CS 586 Introduction to Databases
Graduate Project Database Implementation
Winter 2025 Quarter

Important notes or clarifications will be added here.

Name: Aura Castellanos Calderon & Marchelle Le
Due Dates (everything submitted on Canvas):
Part 1: Domain Description and Data Source - Friday, Feb 21st, 11:59 pm
Part 2: ER Diagram and Relational Schema - Friday, Feb 28th, 11:59 pm
Part 3: Final Write-up with Queries and Results - Monday, March 17th, 11:59 pm
This project is for CS 586 students only.
You may do this assignment individually or you may work with one partner. If you work with a partner, both of you need to turn in identical assignments, and put both of your names on the assignment. Submissions are through Canvas. Submit PDF files and include a GitHub link that you update throughout the project. I encourage you to start early.

Project Overview
This project's goal is to gain experience with database design and implementation. You will choose a real-world subject area on which you implement a database. Example topics are campus sports teams, climate change, economics, hospitals, etc. You will create a database for that subject and then run queries over that database. Please avoid topics such as board games or video games. If you have any questions, please contact the professor. There are three parts/submissions for this project.

Part 1: Data Description (Due Feb 21st):
a) Select a **real-world** subject area on which you wish to build a database. Write approximately one paragraph and give a general description/background information on that subject area.

The area I chose, based on the database I found, is the relationship between academic performance and career success. The database contains 5,000 records with attributes such as gender, age, GPA, starting salary, and internship experience. This database seems very complete and has great potential for analysis. Some possible projects include predicting job success based on education, identifying key factors influencing salaries, and understanding the role of networking and internships in career growth.

b) List 20 questions (in English) that someone might want to ask about the domain. You will later translate these questions to SQL queries in part 3 of this project. Your questions need to be diverse. Think about designing your questions in a way that you can use a variety of topics that we covered in class. The questions shouldn't be intended to use only basic SQL queries.

Note: You are allowed to revise these questions later if needed (with mentioned reasons for the modification.)

Create a query that shows only professors who teach in the Computer Science department.

1. Retrieve all student information who enrolled after 2020 and have a GPA above 3.0.
2. Find students with a high school GPA above 3.5 who are majoring in Business or Computer Science.
3. Show students who scored above 1200 on the SAT, along with their age, gender, and field of study, but only include students from universities ranked in the top 50.
4. Create a query that shows fields of study in descending order by average GPA, but exclude fields with fewer than 5 students.
5. List students attending universities ranked in the top 100 who also have at least one internship.
6. Find students majoring in Computer Science who also have a work-life balance score above 7.
7. Retrieve students who have completed at least 2 internships and scored above 1300 on the SAT.
8. Find the best student by GPA in each department, but only include students who have at least one certification.
9. Show the average salary of each department, but only include departments with at least 10 students.
10. Create a query that shows career satisfaction by field of study, grouped by gender.
11. Create a query with the 10 students who have completed the most projects and have a GPA above 3.5.
12. Give the average age by each field of study, but only include fields with at least 3 students.
13. Show the average salary by gender for students majoring in STEM fields.
14. Create a query that shows the current job level by field of study for students with a GPA above 3.0.
15. Get students with a soft skills score greater than 8 and a career satisfaction score above 4.
16. Find students with a networking score of at least 7 and a GPA greater than 3.2.
17. Get students with a starting salary above $50,000 who graduated within the last 5 years.
18. Find students who rated their career satisfaction as 5 and have more than 2 certifications.
19. List students promoted within 3 years, showing age, gender, and field of study, but only for those with at least 1 internship.
20. Find students currently in a senior-level job with a soft skills score above 7.
21. Get students with a work-life balance score of 8 or higher and a starting salary above $60,000.
22. Find students who are entrepreneurs, showing gender, age, and field of study, but only if they have completed at least 2 projects.

23. Retrieve students who meet both GPA > 3.8 and SAT > 1300 and have a job offer count above 2.
24. Find students with internships, certifications, and projects all above 5, but only for those from top 100 universities.
25. Get students majoring in STEM fields with a job offer count above 4 and a starting salary above $70,000.

