Huskies Job Search Matching Platform

Nicholas Wong, Peisong Yuan, Xuanhao Wu

Northeastern University, Boston, MA, USA

Abstract

Northeastern University students (Huskies) face a complicated job-search environment where they must match job opportunities to their academic background, skills, and preferences. This end-to-end process requires a structural dataset and a smart filtering function. Our project was to develop a comprehensive relational database that aims to structure Huskies' job-seeking system model by reference to ideas of NUworks and Linkedin, integrating detailed information of users, positions, skills, applications, and companies to mock real-life application processes.

This database enables many operational scenarios, from basic filtering to multi-condition logic and automatic system functions. The basic application is a basic inquiry function for user search specific position from expectation and preference. Also include complicated triggers that validate eligibility before inserting any new application record. Those functions illustrate the realism of database design.

To support these functions, we designed a normalized multi-table schema in MySQL, generated data using the Faker library, and finally built a query-driven data model that highlights how it supports users and employers to make decisions.

Introduction

Finding the "right" job or co-op is a big challenge for many NU students. We're constantly bouncing between NUworks, LinkedIn, and company websites, trying to match our majors, skills, GPA, and interests to hundreds of openings. But most of these platforms are basically black boxes; we can click filters, yet we never see the underlying data model or understand how the matching logic actually works.

In this project, we designed a relational database that models a simplified version of the NU job-search ecosystem. Our schema connects students, companies, positions, skills, and applications in a normalized way so we can support realistic queries and system behaviors in MySQL. With our data and sample use cases, we can do things like search for positions that fit a student's major and skills, check eligibility before inserting a new application, count applicants per role, and rank candidates by GPA for a company's openings. Overall, this database is meant to make the job-matching process more transparent, flexible, and useful for both students and employers.

Database Design

Our database design models the Northeastern University job-search ecosystem by normalizing relationships between Users(Student), Companies, Positions, Skills, and Applications (as shown in Figure 1). All tables were created in MySQL Workbench, and using the Reverse Engineer function to generate the final EER diagram. This ensures the diagram reflects the real model and contains the same constraints, primary keys, foreign keys and relationships. This MySQL setup script is then executed by a jupyter notebook file to create the database in the repo (using a MySQL server).

Key Entities:

1. Users:

Users table stored all student information that used to match position. It includes academic backgrounds like major, GPA, preference, contact information and so on. Every user can have multiple skills and can apply for multiple applications. The primary key is user_id, holding one-to-many relationship with Application, and many-to-many relationship with Skills via User_Skill.

2. Company:

The Company table has employers information who post co-op, internship, or full-time work type, storing company size, website, rating, headquarter location and so on. The primary key is company_id, keeping a one-to-many relationship with the Positions table.

