CPS 610 – Assignment 2 Report

Student Numbers and Names:
1) 500981017 – Mohamed Shrief
2) 500189282 – Chris Fontein
3) 501108034 – Ekrem Yilmaz

Task 1: Creating Relational Schemas

(1.a) Relational Schema for Central Database:

Central_Student

StudentNo	*StudentName*	*Degree*	*GPA*
INT (PK)	*VARCHAR(100)*	*VARCHAR(50)*	*FLOAT*

Central_Professor

ProfName	*ProfOffice*	*ProfPhone*
VARCHAR(100) (PK)	*VARCHAR(100)*	*VARCHAR(10)*

Central_Course

CourseNo	*CourseName*	*Credits Department*	*Department*
INT (PK)	*VARCHAR(100)*	*INT*	*VARCHAR(50)*

Central_Can_Teach

ProfName	*CourseNo*	*Evaluation*
VARCHAR(100)	*INT*	*CHAR(1)*
(FK: Central_Professor)	*(FK: Central_Course)*	*(Y or N)*

Central_Teaches

ProfName	*CourseNo*	*Term*
VARCHAR(100)	*INT*	*VARCHAR(50)*
(FK: Central_Professor)	*(FK: Central_Course)*	

Central_Enrolled

StudentNo	*CourseNo*	*Status*
INT	*INT*	*CHAR(1)*
(FK: Central_Student)	*(FK: Central_Course)*	*(Y or N)*

(1.b) Relational Schema for Engineering Database:

Engineering_Student

StudentNo	*StudentName*	*Degree*	*GPA*
INT (PK)	*VARCHAR(100)*	*VARCHAR(50)*	*FLOAT*

Engineering_Professor

ProfName	*ProfOffice*	*ProfPhone*
VARCHAR(100) (PK)	*VARCHAR(100)*	*VARCHAR(10)*

Engineering_Course

CourseNo	*CourseName*	*Credits Department*	*Department*
INT (PK)	*VARCHAR(100)*	*INT*	*VARCHAR(50)*

Engineering_Can_Teach

ProfName	*CourseNo*	*Evaluation*
VARCHAR(100)	*INT*	*CHAR(1)*
(FK: Central_Professor)	*(FK: Central_Course)*	*(Y or N)*

Engineering_Teaches

ProfName	*CourseNo*	*Term*
VARCHAR(100)	*INT*	*VARCHAR(50)*
(FK: Central_Professor)	*(FK: Central_Course)*	

Engineering_Enrolled

StudentNo	*CourseNo*	*Status*
INT	*INT*	*CHAR(1)*
(FK: Central_Student)	*(FK: Central_Course)*	*(Y or N)*

(1.c) Relational Schema for Science Database:

Science_Student

StudentNo	*StudentName*	*Degree*	*GPA*
INT (PK)	*VARCHAR(100)*	*VARCHAR(50)*	*FLOAT*

Science_Professor

ProfName	*ProfOffice*	*ProfPhone*
VARCHAR(100) (PK)	*VARCHAR(100)*	*VARCHAR(10)*

Science_Course

CourseNo	*CourseName*	*Credits Department*	*Department*
INT (PK)	*VARCHAR(100)*	*INT*	*VARCHAR(50)*

Science_Can_Teach

ProfName	*CourseNo*	*Evaluation*
VARCHAR(100)	*INT*	*CHAR(1)*
(FK: Central_Professor)	*(FK: Central_Course)*	*(Y or N)*

Science_Teaches

ProfName	*CourseNo*	*Term*
VARCHAR(100)	*INT*	*VARCHAR(50)*
(FK: Central_Professor)	*(FK: Central_Course)*	

Science_Enrolled

StudentNo	*CourseNo*	*Status*
INT	*INT*	*CHAR(1)*
(FK: Central_Student)	*(FK: Central_Course)*	*(Y or N)*

Task 2: Relational Algebra and SQL for Populating DDBs

(2.a) Relational Algebra and SQL: Populating Engineering Database

Engineering_Student

Relational Algebra:
Engineering_Student ← π StudentNo, StudentName, Degree, GPA (Central_Student ⋈ Central_Enrolled ⋈ σ Department = 'ENG' (Central_Course))