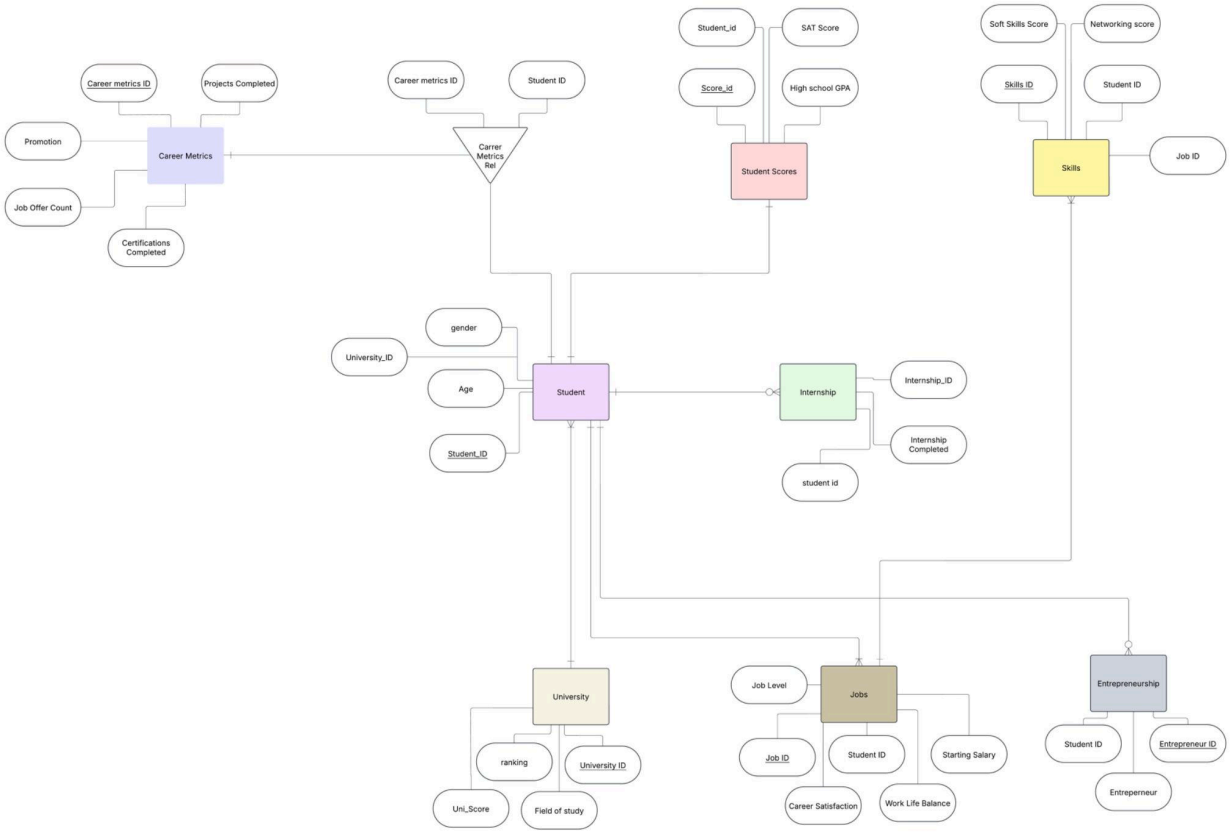
c) Describe what sources you intend to use as your data, and how you *intend to ingest the data into your database*. You must select a subject area from which you can get several hundred rows of data.
Your dataset should be diverse so you can build an interesting real-world database.
The database Education and Career Success.
Reference:
Shamim, A. (n.d.). *Education and Career Success* [Dataset]. Kaggle. Retrieved February 15, 2025, from https://www.kaggle.com/datasets/adilshamim8/education-and-career-success

d) Start your GitHub project, share it with the instructor, and include the link in your PDF submission.
https://github.com/MarsTL/cs586-database-project

Part 2: ER Diagram and Relational Schema (Due Feb 28th):
Produce an ER diagram for your chosen domain and its translation into a relational schema, including all keys and foreign keys. Your database must contain 5 – 10 tables. Please cite references that you used for your data. You should also submit evidence that you have created at least one table from your schema and populated it with at least one row.
Note on 5 – 10 tables.

- You have to use multiple CSV files as your tables.
- Some examples of resources to look for CSV files:
    - https://data.world/datasets/csv
    - https://www.kaggle.com/datasets
    - https://www.stats.govt.nz/large-datasets/csv-files-for-download/
    - https://datahub.io/collections
    - https://datahub.io/collections or https://datahub.io/search
    - https://apps.who.int/gho/data/node.home?search= or https://apps.who.int/gho/data/node.home
    - https://datasetsearch.research.google.com (choose the "free" label while searching for your topic)
    - Also, search in Google or the search engine of your choice with the keywords of your topic in addition to "CSV" or "Dataset", etc.

ER diagram:



Student(student_id, university_id (FK), age, gender)
StudentScores(score_id, student_id (FK), high_school_GPA, SAT_score)
University(uiversity_id, ranking, uni_score, field_of_study)
Internship(intership_id, student_id (FK), internship_completed)
Jobs(job_id, student_id (FK), starting_salary, job_level, career_satisfaction, life_balance)
Entrepreneurship(entrepreneurship_id, student_id (FK), entrepreneurship)
CareerMetrics(career_metrics_id, projects_completed, promotion, job_offer, certificate_completed)
CareerMetricsRel(career_metrics_id, student_id, career_metrics_id(FK), student_id (FK))
Student_Skills(skills_id , student_id (FK), skills_id (FK), networking_score, soft_skills_score)

SQL command

```
CREATE TABLE Students (
    Student_ID VARCHAR(10) PRIMARY KEY,
    Age INT,
    Gender VARCHAR(10),
    Score_ID INT,
    University_ID INT
)
```

Query executed OK, 0 rows affected. (0.020 s) Edit

```
CREATE TABLE Students (
    Student_ID VARCHAR(10) PRIMARY KEY,
    Age INT,
    Gender VARCHAR(10),
    Score_ID INT,
    University_ID INT
);
```