3. Positions:

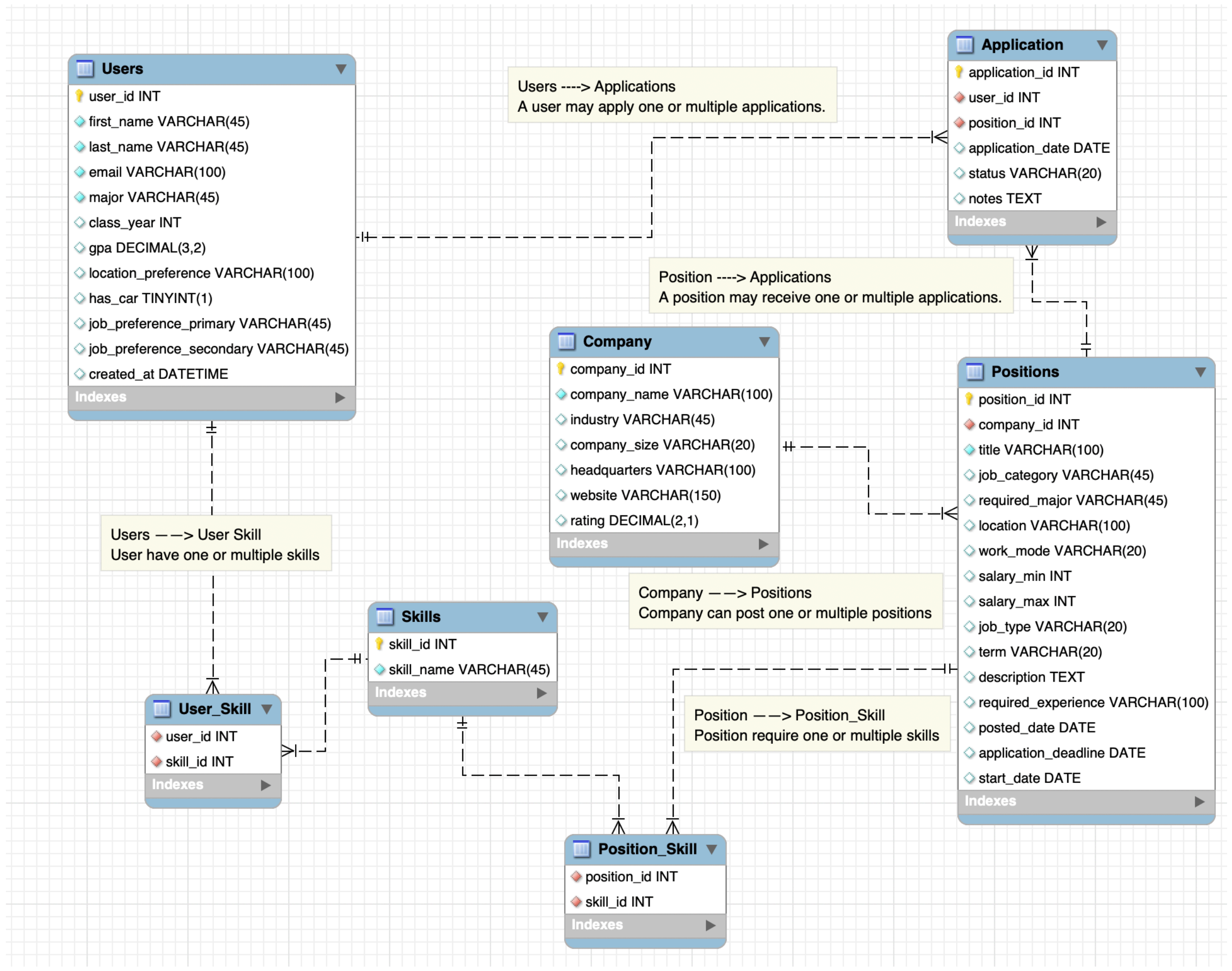
The Positions table represents all position information including job type, required major, required work experience, term, work types, location, due date etc. Each position can require one or multiple skills, and may receive one or multiple applications. The primary key is position_id and foreign key is company_id. Positions has many-to-many relationships with Skills via Position_skill and one-to-many relationship with application.

4. Application:

The Application table records every student's application information to position(s). This table tracks application date and status. The table structure makes sure every application belongs to exactly one user and one position. The primary key is application_id, and foreign keys are user_id, and position_id.

These entities form an end-to-end structure that support real life job search behaviors. The connection between Users, Skills, Application, Positions ensure the database can handle different user cases and automotive functions that will show later parts.

Figure 1



Data Sources and Methods

We generate our data using Python packages called “Faker”[1] and “Random”[2] Faker allows us to simulate real users and companies to be used in our example use cases.

Data Generation: To ensure reproducibility, we seed both the Faker generator as well as the random generator with a seed value of 32. This makes our output completely deterministic and fully replicable across servers and between runs.

Data Structure: For categorical fields (like major, industries, job categories, etc.), we create a predefined array of common entries and randomly assign them using the random library. For other, more simple data points such as dates and names (using fake.name()), we use Faker to create realistic, but randomized data.

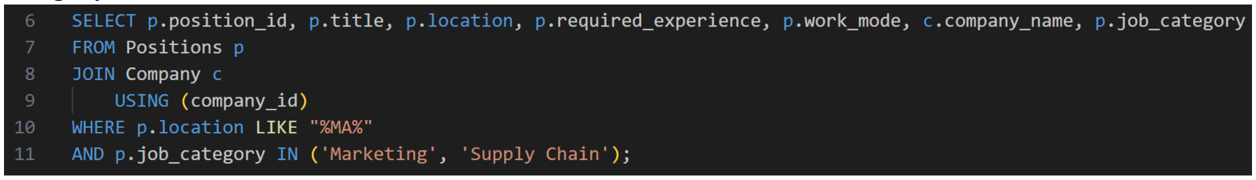
Assumptions: All users are assumed to be job-seeking students. Application dates fall within a (give or take) 6 month window.

No data cleaning was necessary as it was generated and not sourced.

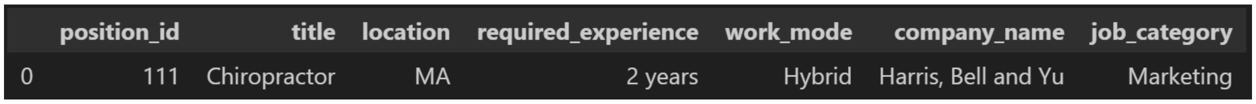
The steps to create the same database is clearly described in the github repo, where anyone can clone the repo and create the same exact database themselves following the steps in the README file.

