Databases Individual Assignment

Tiffany Woodley

April 2019

1 Section A

1.1 Give the Create Table statements used to create your relations.

Creating Table structure for Matric Data

```
CREATE TABLE `Assignment1`.`MatricData` (
  `StuID` INT NOT NULL,
  `Prefix` VARCHAR(45) NULL,
  `FirstName` VARCHAR(45) NULL,
  `LastName` VARCHAR(45) NULL,
  `Gender` VARCHAR(45) NULL,
  `Citizenship` VARCHAR(45) NULL,
  `SchoolName` VARCHAR(45) NULL,
  `SchoolCode` INT NULL,
  `Year` INT NULL,
  `ExaminingAuthority` VARCHAR(45) NULL,
  `OldExAuth` VARCHAR(45) NULL,
  `UCTScore` VARCHAR(45) NULL,
  PRIMARY KEY (`StuID`));
```

Loading Matric Data into Table

```
LOAD DATA INFILE '/Users/tiffanywoodley/Desktop/databases/MatricData.csv '
INTO TABLE Assignment1.`MatricData`
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(StuID, Prefix , FirstName, LastName, Gender, Citizenship,
SchoolName, SchoolCode, Year, ExaminingAuthority, OldExAuth, UCTScore);
```

Creating Table for University Data

```
CREATE TABLE `Assignment1`.`UniData` (
```

```
`StuID` INT NOT NULL,
`Course` VARCHAR(45) NULL,
`Percent` VARCHAR(45) NULL,
`Symbol` VARCHAR(45) NULL,
`Credits` VARCHAR(45) NULL,
`Senior` VARCHAR(45) NULL,
`Science` VARCHAR(45) NULL);
```

Loading University Data into Table

```
LOAD DATA INFILE '/Users/tiffanywoodley/Desktop/databases/UniData.csv '
INTO TABLE Assignment1.`UniData`
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(StuID, Course, Percent, Symbol, Credits, Senior, Science);
```

1.2 Output any one tuple from each of your relations.

```
SELECT * FROM MatricData LIMIT 1;
SELECT * FROM UniData LIMIT 1;
```

1.3 Output the number of tuples in each of your relations.

```
SELECT COUNT(*) as NumberofTuples FROM MatricData;
SELECT COUNT(*) as NumberofTuples FROM UniData;
```

1.4 Show all entries where the gender disagrees with the prefix.

Assumptions made were that men can only have the prefix Mr

```
SELECT * FROM MatricData
WHERE (Gender = "F" AND Prefix = "Mr")
OR (Gender = "M" AND Prefix != "Mr");
```

1.5 For each course, give course code and number of units, in alphabetical order of course code.

```
SELECT DISTINCT Course, Credits/6 as Units
FROM UniData;
```

1.6 Give names of all students with citizenship status "C" who are missing their "examining authority", "old ex auth" and "UCT score" data.

```
SELECT FirstName, LastName FROM MatricData
```

```
WHERE  Citizenship = 'C'
AND UCTScore = ''
AND OldExAuth = ''
AND ExaminingAuthority = '';
```

1.7 Give the StuID of students who registered for both ZOO3 and ACCS1.

```
SELECT StuID FROM UniData
WHERE Course = "ZOO3" OR Course = "ACCS1"
GROUP BY StuID
HAVING COUNT(StuID) > 1 ;
```

1.8 Find the course and student in the database with the highest university result,along with that mark.

```
SELECT StuID, Course FROM UniData
 WHERE Percent = (SELECT MAX(Percent)
 FROM UniData
 WHERE Percent != "AB" AND Percent !="DPR");
```

1.9 How many students have a mark (in any course) lower than some mark that student 1027 achieved?

Because the word some was used the assumption was made that the marks should be checked against all of the student 1027's marks

```
SELECT DISTINCT table1.StuID, table1.Course, table1.Percent
FROM UniData as Table1, UniData as Table2
WHERE Table2.StuID = 1027 AND Table1.Percent < Table2.Percent;
```

1.10 What result symbols does UCT use?

```
SELECT DISTINCT Symbol FROM UniData;
```

1.11 Find the mark range for each CS course (so each result line has CS course code, lowest and highest mark obtained for that course).

```
SELECT DISTINCT Course, MIN(Percent) as Lowest, MAX(Percent) as Highest
 FROM UniData WHERE Course LIKE  'CS%'
 AND Percent != "AB" AND Percent != "DPR"
 GROUP BY Course;
```