Execute Limit rows: Stop on error Show only errors

History

Select: students

5,000 rows have been imported. 22:59:29 SQL command

Select data Show structure Alter table New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT * FROM "students" LIMIT 50 (0.001 s) Edit
```

Modify	student_id	age	gender	score_id	university_id
edit	S00001	24	Male	1	1
edit	S00002	21	Other	2	2
edit	S00003	28	Female	3	3
edit	S00004	25	Male	4	4
edit	S00005	22	Male	5	5
edit	S00006	24	Male	6	6
edit	S00007	27	Male	7	7
edit	S00008	20	Male	8	8
edit	S00009	24	Male	9	9
edit	S00010	28	Male	10	10
edit	S00011	28	Female	11	11
edit	S00012	25	Female	12	12
edit	S00013	22	Female	13	13
edit	S00014	21	Male	14	14
edit	S00015	25	Male	15	15
edit	S00016	25	Female	16	16
edit	S00017	20	Female	17	17
edit	S00018	23	Male	18	18
edit	S00019	22	Female	19	19
edit	S00020	19	Female	20	20
edit	S00021	25	Female	21	21
edit	S00022	29	Male	22	22
edit	S00023	23	Female	23	23

Page: 1 2 3 4 5 ... 100 | Whole result: 5,000 rows | Modify: Save | Selected (0): Edit Clone Delete | Export

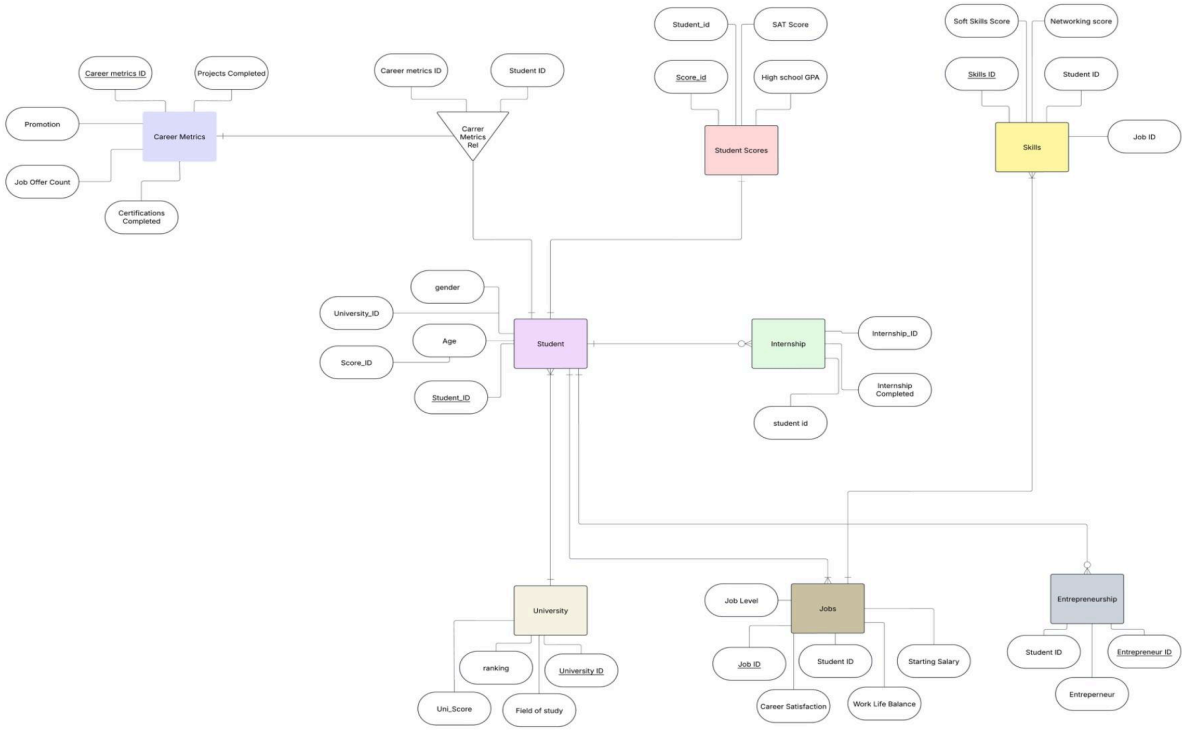
Part 3: Final Write-up (Due March 17th):

- Full references to the original data used for your tables.

The original data used for the project is listed in the references section at the end of the file. Additional primary keys were added to the original database to meet project requirements and ensure relational integrity and consistency. The fixed database file is **education career success.csv**. The file is in the github project

- Your ER diagram, showing any changes you made during the implementation process

The ER diagram was updated by adding the attribute score_ID in the students table.



- The CREATE TABLE Statements from the education career success.csv database

Students table:

```
CREATE TABLE students (
        student_id VARCHAR(10) PRIMARY KEY,
        age INT,
        gender VARCHAR(10),
        score_id INT,
        university_id INT,
        score_id INT REFERENCES Student_Score(Score_ID),
        university_id INT REFERENCES University(University_ID));
```

Table: students

Select data **Show structure** Alter table New item

Column	Type	Comment
student_id	character varying(10)	
university_id	integer *NULL*	
age	integer *NULL*	
gender	character varying(10) *NULL*	
score_id	integer *NULL*	

Indexes

PRIMARY	*student_id*

Alter indexes

Foreign keys

Source	Target	ON DELETE	ON UPDATE	
university_id	university(*university_id*)	CASCADE	NO ACTION	Alter
score_id	student_score(*score_id*)	SET NULL	NO ACTION	Alter

Add foreign key

Triggers

Add trigger

Select: students

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100

```
SELECT * FROM "students" LIMIT 50 (0.001 s) Edit
```

Modify	student_id	age	gender	score_id	university_id
edit	S00001	24	Male	1	1
edit	S00002	21	Other	2	2
edit	S00003	28	Female	3	3
edit	S00004	25	Male	4	4
edit	S00005	22	Male	5	5
edit	S00006	24	Male	6	6
edit	S00007	27	Male	7	7

total row: 5,000

Student_Score table:

```
CREATE TABLE Student_Score (
  Student_ID VARCHAR(10) PRIMARY KEY,
  Score_ID INT,
  High_School_GPA DECIMAL(3,2),
  SAT_Score INT);
```

Table: student_score

Select data **Show structure** Alter table New

Column	Type	Comment
student_id	character varying(10)	
score_id	integer	
high_school_gpa	numeric(3,2) *NULL*	
sat_score	integer *NULL*	

Indexes

PRIMARY	*score_id*

Alter indexes

Select: student_score

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

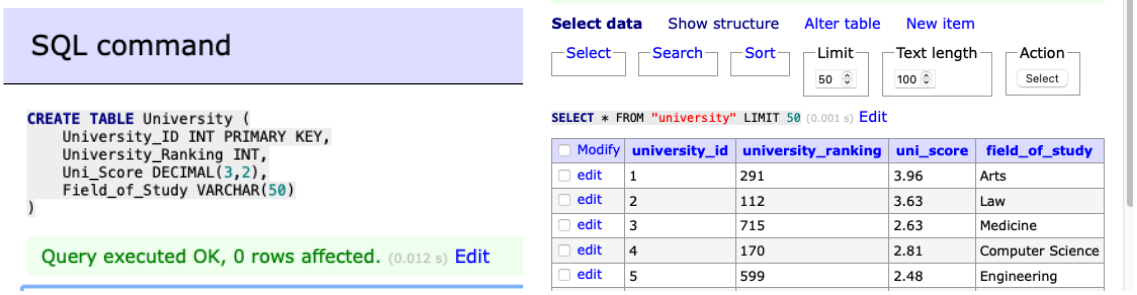
```
SELECT * FROM "student_score" LIMIT 50 (0.002 s) Edit
```

Modify	student_id	score_id	high_school_gpa	sat_score
edit	S00001	1	3.58	1052
edit	S00002	2	2.52	1211
edit	S00003	3	3.42	1193
edit	S00004	4	2.43	1497
edit	S00005	5	2.08	1012
edit	S00006	6	2.40	1600
edit	S00007	7	2.36	1011

total row: 5,000

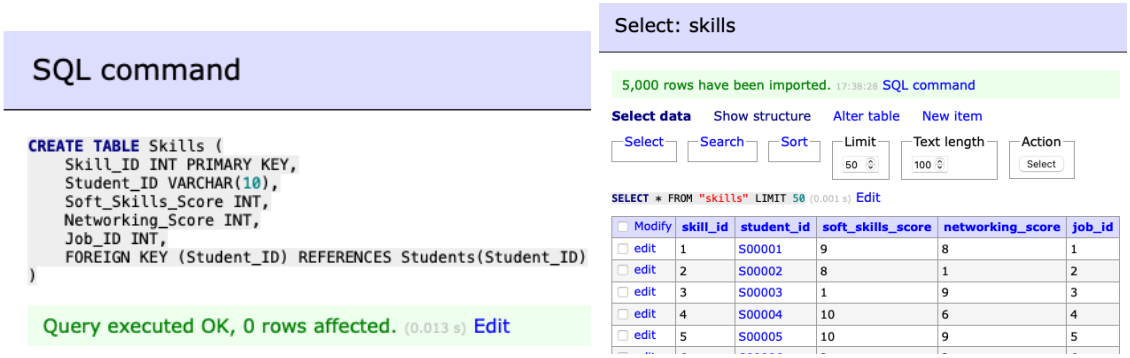
University table:

```
CREATE TABLE University (
        University_ID INT PRIMARY KEY,
        University_Ranking INT,
        Uni_Score DECIMAL(3,2),
        Field_of_Study VARCHAR(50));
```



Skills table:

```
CREATE TABLE Skills (
        Skill_ID INT PRIMARY KEY,
        Student_ID VARCHAR(10),
        Soft_Skills_Score INT,
        Networking_Score INT,
        Job_ID INT, FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID) );
```



Jobs tables

```
CREATE TABLE Jobs (
        Job_ID INT PRIMARY KEY,
        Student_ID VARCHAR(10),
        Starting_Salary INT,
        Career_Satisfaction INT,
        Current_Job_Level VARCHAR(20),
        Work_Life_Balance INT,
        FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID) );
```

SQL command

```
CREATE TABLE Jobs (
    Job_ID INT PRIMARY KEY,
    Student_ID VARCHAR(10),
    Starting_Salary INT,
    Career_Satisfaction INT,
    Current_Job_Level VARCHAR(20),
    Work_Life_Balance INT,
    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID)
)
```

Query executed OK, 0 rows affected. (0.015 s) Edit

Select: jobs

5,000 rows have been imported. 17:42:56 SQL command

Select data | Show structure | Alter table | New item

Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

SELECT * FROM "jobs" LIMIT 50 (0.001 s) Edit

Modify	job_id	student_id	starting_salary	career_satisfaction	current_job_level	work_life_balance
edit	1	S00001	27200	4	Entry	7
edit	2	S00002	25000	1	Mid	7
edit	3	S00003	42400	9	Entry	7
edit	4	S00004	57400	7	Mid	5
edit	5	S00005	47600	9	Entry	2

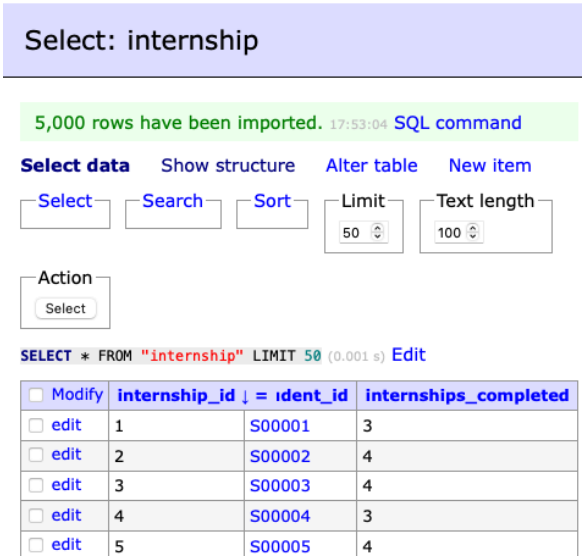
Internship table