User Cases (Application Prototype)

Use Case 1: Jack is seeking a co-op in marketing or supply chain and is open to roles anywhere in Massachusetts.
This use case describes a common user looking for roles with a location and multiple job category restrictions.

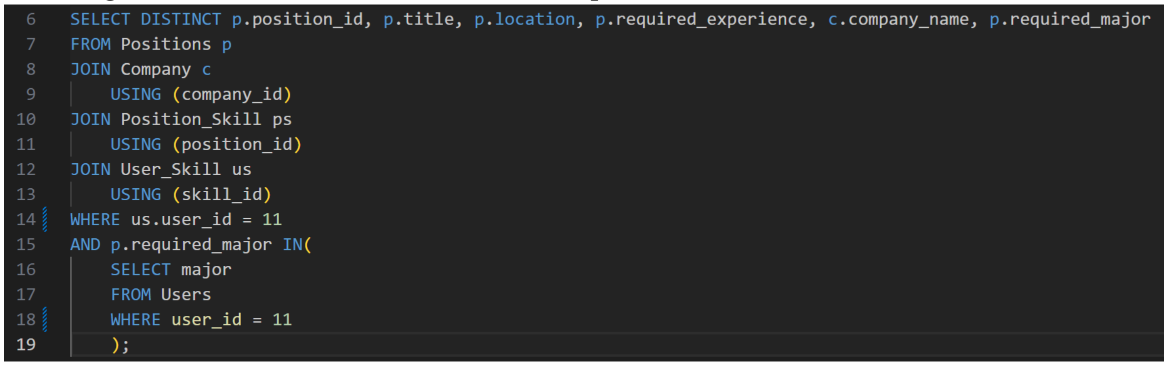


```sql
SELECT p.position_id, p.title, p.location, p.required_experience, p.work_mode, c.company_name, p.job_category
FROM Positions p
JOIN Company c
    USING (company_id)
WHERE p.location LIKE "%MA%"
AND p.job_category IN ('Marketing', 'Supply Chain');
```

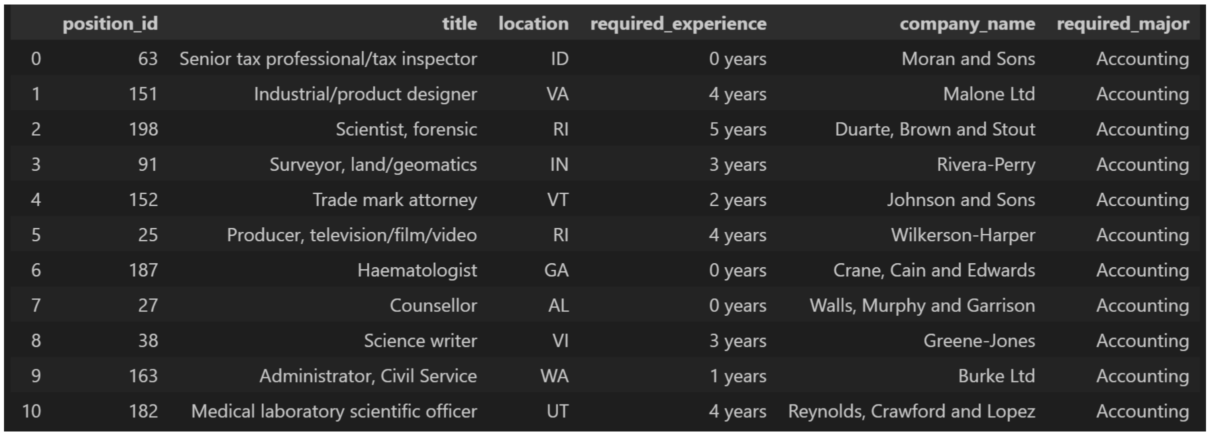


	position_id	title	location	required_experience	work_mode	company_name	job_category
0	111	Chiropractor	MA	2 years	Hybrid	Harris, Bell and Yu	Marketing

Use Case 2: Craig is a student majoring in Accounting, and he wants to find positions that match both his major and at least one of his technical skills (Python, Excel, etc.).
This use case describes a common user filtering their job search with a required major as well as matching one or more of his skills with the required skills.