SQL:

```
   INSERT INTO Engineering_Student@EngineeringDB
       SELECT Central_Student.*
           FROM Central_Student
           JOIN Central_Enrolled
               ON Central_Student.StudentNo=Central_Enrolled.StudentNo
           JOIN Central_Course
               ON Central_Course.CourseNo=Central_Enrolled.CourseNo
               AND Central_Course.department='ENG'
           GROUP BY Central_Student.StudentNo, Central_Student.StudentName,
Central_Student.Degree, Central_Student.GPA
   ;
```

Engineering_Professor

Relational Algebra:
Engineering_Professor ← π ProfName, ProfOffice, ProfPhone (Central_Professor ⋈ Central_Teaches ⋈ σ Department = 'ENG' (Central_Course))

SQL:

```
   INSERT INTO Engineering_Professor@EngineeringDB
       SELECT Central_Professor.*
           FROM Central_Professor
           JOIN Central_Teaches
               ON Central_Professor.ProfName=Central_Teaches.ProfName
           JOIN Central_Course
               ON Central_Course.CourseNo=Central_Teaches.CourseNo
               AND Central_Course.department='ENG'
       UNION
       SELECT Central_Professor.*
           FROM Central_Professor
           JOIN Central_Can_Teach
               ON Central_Professor.ProfName=Central_Can_Teach.ProfName
           JOIN Central_Course
               ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
```

```
            AND Central_Course.department='ENG'
;
```

Engineering_Course

Relational Algebra:
Engineering_Course ← σ Department = 'ENG' (Central_Course)

SQL:

```
INSERT INTO Engineering_Course@EngineeringDB
    SELECT Central_Course.*
        FROM Central_Course
        WHERE Central_Course.department='ENG'
;
```

Engineering_Can_Teach

Relational Algebra:
Engineering_Can_Teach ← π ProfName, CourseNo, Evaluation (Central_Can_Teach ⋈ σ Department = 'ENG' (Central_Course))

SQL:

```
INSERT INTO Engineering_Can_Teach@EngineeringDB
    SELECT Central_Can_Teach.*
        FROM Central_Can_Teach
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
            AND Central_Course.department='ENG'
;
```

Engineering_Teaches

Relational Algebra:
Engineering_Teaches ← π ProfName, CourseNo, Term (Central_Teaches ⋈ σ Department = 'ENG' (Central_Course))

SQL:

```
INSERT INTO Engineering_Teaches@EngineeringDB
    SELECT Central_Teaches.*
        FROM Central_Teaches
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Teaches.CourseNo
            AND Central_Course.department='ENG'
;
```

Engineering_Enrolled

Relational Algebra:
Engineering_Enrolled ← π StudentNo, CourseNo, Status (Central_Enrolled ⋈ σ Department = 'ENG' (Central_Course))

SQL:

```
INSERT INTO Engineering_Enrolled@EngineeringDB
    SELECT Central_Enrolled.*
        FROM Central_Enrolled
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Enrolled.CourseNo
            AND Central_Course.department='ENG'
;
```

(2.b) Relational Algebra and SQL: Populating Science Database

Science_Student

Relational Algebra:
Science_Student ← π StudentNo, StudentName, Degree, GPA (Central_Student ⋈ Central_Enrolled ⋈ σ Department = 'SCI' (Central_Course))

SQL:

```
INSERT INTO Science_Student@ScienceDB
    SELECT Central_Student.*
        FROM Central_Student
        JOIN Central_Enrolled
            ON Central_Student.StudentNo=Central_Enrolled.StudentNo
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Enrolled.CourseNo
            AND Central_Course.department='SCI'
        GROUP BY Central_Student.StudentNo, Central_Student.StudentName,
Central_Student.Degree, Central_Student.GPA
;
```

Science_Professor

Relational Algebra:
Science_Professor ← π ProfName, ProfOffice, ProfPhone (Central_Professor ⋈ Central_Teaches ⋈ σ Department = 'SCI' (Central_Course))

SQL:

```
INSERT INTO Science_Professor@ScienceDB
    SELECT Central_Professor.*
```

```
        FROM Central_Professor
        JOIN Central_Teaches
            ON Central_Professor.ProfName=Central_Teaches.ProfName
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Teaches.CourseNo
            AND Central_Course.department='SCI'
    UNION
    SELECT Central_Professor.*
        FROM Central_Professor
        JOIN Central_Can_Teach
            ON Central_Professor.ProfName=Central_Can_Teach.ProfName
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
            AND Central_Course.department='SCI'
;
```