```
CREATE TABLE Internship (
        Internship_ID INT PRIMARY KEY,
        Student_ID VARCHAR(10),
        Internships_Completed INT,
        FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID));
```

```
CREATE TABLE Internship (
    Internship_ID INT PRIMARY KEY,
    Student_ID VARCHAR(10),
    Internships_Completed INT,
    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID)
)
```

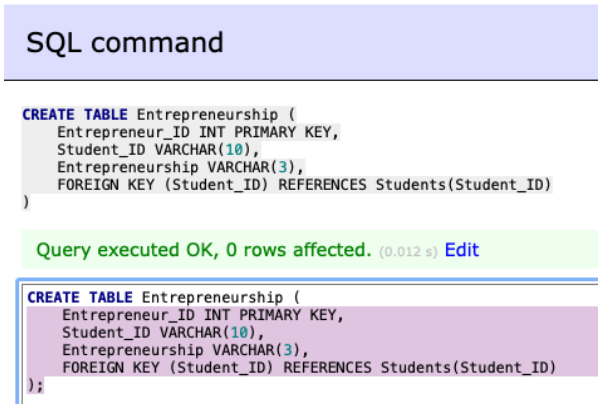
Query executed OK, 0 rows affected. (0.014 s) Edit

```
CREATE TABLE Internship (
    Internship_ID INT PRIMARY KEY,
    Student_ID VARCHAR(10),
    Internships_Completed INT,
    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID)
);
```



Entrepreneurship table:

CREATE TABLE Entrepreneurship (
Entrepreneur_ID INT PRIMARY KEY,
Student_ID VARCHAR(10),
Entrepreneurship VARCHAR(3),
FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID));



Career metrics table

```
CREATE TABLE Career_Metrics (
        Career_Metrics_ID INT PRIMARY KEY,
        Projects_Completed INT,
         Certifications INT,
        Years_to_Promotion INT, Job_Offers INT );
```

SQL command

```
CREATE TABLE Career_Metrics (
    Career_Metrics_ID INT PRIMARY KEY,
    Projects_Completed INT,
    Certifications INT,
    Years_to_Promotion INT,
    Job_Offers INT
)
```

Query executed OK, 0 rows affected.

Select: career_metrics

5,000 rows have been imported. 17:59:49 SQL command

Select data Show structure Alter table New item

Select Search Sort Limit 50 Action Select

```
SELECT * FROM "career_metrics" LIMIT 50 (0.001 s) Edit
```

Modify	career_metrics_id	projects_completed	certifications	years_to_promotion	job_offers
edit	1	7	2	5	5
edit	2	7	3	1	4
edit	3	8	1	3	0
edit	4	9	1	5	1
edit	5	6	4	5	4

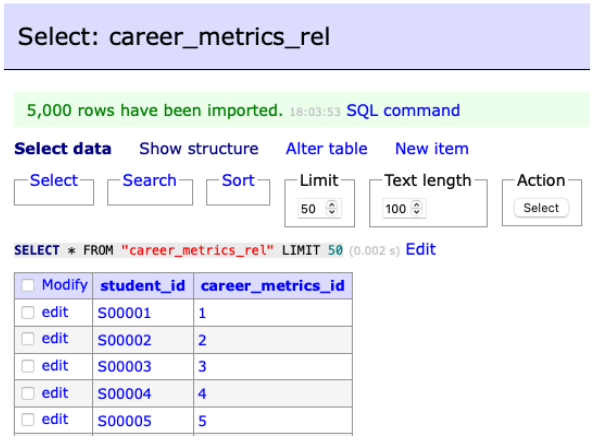
Career metrics rel table

```
CREATE TABLE Career_Metrics_Rel (
        Student_ID VARCHAR(10),
        Career_Metrics_ID INT,
        PRIMARY KEY (Student_ID, Career_Metrics_ID),
        FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID),
        FOREIGN KEY (Career_Metrics_ID) REFERENCES Career_Metrics(Career_Metrics_ID) );
```

SQL command

```
CREATE TABLE Career_Metrics_Rel (
    Student_ID VARCHAR(10),
    Career_Metrics_ID INT,
    PRIMARY KEY (Student_ID, Career_Metrics_ID),
    FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID),
    FOREIGN KEY (Career_Metrics_ID) REFERENCES Career_Metrics(Career_Metrics_ID)
)
```

Query executed OK, 0 rows affected. (0.017 s) Edit



- A brief description of how you populated the database

  The database was populated by importing data from the original file education career success.csv. Separate CSV files were created for each table and then imported into the database as part of the project setup (you can find it in the github project). After creating the schema using CREATE TABLE queries, primary keys and foreign keys were added directly in the table creation statements to ensure relational integrity and consistency. Data was imported following the guidelines provided in document 0 from the course.

- For each of your 20 questions, the question in English, its translation to SQL, and screenshots of the first 5 rows (or less if needed) of the output of your SQL query. (If you need to change any of your original questions), also, list the originals and why you needed to change or replace them.

  The questions were revised based on the feedback from the professor since some were simple. They were adjusted to match the difficulty level of the project and to ensure they covered a range of SQL concepts and complexity.

  1. **Find students with a high school GPA above 3.5.**

