALUES(Accounting, 9890003456, North, Red);

* Delete Query
  + Actor: Administrator
  + Steps:
    - User selects the “Delete Department” option.
    - Prompt the user to enter the department name that they want to delete
    - All of the department’s information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Department

WHERE dept = “CS”;

* Update Query
  + Actor: Administrator
  + Steps:
    - User selects the “Update departments information” option
    - Prompt the user to enter the dept name that they would like to update
    - The department information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Department

SET head = “Wilder”

WHERE dept = Math;

* Aggregate Query
  + Actor: Administrator
  + Steps:
    - User selects “ Total amount of Departments” option
    - Number of current departments is then displayed
  + SQL Code:

SELECT COUNT (dept) from Department;

Courses:

* Insert Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the “New Course” option
    - A new CRN is generated and displayed
    - Prompt user to enter the course\_name, professor\_ID, course\_desc and classroom
    - All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Courses (crn, course\_name, professor\_ID, course\_desc, classroom)

VALUES(5555, CS 1300, 12313, “Visual Basic”, A323);

* Delete Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the “Delete Course” option.
    - Prompt the user to enter the CRN that they want to delete
    - All of the course’s information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Courses

WHERE crn = 9898;

* Update Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the “Update Course’s information” option
    - Prompt the user to enter the CRN that they would like to update
    - The courses information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Courses

SET classroom = E653

WHERE crn = 2323;

* Aggregate Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects “ Total amount of Courses” option
    - Number of active courses is then displayed
  + SQL Code:

SELECT COUNT (crn) from Courses;

Classroom:

* Insert Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the “New Classroom” option
    - A new empty field with classroom is displayed, user enters the classroom number.
    - Prompt user to enter the All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Classroom (classroom, building, capacity)

VALUES(A999, ACAD, 100);

* Delete Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the “Delete Classroom” option.
    - Prompt the user to enter the classroom number that they want to delete
    - All of the classrooms information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Classroom

WHERE classroom = S786;

* Update Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the “Update Clasrooms information” option
    - Prompt the user to enter the clasroom that they would like to update
    - The clasrooms information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Classroom

SET capacity = 25

WHERE classroom = N353;

* Aggregate Query
  + Actor: Administrator
  + Steps:
    - User selects “ Total amount of Clasrooms” option
    - Number of active classrooms is then displayed
  + SQL Code:

SELECT COUNT (classroom) from Classroom;

Advisor:

* Insert Query
  + Actor: Administrator
  + Steps:
    - User selects the “New Advisor” option
    - A blank advisor name field is generated and displayed, user enters the advisors name
    - Prompt user to enter the department, office, phone number and email
    - All information is then displayed and asks for yes or no confirmation
    - User will type “Yes” to confirm
  + SQL:

INSERT INTO Advisor (advisor, dept, office, adv\_phone, adv\_email)

VALUES(“Ron”, CS, A777, 7778889999, ron@uni.edu);

* Delete Query
  + Actor: Administrator
  + Steps:
    - User selects the “Delete Advisor” option.
    - Prompt the user to enter the advisors name that they want to delete
    - All of the student’s information is displayed and asks for confirmation
    - User will then type “Yes”
  + SQL Code:

DELETE FROM Advisor

WHERE advisor = Joe;

* Update Query
  + Actor: Administrator
  + Steps:
    - User selects the “Update Advisors information” option
    - Prompt the user to enter the advisors name that they would like to update
    - The advisors information is then displayed and prompts the user to make the changes.
    - All the new information is displayed and asks for confirmation
    - Users types Yes to confirm
  + SQL Code:

UPDATE Advisor

SET dept = CS

WHERE advisor = Joe;

* Aggregate Query
  + Actor: Administrator
  + Steps:
    - User selects “ Total amount of Advisors” option
    - Number of employed advisors is then displayed
  + SQL Code:

SELECT COUNT (advisors) from Advisor;

* **Joint Queries**

Student & Courses:

* Join Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the option to view which student is taking which class
    - From the Students (students\_ID + first\_name + last\_name) From the Courses (crn + course\_name + course\_desc +classroom) It is also being displayed by alphabetical order by last name.
  + SQL Code: SELECT Students.students\_ID, Students.first\_name, Students.last\_name, Courses.crn, Courses.course\_name, Courses.course\_desc, Courses.classroom  
    FROM Students INNER JOIN  
     Courses ON Students.crn = Courses.crn  
    ORDER BY Students.last\_name

Course & Classroom

* Join Query
  + Actor: Administrator/Staff
  + Steps:
    - User selects the option to view which course is assigned with what classroom
    - From Classroom table (classroom + building) From the courses table (crn + course\_name)
  + SQL Code: SELECT Classroom.classroom, Classroom.building, Courses.crn, Courses.course\_name  
    FROM Classroom INNER JOIN  
     Courses ON Classroom.classroom = Courses.classroom