Science_Course

Relational Algebra:
Science_Course ← σ Department = 'SCI' (Central_Course)

SQL:

```
INSERT INTO Science_Course@ScienceDB
    SELECT Central_Course.*
        FROM Central_Course
        WHERE Central_Course.department='SCI'
;
```

Science_Can_Teach

Relational Algebra:
Science_Can_Teach ← π ProfName, CourseNo, Evaluation (Central_Can_Teach ⋈ σ Department = 'SCI' (Central_Course))

SQL:

```
INSERT INTO Science_Can_Teach@ScienceDB
    SELECT Central_Can_Teach.*
        FROM Central_Can_Teach
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
            AND Central_Course.department='SCI'
;
```

Science_Teaches

Relational Algebra:

Science_Teaches ← π ProfName, CourseNo, Term (Central_Teaches ⋈ σ Department = 'SCI' (Central_Course))

SQL:

```
INSERT INTO Science_Teaches@ScienceDB
    SELECT Central_Teaches.*
        FROM Central_Teaches
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Teaches.CourseNo
            AND Central_Course.department='SCI'
;
```

Science_Enrolled

Relational Algebra:

Science_Enrolled ← π StudentNo, CourseNo, Status (Central_Enrolled ⋈ σ Department = 'SCI' (Central_Course))

SQL:

```
INSERT INTO Science_Enrolled@ScienceDB
    SELECT Central_Enrolled.*
        FROM Central_Enrolled
        JOIN Central_Course
            ON Central_Course.CourseNo=Central_Enrolled.CourseNo
            AND Central_Course.department='SCI'
;
```

Task3: Relational Algebra and SQL for Reconstruction of Central DB

Central_Student

Relational Algebra:
Central_Student ← Engineering_Student ∪ Science_Student

SQL:

```
INSERT INTO Central_Student
    SELECT * FROM Science_Student@ScienceDB
    UNION
    SELECT * FROM Engineering_Student@EngineeringDB
    ;
```

Central_Professor

Relational Algebra:
Central_Professor ← Engineering_Professor ∪ Science_Professor

SQL:

```
INSERT INTO Central_Professor
    SELECT * FROM Science_Professor@ScienceDB
    UNION
    SELECT * FROM Engineering_Professor@EngineeringDB
    ;
```

Central_Course

Relational Algebra:
Central_Course ← Engineering_Course ∪ Science_Course

SQL:

```
INSERT INTO Central_Course
    SELECT * FROM Science_Course@ScienceDB
    UNION
    SELECT * FROM Engineering_Course@EngineeringDB
    ;
```

Central_Can_Teach

Relational Algebra:
Central_Can_Teach ← Engineering_Can_Teach ∪ Science_Can_Teach

SQL:

```
INSERT INTO Central_Can_Teach
```

```
    SELECT * FROM Science_Can_Teach@ScienceDB
    UNION
    SELECT * FROM Engineering_Can_Teach@EngineeringDB
    ;
```

Central_Teaches

Relational Algebra:
Central_Teaches ← Engineering_Teaches ∪ Science_Teaches

SQL:

```
INSERT INTO Central_Teaches
    SELECT * FROM Science_Teaches@ScienceDB
    UNION
    SELECT * FROM Engineering_Teaches@EngineeringDB
    ;
```

Central_Enrolled

Relational Algebra:
Central_Enrolled ← Engineering_Enrolled ∪ Science_Enrolled

SQL:

```
INSERT INTO Central_Enrolled
    SELECT * FROM Science_Enrolled@ScienceDB
    UNION
    SELECT * FROM Engineering_Enrolled@EngineeringDB
    ;
```

Task 4: Relational Algebra and Queries
(1) List of classes taken by students in Engineering and Science faculties.
Relational Algebra:
π CourseName,StudentName (σ Department = 'ENG' ∨ Department = 'SCI' (Central_Student ⋈ Central_Enrolled ⋈ Central_Course))

SQL:

```
SELECT *
   FROM Central_Course
   WHERE Central_Course.department='SCI'
       OR Central_Course.department='ENG'
;
```