```
SELECT s.Student_ID, s.age, s.gender, ss.High_School_GPA
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
WHERE ss.High_School_GPA > 3.5
```

```
SELECT s.Student_ID, s.age, s.gender, ss.High_School_GPA
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
WHERE ss.High_School_GPA > 3.5
```

student_id	age	gender	high_school_gpa
S00001	24	Male	3.58
S00017	20	Female	3.73
S00019	22	Female	3.72
S00026	22	Female	3.65
S00028	29	Female	3.81

Total: 1,233 rows

2. **Create a query to show students who scored above 1200 on the SAT, displaying age, gender, and field of study.**

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
JOIN University u ON s.university_id = u.University_ID
WHERE ss.SAT_Score > 1200;
```

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
JOIN University u ON s.university_id = u.University_ID
WHERE ss.SAT_Score > 1200
```

student_id	age	gender	field_of_study
S00002	21	Other	Law
S00004	25	Male	Computer Science
S00006	24	Male	Law
S00009	24	Male	Business
S00010	28	Male	Computer Science
S00013	22	Female	Engineering

Total: 2,873 rows

3. **Create a query that shows fields of study in descending order by average GPA of students.**

```
SELECT u.Field_of_Study, AVG(ss.High_School_GPA) AS Avg_GPA
FROM University u
JOIN Students s ON u.University_ID = s.university_id
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
GROUP BY u.Field_of_Study
ORDER BY Avg_GPA DESC;
```

```
SELECT u.Field_of_Study, AVG(ss.High_School_GPA) AS Avg_GPA
FROM University u
JOIN Students s ON u.University_ID = s.university_id
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
GROUP BY u.Field_of_Study
ORDER BY Avg_GPA DESC
```

field_of_study	avg_gpa
Mathematics	3.0167785234899329
Arts	3.0166355140186916
Business	3.0164394993045897
Medicine	3.0036139332365747
Law	2.9820770288858322
Computer Science	2.9721194029850746
Engineering	2.9676604850213980

7 rows (0.024 s) Edit, Explain, Export

Total: 7 rows

4. **List students who attended universities ranked in the top 100.**

```
SELECT s.Student_ID, s.age, s.gender, u.University_Ranking
FROM Students s
JOIN University u ON s.university_id = u.University_ID
WHERE u.University_Ranking <= 100
```

student_id	age	gender	university_ranking
S00015	25	Male	3
S00037	29	Female	27
S00042	21	Female	14
S00043	26	Female	98
S00073	19	Male	39
S00079	19	Male	30

Total: 511 rows

5. **Find students majoring in Computer Science.**

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study
FROM Students s
JOIN University u ON s.university_id = u.University_ID
WHERE u.Field_of_Study = 'Computer Science'
```

student_id	age	gender	field_of_study
S00004	25	Male	Computer Science
S00007	27	Male	Computer Science
S00008	20	Male	Computer Science
S00010	28	Male	Computer Science
S00016	25	Female	Computer Science
S00039	20	Female	Computer Science

Total: 670 rows

6. **Retrieve students who have completed at least 2 internships.**

```
SELECT s.Student_ID, s.age, s.gender, i.Internships_Completed
FROM Students s
JOIN Internship i ON s.Student_ID = i.Student_ID
WHERE i.Internships_Completed >= 2
```

student_id	age	gender	internships_completed
S00001	24	Male	3
S00002	21	Other	4
S00003	28	Female	4
S00004	25	Male	3
S00005	22	Male	4

Total: 2,962 rows

7. **Find the best student by GPA in each department.**

SELECT DISTINCT ON (u.Field_of_Study)
u.Field_of_Study, s.Student_ID, ss.High_School_GPA
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
JOIN University u ON s.university_id = u.University_ID
ORDER BY u.Field_of_Study, ss.High_School_GPA DESC

```
SELECT DISTINCT ON (u.Field_of_Study)
    u.Field_of_Study, s.Student_ID, ss.High_School_GPA
FROM Students s
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
JOIN University u ON s.university_id = u.University_ID
ORDER BY u.Field_of_Study, ss.High_School_GPA DESC
```

field_of_study	student_id	high_school_gpa
Arts	S04105	4.00
Business	S03408	3.99
Computer Science	S04660	4.00
Engineering	S02075	4.00
Law	S03618	4.00
Mathematics	S04753	4.00
Medicine	S04796	4.00

7 rows (0.041 s) Edit, Explain, Export

8. **Give me the average salary of each department.**

SELECT u.Field_of_Study, **AVG**(j.Starting_Salary) AS Avg_Salary
FROM Jobs j
JOIN Students s ON j.Student_ID = s.Student_ID
JOIN University u ON s.university_id = u.University_ID
GROUP BY u.Field_of_Study

```
SELECT u.Field_of_Study, AVG(j.Starting_Salary) AS Avg_Salary
FROM Jobs j
JOIN Students s ON j.Student_ID = s.Student_ID
JOIN University u ON s.university_id = u.University_ID
GROUP BY u.Field_of_Study
```

field_of_study	avg_salary
Arts	51422.830440587450
Medicine	50219.158200290276
Law	50081.155433287483
Business	50262.169680111266
Mathematics	50725.906040268456
Computer Science	50777.164179104478
Engineering	50416.547788873039