1.12 Find the highest "UCT Score" obtained each year, but only for years where the average "UCT Score" of that year was between 30 and 40 (inclusive).

```
SELECT Table1.Year,
MAX(table2.UCTScore) as HighestUCTScore
FROM MatricData as Table1, MatricData as Table2
WHERE Table1.Year = Table2.Year
GROUP BY Table1.Year
HAVING AVG(Table1.UCTScore) >= 30 AND AVG(Table1.UCTScore) <=40;
```

1.13 Which course(s) have the lowest number of fails ?

Assumption made that only courses with fails should be included, the courses are then ordered by number of fails ascending

```
SELECT Course, COUNT(Course) as Fails FROM UniData
WHERE Symbol = "F"
GROUP BY Course
ORDER BY COUNT(Course) ASC ;
```

If a user wants to see the courses under or equal to a certain number of fails the following statement could be used. This example shows the course with less than or equal to 2 fails.

```
SELECT Course, COUNT(Course) as Fails FROM UniData
WHERE Symbol = "F"
GROUP BY Course
HAVING COUNT(Course) <= 2
ORDER BY COUNT(Course) ASC ;
```

1.14 For each PSY2 student ,show the number of senior Science courses they have passed.

```
SELECT DISTINCT Table2.StuID, COUNT(Table2.Course) as NumOfCourses
From UniData as Table1 , UniData as Table2
WHERE Table1.Course = "PSY2"
AND Table1.StuID = Table2.StuID
AND Table2.Senior = "Y"
AND Table2.Science= "Y"
GROUP BY Table2.StuID;
```

1.15 Delete all students who matriculated before the year 2000.

Firstly make sure that any associated foreign keys are deleted

```
DELETE FROM UniData
WHERE stuID = (SELECT stuID
```

```
FROM MatricData
WHERE Year < 2000);
```

Then delete the rows of the students who matriculated before the year 2000

```
DELETE FROM MatricData WHERE Year <2000;
```

1.16 Insert a new result for 1001, who also took ZOO1 but was absent from the exam.

```
INSERT INTO UniData
(StuID, Course, Percent, Symbol, Credits, Senior, Science)
 VALUES(1001, "ZOO1", "AB", "AB",18, "N","Y");
```

1.17 The university has decided to use sum-of-6-best-matric-percentages as "UCTscores", instead of the old system. Change all "UCTScores" in the database accordingly by multiplying them by 10 and adding 30 (as a rough approximation).

```
UPDATE MatricData SET UCTScore = UCTScore*10 + 30
WHERE UCTScore != '';
```

2 Section B

Query Purpose

The purpose of the query is see the academic performance of each student by viewing their

* Average mark over all their passed courses
* Minimum mark over all their passed courses
* Maximum mark over all their passed courses
* Total credits of passed courses
* Their UCT score

Query Statement

```
SELECT DISTINCT
table2.StuID,
CONCAT(table1.FirstName, + " ", + table1.LastName) as Name,
ROUND(table2.average,2) as Average,
table2.minimum as Minimum,
table2.maximum as Maximum,
table2.totalcredits as TotalCredits,
table1.UCTScore
FROM (SELECT UniData.StuID, MatricData.FirstName ,
MatricData.LastName, MatricData.UCTScore
FROM MatricData INNER JOIN UniData
ON MatricData.StuID = UniData.StuID) as table1,
(SELECT stuID, AVG(percent) as average,
MIN(percent) as Minimum,
MAX(percent) as Maximum,
SUM(credits) as totalcredits
FROM uniData
WHERE percent != "AB" AND percent != "DPR" AND symbol != "F"
GROUP BY StuID) as table2
WHERE table1.StuID = table2.StuID;
```

View of first 6 lines of the table the query returns

StuID	Name	Average	Minimum	Maximum	TotalCredits	UCTScore
1001	Aaron Zwillig	66.20	58	88	108	34
1002	Aarthi Zulu	62.50	50	73	126	48
1003	Abeedah Zowa	64.20	56	73	96	42
1004	Abigail Zinn	64.50	55	72	132	39
1005	Adrienne Zide	61.83	50	77	186	35
1006	Aimee Zalgaonker	68.33	62	75	144	

Checking if the returned table is correct

Each column is checked for a different student number then compared to the outputted table

Checking Name column on student 1001

```
SELECT FirstName, LastName
FROM MatricData
WHERE StuID = 1001;
```

RETURNED: Aaron Zwilling

Checking Average column on student 1002

```
SELECT AVG(Percent)
FROM UniData
WHERE StuID = 1002
GROUP BY StuID;
```

RETURNED: 62.5

Checking Minimum column on student 1003

```
SELECT MIN(Percent)
FROM UniData
WHERE StuID = 1003
AND symbol != "F"
GROUP BY StuID;
```

