1. Analyze the requirements of the organization (what is needed for this organization and database)

Our organization is focused on providing educational information for Mainland China undergraduate students who wish to study abroad for graduate learning. Therefore, we designed EduSpark which is a graduate programs application database. The database should store information about universities, their programs, standardized tests, employment outcomes, and applicant details. Based on the information, students can get the information they want and also get the data analysis provided by us. We want to offer application support and suggestions to students. Therefore, the database should be able to manage application details, including the application status and necessary documents. We believe that with our help, there’s no obstacles in graduate program application!

1. Identify the relevant entities, attributes, and relationships together with any constraints and properties

Entities:

* University
* Uni\_maj (University Majors)
* Program
* Standardized\_Test
* Employment
* Undergra\_univers (Undergraduate Universities)
* Appliers
* Applications

Attributes:

* University: UID, University\_name, Abbreviation, Branch\_unniversity, Region, Location, QSranking
* Uni\_maj: Rank, UID, MID, Major\_name, Major\_type, Teaching\_language, Major\_scale
* Program: Program\_ID, University\_ID, Program\_Name, Early\_DDL, Final\_DDL, Program\_Link, GPA, Recommendation\_Min\_Amount, Recommendation\_Requirement, Tuition\_Fee
* Standardized\_Test: Program\_ID, GRE\_Required, GRE\_Cut\_Off\_Point, TOEFL\_Accepted, TOEFL\_Requirement, IELTS\_Accepted, IELTS\_Requirement
* Employment: Program\_ID, Avg\_Salary\_Per\_Year, Employment\_Rate, Employer\_Satisfaction, Company, Industry
* Undergra\_univers: univer\_id, univer\_name, univer\_city
* Appliers: applier\_id, first\_name, last\_name, email\_address, phone\_number, gender, date\_of\_birth, undergraduate\_program, univer\_id, GPA, toefl\_score, ielts\_score, gre\_score, gmat\_score
* Applications: program\_id, applier\_id, status, date, recom\_id, ps\_id

Relationships:

* University (1) -- (N) Uni\_maj
* University (1) -- (N) Program
* Program (1) -- (1) Standardized\_Test
* Program (1) -- (1) Employment
* Undergra\_univers (1) -- (N) Appliers
* Program (N) -- (N) Appliers (through Applications)

Constraints:

* Primary keys and foreign keys are clearly specified in the schema.
* On delete cascade for foreign keys to maintain referential integrity.

1. Convert the E-R diagrams to relational schemas (clearly indicating the primary keys, foreign keys, functional and/or multivalued dependencies, as well as justifying that your designs are in good, normalized form)

program

PK: Program\_ID

FK: university\_UID

All attributes depend on the primary key, which is Program\_ID.

There are no transitive dependencies, as each non-key attribute is directly dependent on Program\_ID.

Therefore, the 'program' table is in 3NF and possibly even in higher normal forms.

The primary key is Program\_ID, and all other attributes are dependent on it.

There are no multivalued dependencies in this table.

standardized\_test

PK: Program\_ID

FK: Program\_ID

All attributes depend on the primary key, which is Program\_ID.

There are no transitive dependencies, as each non-key attribute is directly dependent on Program\_ID.

Therefore, the 'standardized\_test' table is in 3NF and possibly even in higher normal forms.

The primary key is Program\_ID, and all other attributes are dependent on it.

There are no multivalued dependencies in this table.

employment

PK: Program\_ID

FK: Program\_ID

All attributes depend on the primary key, which is Program\_ID.

There are no transitive dependencies, as each non-key attribute is directly dependent on Program\_ID.

Therefore, the 'employment' table is in 3NF and possibly even in higher normal forms.

The primary key is Program\_ID, and all other attributes are dependent on it.

There are no multivalued dependencies in this table.

1. Index and hashing

Indexes:

* University(University\_name): Indexing the University\_name will speed up queries that filter or join based on the university name.
* Program(Program\_Name): Indexing the Program\_Name can speed up queries filtering or joining based on the program name.
* Appliers(GPA): Indexing the GPA attribute can improve the performance of queries that filter or sort applicants by their GPA.
* Applications(program\_id, applier\_id): Indexing the primary key attributes of the Applications table can enhance the performance of queries that involve filtering or joining based on program\_id and/or applier\_id.

Hashing:

* Appliers(email\_address): Hashing email addresses is a good practice to protect the privacy of applicants in case the database is breached.