7 rows (0.025 s) Edit, Explain, Export

9. Create a query that shows career satisfaction by field of study.

```
SELECT u.Field_of_Study, AVG(j.Career_Satisfaction) AS Avg_Career_Satisfaction
FROM Jobs j
JOIN Students s ON j.Student_ID = s.Student_ID
JOIN University u ON s.university_id = u.University_ID
GROUP BY u.Field_of_Study
```

field_of_study	avg_career_satisfaction
Arts	5.6074766355140187
Medicine	5.6676342525399129
Law	5.6781292984869326
Business	5.2892906815020862
Mathematics	5.7033557046979866
Computer Science	5.5835820895522388
Engineering	5.5121255349500713

7 rows (0.022 s) Edit, Explain, Export

10. Create a query with the 10 students who have completed the most projects.

```
SELECT s.Student_ID, cm.Projects_Completed
FROM Career_Metrics cm
JOIN Career_Metrics_Rel cmr ON cm.Career_Metrics_ID = cmr.Career_Metrics_ID
JOIN Students s ON cmr.Student_ID = s.Student_ID
ORDER BY cm.Projects_Completed DESC
LIMIT 10
```

student_id	projects_completed
S00060	9
S00069	9
S00042	9
S00055	9
S00027	9
S00025	9
S00004	9
S00041	9
S00053	9
S00077	9

10 rows (0.027 s) Edit, Explain, Export

11. **Give me the average gender by field of study and show the average salary by gender.**

```
SELECT
   u.Field_of_Study,
   s.gender,
   COUNT(s.Student_ID) AS student_count,
   AVG(j.Starting_Salary) AS avg_salary
FROM Students s
JOIN University u ON s.university_id = u.University_ID
LEFT JOIN Jobs j ON s.Student_ID = j.Student_ID
GROUP BY u.Field_of_Study, s.gender
```

```
SELECT
    u.Field_of_Study,
    s.gender,
    COUNT(s.Student_ID) AS student_count,
    AVG(j.Starting_Salary) AS avg_salary
FROM Students s
JOIN University u ON s.university_id = u.University_ID
LEFT JOIN Jobs j ON s.Student_ID = j.Student_ID
GROUP BY u.Field_of_Study, s.gender
```

field_of_study	gender	student_count	avg_salary
Arts	Female	357	50668.627450980392
Law	Female	361	50449.030470914127
Medicine	Male	345	49892.463768115942
Business	Other	25	51896.000000000000
Engineering	Female	322	50166.459627329193

Total: 21 rows

12. **Create a query by field of study with the current job level.**

```
SELECT
    u.Field_of_Study,
    j.Current_Job_Level,
    COUNT(s.Student_ID) AS num_students
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Jobs j ON s.Student_ID = j.Student_ID
GROUP BY u.Field_of_Study, j.Current_Job_Level
ORDER BY u.Field_of_Study
```

field_of_study	current_job_level	num_students
Arts	Entry	376
Arts	Senior	107
Arts	Mid	227
Arts	Executive	39
Business	Senior	115
Business	Mid	231
Business	Entry	348
Business	Executive	25

Total: 28 rows

13. Get students with a starting salary above $50,000.

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, j.Starting_Salary
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Jobs j ON s.Student_ID = j.Student_ID
WHERE j.Starting_Salary > 50000
```

student_id	age	gender	field_of_study	starting_salary
S00004	25	Male	Computer Science	57400
S00006	24	Male	Law	68400
S00007	27	Male	Computer Science	55500
S00009	24	Male	Business	68900
S00010	28	Male	Computer Science	58900
S00014	21	Male	Arts	76500
S00015	25	Male	Business	61100
S00017	20	Female	Law	97500

Total: 2,541 rows

14. Find students who rated their career satisfaction as 5.

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, j.Career_Satisfaction
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Jobs j ON s.Student_ID = j.Student_ID
WHERE j.Career_Satisfaction = 5
```

student_id	age	gender	field_of_study	career_satisfaction
S00023	23	Female	Engineering	5
S00028	29	Female	Engineering	5
S00035	28	Female	Engineering	5
S00047	24	Male	Engineering	5
S00049	26	Male	Business	5
S00050	24	Male	Law	5

Total: 497 rows

15. List students promoted within 3 years, showing age, gender, and field of study.

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, cm.Years_to_Promotion
FROM Students s
JOIN University u ON s.university_id = u.University_ID
```

JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE cm.Years_to_Promotion <= 3

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, cm.Years_to_Promotion
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE cm.Years_to_Promotion <= 3
```

student_id	age	gender	field_of_study	years_to_promotion
S00002	21	Other	Law	1
S00003	28	Female	Medicine	3
S00006	24	Male	Law	2
S00008	20	Male	Computer Science	3
S00009	24	Male	Business	2
S00010	28	Male	Computer Science	2

Total: 2,965 rows

16. Get students with a work-life balance score of 8 or higher.

SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, j.Work_Life_Balance
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Jobs j ON s.Student_ID = j.Student_ID
WHERE j.Work_Life_Balance >= 8