Course & Professor

* Join Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the option to view what professor is teaching what course
    - From the Professor table (professor\_ID + Professor\_lname) and from the Courses Table (course\_name + course\_desc) Ordered by last name
  + SQL Code: SELECT Professor.professor\_ID, Professor.professor\_lname, Courses.course\_name, Courses.course\_desc  
    FROM Courses INNER JOIN  
     Professor ON Courses.professor\_ID = Professor.professor\_ID  
    ORDER BY Professor.professor\_lname

Professor & Department

* Join Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the option view to see what department is the professor from
    - From the department table (dept) and from the professors table (professor\_ID + professor\_lname) ascending by last name
  + SQL Code: SELECT Department.dept, Professor.professor\_ID, Professor.professor\_lname  
    FROM Department INNER JOIN  
     Professor ON Department.dept = Professor.dept  
    ORDER BY Professor.professor\_lname

Students & Credits

* Join Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the option view to see students information and their credits/GPA
    - From the students table (students\_id + first\_name + last\_name) from the credits table (credits\_earned+ credits attempted + GPA + student\_level) ascending by last name
  + SQL Code: SELECT Students.students\_ID, Students.first\_name, Students.last\_name, Credits.credits\_earned, Credits.credits\_attempted, Credits.GPA, Credits.student\_level  
    FROM Credits INNER JOIN  
     Students ON Credits.student\_ID = Students.students\_ID  
    ORDER BY Students.last\_name

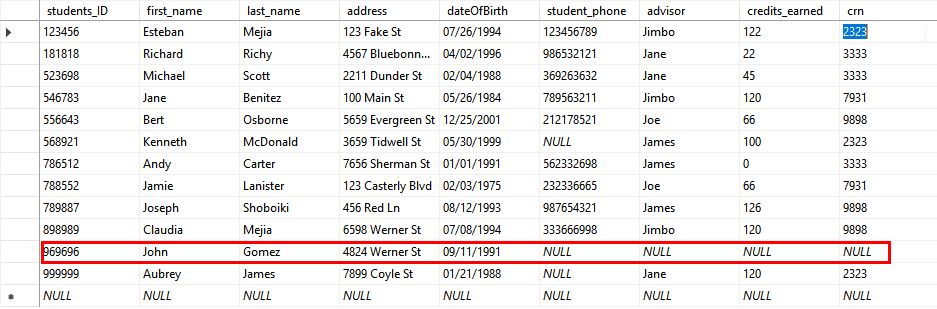
Students & Advisor

* Join Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the option view to see the advisors information for each student
  + SQL Code:SELECT Students.first\_name, Students.last\_name, Advisor.advisor, Advisor.adv\_phone, Advisor.adv\_email  
    FROM Advisor INNER JOIN  
     Students ON Advisor.advisor = Students.advisor  
    ORDER BY Students.last\_name

Advisor & Department

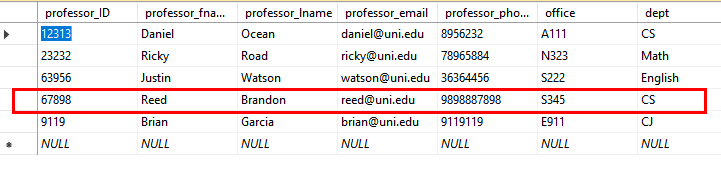
* Join Query
  + Actor: Administrator/Staff/Student
  + Steps:
    - User selects the option view to see what department is the advisor from
    - From the department table (dept + building + head of the department) and from the advisors table (advisor) ascending by advisors name
  + SQL Code: SELECT Advisor.advisor, Department.dept, Department.building, Department.head  
    FROM Advisor INNER JOIN  
     Department ON Advisor.dept = Department.dept  
    ORDER BY Advisor.advisor
* **Test Plan:**

Students Insert Query:

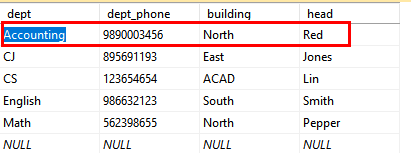


Professor Insert Query:

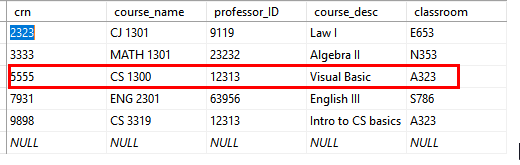
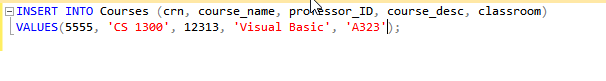




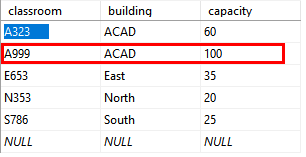
Department Insert Query:



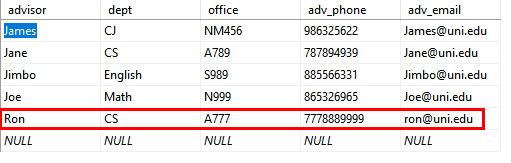
Courses Insert Query:



Classroom Insert Query:



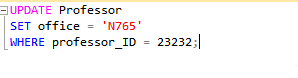
Advisor Insert Query:



Students Update Query:



Professor Update Query:



Department Update Query:







Courses update Query:



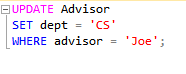
Classroom Update Query:



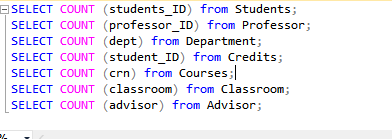


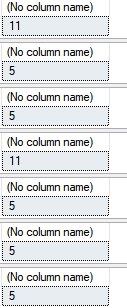


Advisor Update Query:

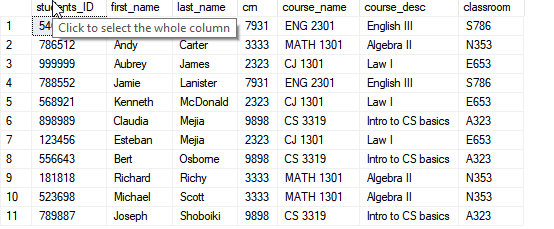
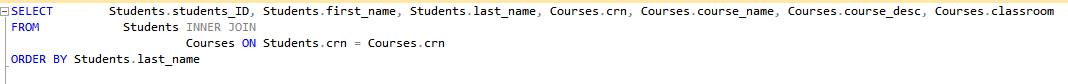


Aggregate Function:

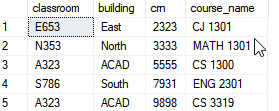
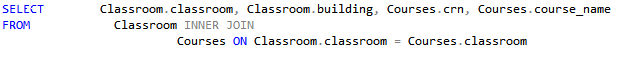




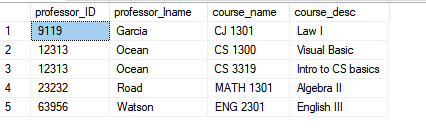
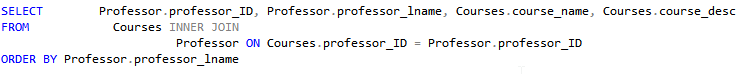
Students & Courses Join Query:



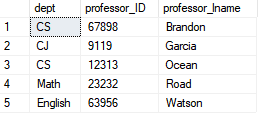
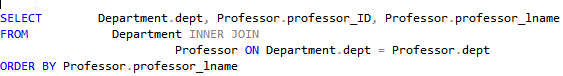
Courses & Classrooms Join Query:



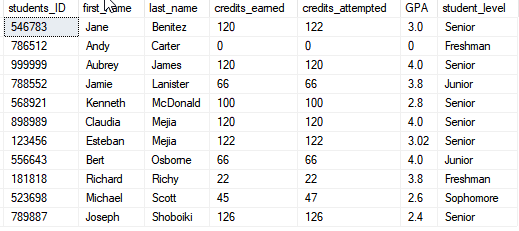
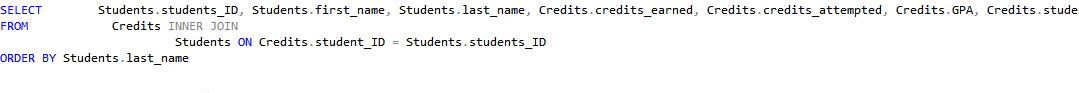
Course & Professor Join Query:



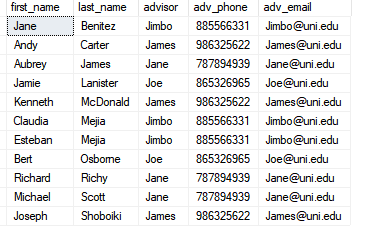
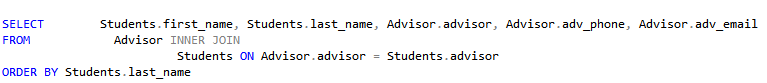
Professor & Department Join Query:



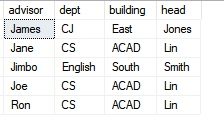
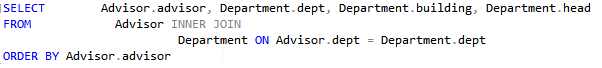
Students & Credits Join Query:



Students & Advisors Join Query:



Advisor & Department Join Query:



* **Conclusion**

H-Town database was an interesting learning experience that allowed group H-Town to develop skills in SQL management. From Functional dependency, creating, modifying tables to normalization. It was also an experience in recognizing anomalies and correcting them. This database is used to help students, staff and administration to manage the school better. Administrators and staff can use it to preform to add/remove/updating: students, courses, departments, etc.,

* **Reference:**

“SQL Tutorial.” W3Schools Online Web Tutorial www.w3schools.com/sql/default.asp.