(2) Retrieve students with a GPA greater than or equal to 4.
Relational Algebra:
σ GPA≥4 (Central_Student)

SQL:

```
SELECT *
   FROM Central_Student
   WHERE Central_Student.GPA >= 4
;
```

(3) Calculate the average GPA for students enrolled in each course.
Relational Algebra:
γ CourseNo,AVG(GPA) (Central_Student⋈Central_Enrolled)

SQL:

```
SELECT Central_Enrolled.CourseNo, AVG(Central_Student.GPA)
   FROM Central_Student
   JOIN Central_Enrolled
       ON Central_Student.StudentNo=Central_Enrolled.StudentNo
   GROUP BY Central_Enrolled.CourseNo
;
```

(4) Identify professors teaching courses they have not listed as teachable.
Relational Algebra:
π ProfName,CourseNo,Term (Central_Teaches⋈(ρ ProfName,CourseNo (Central_Can_Teach)) ÷ Central_Can_Teach)

SQL:

```
SELECT Central_Professor.*, Central_Teaches.CourseNo, Central_Teaches.Term,
Central_Can_Teach.Evaluation
   FROM Central_Professor
```

```
   JOIN Central_Teaches
       ON Central_Professor.ProfName=Central_Teaches.ProfName
   JOIN Central_Can_Teach
       ON Central_Professor.ProfName=Central_Can_Teach.ProfName
       AND Central_Can_Teach.Evaluation='N'
   GROUP BY Central_Professor.profname, Central_Professor.profoffice,
Central_Professor.profphone, Central_Teaches.CourseNo, Central_Teaches.Term,
Central_Can_Teach.Evaluation
;
```

(5) Retrieve professors who can teach both Science and Engineering courses.
Relational Algebra:
π ProfName (σ Department = 'SCI' (Central_Can_Teach⋈Central_Course)) ∩ π ProfName (σ Department = 'ENG' (Central_Can_Teach⋈Central_Course))

SQL:

```
SELECT Central_Professor.*
   FROM Central_Professor
   JOIN Central_Can_Teach
       ON Central_Professor.ProfName=Central_Can_Teach.ProfName
       AND Central_Can_Teach.Evaluation='Y'
   JOIN Central_Course
       ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
       AND Central_Course.department='SCI'
INTERSECT
SELECT Central_Professor.*
   FROM Central_Professor
   JOIN Central_Can_Teach
       ON Central_Professor.ProfName=Central_Can_Teach.ProfName
       AND Central_Can_Teach.Evaluation='Y'
   JOIN Central_Course
       ON Central_Course.CourseNo=Central_Can_Teach.CourseNo
       AND Central_Course.department='ENG'
;
```

(6) List students enrolled in Professor Harry's classes, including course names.
Relational Algebra:
π StudentName,CourseName (σ ProfName = 'Dr.Harry' (Central_Student ⋈ Central_Enrolled ⋈ Central_Course ⋈ Central_Teaches))

SQL:

```
SELECT Central_Student.*, Central_Course.CourseNo, Central_Course.CourseName,
Central_Teaches.ProfName
   FROM Central_Student
   JOIN Central_Enrolled
       ON Central_Student.StudentNo=Central_Enrolled.StudentNo
   JOIN Central_Course
       ON Central_Course.CourseNo=Central_Enrolled.CourseNo
   JOIN Central_Teaches
       ON Central_Teaches.CourseNo=Central_Enrolled.CourseNo
       AND Central_Teaches.ProfName='Professor Harry'
;
```

(7) Identify courses worth 1 credit in both Science and Engineering.
Relational Algebra:
σ Credits = 1 ∧ (Department = 'SCI' ∨ Department = 'ENG') (Central_Course)

SQL:

```
SELECT *
   FROM Central_Course
   WHERE (Central_Course.department='SCI'
       OR Central_Course.department='ENG')
       AND Central_Course.credits=1
;
```

(8) Retrieve professors teaching during the summer term.
Relational Algebra:
π ProfName (σ Term = 'Summer' (Central_Teaches))

SQL:

```
SELECT Central_Professor.*, Central_Teaches.CourseNo, Central_Teaches.Term
   FROM Central_Professor
   JOIN Central_Teaches
       ON Central_Professor.ProfName=Central_Teaches.ProfName
   WHERE LOWER(Central_Teaches.Term)='summer'
;
```