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, j.Work_Life_Balance
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Jobs j ON s.Student_ID = j.Student_ID
WHERE j.Work_Life_Balance >= 8
```

student_id	age	gender	field_of_study	work_life_balance
S00006	24	Male	Law	8
S00013	22	Female	Engineering	8
S00017	20	Female	Law	9
S00020	19	Female	Mathematics	8
S00021	25	Female	Medicine	9
S00025	29	Female	Engineering	10

Total: 1,502 rows

17. Find students who are entrepreneurs, showing gender, age, and field of study.

SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Entrepreneurship e ON s.Student_ID = e.Student_ID
WHERE e.Entrepreneurship = 'Yes'

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Entrepreneurship e ON s.Student_ID = e.Student_ID
WHERE e.Entrepreneurship = 'Yes'
```

student_id	age	gender	field_of_study
S00006	24	Male	Law
S00012	25	Female	Law
S00015	25	Male	Business
S00035	28	Female	Engineering
S00039	20	Female	Computer Science
S00040	29	Female	Business
S00044	20	Male	Mathematics
S00045	22	Male	Mathematics

Total: 1,008 rows

18. Retrieve students who meet both GPA > 3.8 and SAT > 1300.

SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, ss.High_School_GPA, ss.SAT_Score
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
WHERE ss.High_School_GPA > 3.8 **AND** ss.SAT_Score > 1300

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, ss.High_School_GPA, ss.SAT_Score
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Student_Score ss ON s.Student_ID = ss.Student_ID
WHERE ss.High_School_GPA > 3.8 AND ss.SAT_Score > 1300
```

student_id	age	gender	field_of_study	high_school_gpa	sat_score
S00037	29	Female	Engineering	3.88	1321
S00180	24	Male	Engineering	3.91	1305
S00183	24	Male	Arts	3.86	1547
S00184	18	Female	Business	3.86	1368
S00218	18	Male	Business	3.85	1559
S00231	20	Male	Business	3.82	1573

Total: 225 rows

19. Find students with internships, certifications, and projects all above 5.

SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, i.Internships_Completed, cm.Certifications, cm.Projects_Completed
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Internship i ON s.Student_ID = i.Student_ID
JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE i.Internships_Completed > 5
AND cm.Certifications > 5

```
AND cm.Projects_Completed > 5
```

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, i.Internships_Completed, cm.Certifications, cm.Projects_Complete
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Internship i ON s.Student_ID = i.Student_ID
JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE i.Internships_Completed > 5
AND cm.Certifications > 5
AND cm.Projects_Completed > 5
```

No rows.

20. Get students majoring in STEM fields with a job offer count above 4.

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, cm.Job_Offers
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE u.Field_of_Study IN ('Computer Science', 'Engineering', 'Mathematics', 'Physics', 'Biology', 'Chemistry')
AND cm.Job_Offers > 4
```

```
SELECT s.Student_ID, s.age, s.gender, u.Field_of_Study, cm.Job_Offers
FROM Students s
JOIN University u ON s.university_id = u.University_ID
JOIN Career_Metrics_Rel cmr ON s.Student_ID = cmr.Student_ID
JOIN Career_Metrics cm ON cmr.Career_Metrics_ID = cm.Career_Metrics_ID
WHERE u.Field_of_Study IN ('Computer Science', 'Engineering', 'Mathematics', 'Physics', 'Biology', 'Chemistry')
AND cm.Job_Offers > 4
```

student_id	age	gender	field_of_study	job_offers
S00011	28	Female	Mathematics	5
S00018	23	Male	Mathematics	5
S00019	22	Female	Mathematics	5
S00046	20	Female	Engineering	5
S00059	22	Female	Computer Science	5
S00062	29	Male	Engineering	5
S00075	23	Female	Mathematics	5

Total: 346 rows

- Description of data verification, integrity verification, and how data cleaning and processing were conducted.
    - **Data Verification**
      Verified that data types are consistent across tables (e.g., INT, VARCHAR). Checked that each primary key and foreign key is correctly assigned and references valid data. Confirmed that foreign keys in child tables match the data type of primary keys in parent tables. Ensured data consistency in linked tables (e.g., Student_ID in Skills matches Student_ID in Students).
    - **Integrity Verification**
      Enforced PRIMARY KEY and FOREIGN KEY constraints to maintain data integrity. Tested ON DELETE CASCADE and ON UPDATE CASCADE for

consistent behavior. Ensured no orphaned records by confirming that referenced keys exist in parent tables. The original table (education career success.csv) did not contain all the necessary primary keys for creating a relational database. Therefore, new primary keys were created for some tables to align with relational database requirements.

- **Data Cleaning and Processing**
  Removed duplicates and fixed incorrect formatting. Corrected inconsistent capitalization and data types. Standardized numerical values and adjusted text format consistency.This process ensured that the database schema and data are accurate, consistent, and optimized for querying.

Grading

Submission part 3 is worth 70 points. Grading is based on:

- Correct and efficient translation of questions from Part 1 into queries.
- Demonstrates a variety of schema and SQL features (variety of data types, keys, foreign keys, other constraints, variety of types of queries - joins, aggregates, selects, subqueries)
- Effectiveness of data cleaning and preprocessing approaches.
- Effectiveness of data entry - were you able to avoid large amounts of manual data entry?
- Organization of your write-up.

References:

Shamim, A. (n.d.). *Education and Career Success* [Dataset]. Kaggle. Retrieved February 15, 2025, from https://www.kaggle.com/datasets/adilshamim8/education-and-career-success