RETURNED: 56

Checking Maximum column on student 1004

```
SELECT MAX(Percent)
FROM UniData
WHERE StuID = 1004
AND symbol != "F"
GROUP BY StuID;
```

RETURNED: 72

Checking TotalCredits column on student 1005

```
SELECT SUM(Credits)
FROM UniData
WHERE StuID = 1005
AND symbol != "F"
GROUP BY StuID;
```

RETURNED: 186

Checking UCTScore column on student 1006

```
SELECT UCTScore
FROM MatricData
WHERE StuID = 1006;
```

RETURNED: Blank

3 Section C

3.1 Name a foreign key attribute,in the form Tablename.Columnname

The foreign key would be **UniData.StuID**, as its the the primary key from MatricData and can be used to link back to MatricData.

3.2 List all functional dependencies that apply to this data ,in the form X to Y, where Y is all the attributes functionally determined by X.

- StuID → FirstName, LastName, Prefix, Citizenship status, School Name, ExaminingAuthority, OldExAuth, UCTScore
- Prefix → Gender
- SchoolCode → SchoolName
- Course → Credits, Senior, Science
- StuID, Course → Percentage
- Percentage → Symbol

3.3 Is your database in 3rd normal form or not? Why?

In the MatricData table the column School Code is dependant on School name, this means a non-prime attribute is dependant on another no-prime attribute within the table. This is a transitive dependency and means the database is not in 3rd normal form.

3.4 If you had not been given the schema to use, would your own relational database design for this data have been different? Discuss the change(s) that you would have considered, whether you would have made the change or not, and the reasons for your preference.

Considerations

- I would have considered creating a separate table for courses which held each course's information such as columns Credits, Science and senior
- I would have considered creating a separate table to hold the school codes
- I would have considered creating a separate table for the symbols and their percent ranges

Implementations

- I would have implemented a separate table for course information, this would mean in the UniData table there would only need the course code. This would reduce the risk of conflicting entries about courses and help move the database into second normal form.

  UniData = (StudID, CourseCode, percent, symbol)
  CourseData = (CourseCode,Credits, Senior, Science)

- I would have implemented a separate table for school information, this would mean in the MatricData table there would not be a need for the school name as this would be stored in a separate table. This would reduce the risk of conflicting entries and help move the database into third normal form

  MatricData = (StuID, Prefix, FirstName, LastName, Gender, Citizenship, SchoolCode, Year, ExaminingAuthority, OldExAuth, UCTScore)
  SchoolData = (SchoolCode, SchoolName)

- I would not implement a separate table for the symbol ranges as different courses might have different percentage ranges for symbols. The symbol would then also have to be calculated based on the percentage value which could potentially lead to errors in the database

3.5 Consider the SQL operations below. For all those below that you did not use in Section A of this assignment (if any), give an example usage of this now, to show how it could be applied to this data – in each case give your example SQL use and a brief explanation of the statement's effect/result.

3.5.1 UNION, INTERSECT or MINUS/EXCEPT

This query allows user to check all student IDs over the entire database - can be used to check that it matches the stuID column in the matricData table.

```
SELECT StuID FROM MatricData
UNION
SELECT StuID FROM UniData
ORDER BY StuID ASC;
```

3.5.2 JOIN

This query allows a user to see the first and last names of all the students who have taken the course ZOO1.

```
SELECT DISTINCT FirstName, LastName
FROM MatricData
JOIN UniData
ON MatricData.StuID = UniData.StuID
WHERE UniData.Course = "Z001";
```

3.5.3 IN

This query allows a user to see all the students with a citizenship status of C which have taken the course Z001

```
SELECT StuID, FirstName, LastName
FROM MatricData
WHERE citizenship = "C"
AND StuID IN
(SELECT StuID
FROM UniData
WHERE Course = "Z001");
```

3.5.4 EXISTS

This query shows which students from the matric database are not enrolled in courses

```
SELECT StuID
FROM MatricData
WHERE NOT EXISTS
(SELECT * FROM UniData
WHERE MatricData.StuID = UniData.StuID);
```

3.5.5 UNIQUE

This query will enforce a constraint that any StuID values entered into the MatricData table will have to be unique

```
ALTER TABLE MatricData MODIFY StuID INT UNIQUE;
```

3.5.6 SOME

This query will return the first and last name for any student that takes Z001

```
SELECT FirstName, LastName
FROM MatricData
WHERE StuID = SOME
(SELECT StuID
FROM UniData
WHERE Course = "Z001");
```

3.5.7 ALL

This query returns the StuIDs that are not found in the UniData Table

```
SELECT StuID
FROM MatricData
WHERE StuID != ALL
(SELECT StuID
FROM UniData
WHERE Course = "ZOO1");
```