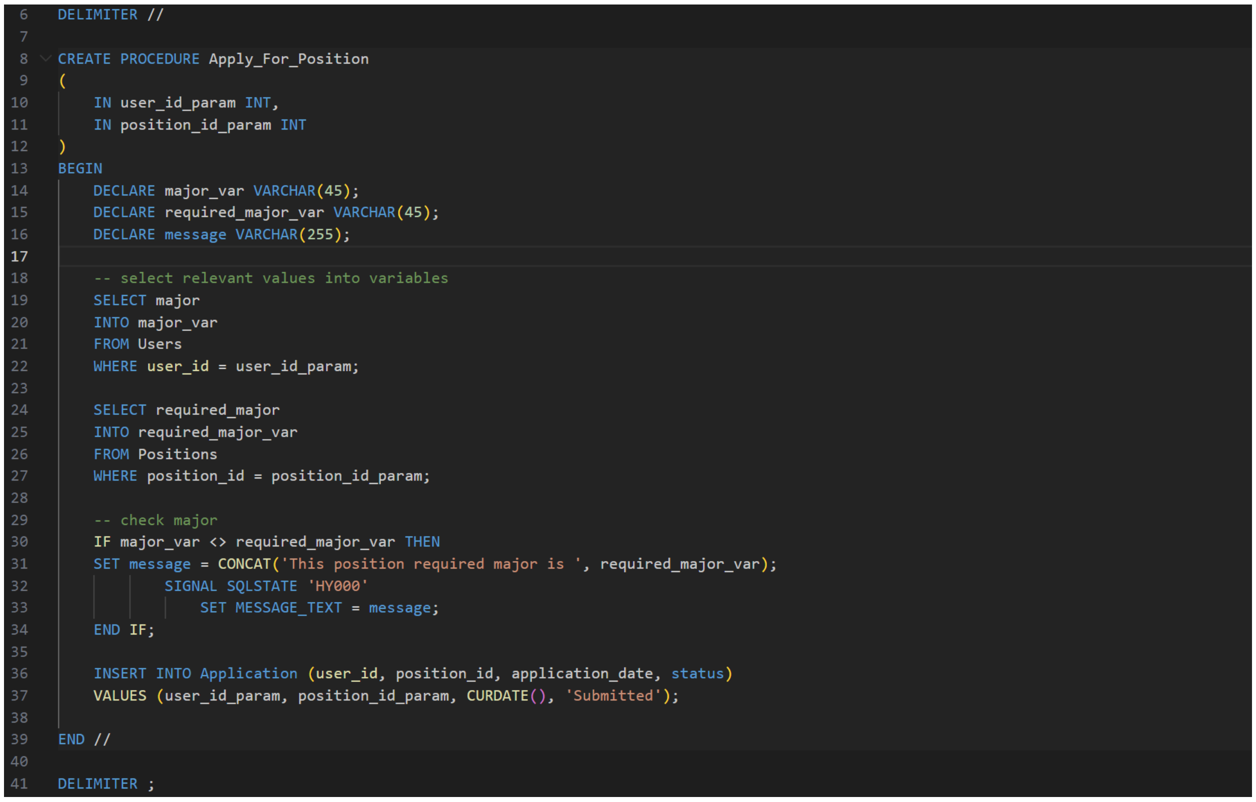


```sql
SELECT DISTINCT p.position_id, p.title, p.location, p.required_experience, c.company_name, p.required_major
FROM Positions p
JOIN Company c
    USING (company_id)
JOIN Position_Skill ps
    USING (position_id)
JOIN User_Skill us
    USING (skill_id)
WHERE us.user_id = 11
AND p.required_major IN(
    SELECT major
    FROM Users
    WHERE user_id = 11
    );
```

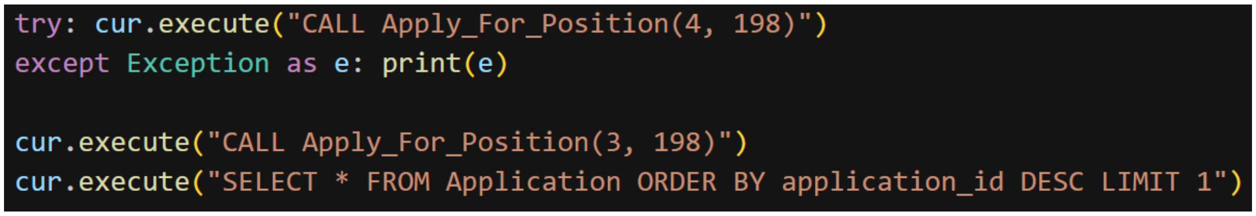


	position_id	title	location	required_experience	company_name	required_major
0	63	Senior tax professional/tax inspector	ID	0 years	Moran and Sons	Accounting
1	151	Industrial/product designer	VA	4 years	Malone Ltd	Accounting
2	198	Scientist, forensic	RI	5 years	Duarte, Brown and Stout	Accounting
3	91	Surveyor, land/geomatics	IN	3 years	Rivera-Perry	Accounting
4	152	Trade mark attorney	VT	2 years	Johnson and Sons	Accounting
5	25	Producer, television/film/video	RI	4 years	Wilkerson-Harper	Accounting
6	187	Haematologist	GA	0 years	Crane, Cain and Edwards	Accounting
7	27	Counsellor	AL	0 years	Walls, Murphy and Garrison	Accounting
8	38	Science writer	VI	3 years	Greene-Jones	Accounting
9	163	Administrator, Civil Service	WA	1 years	Burke Ltd	Accounting
10	182	Medical laboratory scientific officer	UT	4 years	Reynolds, Crawford and Lopez	Accounting

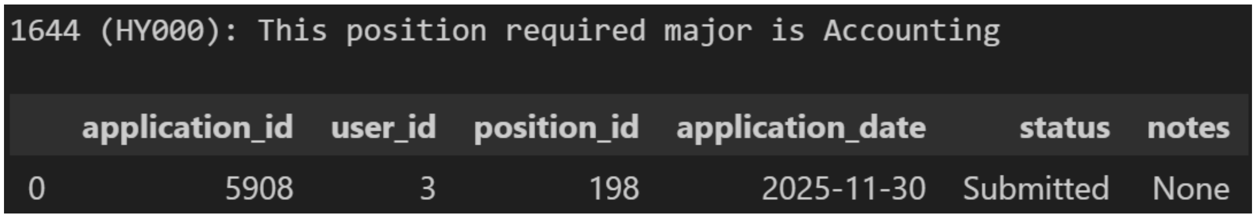
Use Case 3: When a student selects a position and clicks "Apply," the system should verify that the student satisfies the major requirements, and that they have not already applied. If all conditions are met, the procedure should insert a new application record; otherwise, it returns a clear message explaining why the application cannot be submitted.
This use case simulates the application process of a user, throwing an error if it fails or creating a new application if it is valid. The following is two separate application attempts, one of which fails.



```sql
DELIMITER //

CREATE PROCEDURE Apply_For_Position
(
    IN user_id_param INT,
    IN position_id_param INT
)
BEGIN
    DECLARE major_var VARCHAR(45);
    DECLARE required_major_var VARCHAR(45);
    DECLARE message VARCHAR(255);

    -- select relevant values into variables
    SELECT major
    INTO major_var
    FROM Users
    WHERE user_id = user_id_param;

    SELECT required_major
    INTO required_major_var
    FROM Positions
    WHERE position_id = position_id_param;

    -- check major
    IF major_var <> required_major_var THEN
    SET message = CONCAT('This position required major is ', required_major_var);
            SIGNAL SQLSTATE 'HY000'
                SET MESSAGE_TEXT = message;
    END IF;

    INSERT INTO Application (user_id, position_id, application_date, status)
    VALUES (user_id_param, position_id_param, CURDATE(), 'Submitted');

END //

DELIMITER ;
```



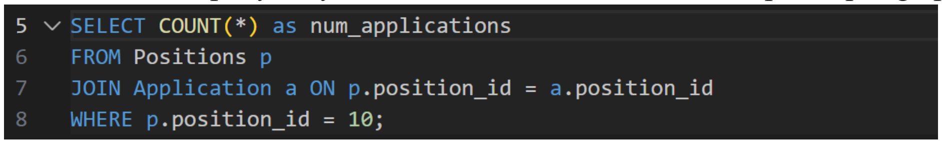
```python
try: cur.execute("CALL Apply_For_Position(4, 198)")
except Exception as e: print(e)

cur.execute("CALL Apply_For_Position(3, 198)")
cur.execute("SELECT * FROM Application ORDER BY application_id DESC LIMIT 1")
```



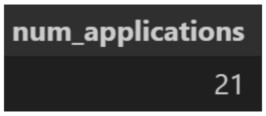
```
1644 (HY000): This position required major is Accounting
```

	application_id	user_id	position_id	application_date	status	notes
0	5908	3	198	2025-11-30	Submitted	None

Use Case 4: A company wants to know the number of applicants to a position.
This use case shows a company who wants to see how many applications to a position they received. A company may use this to determine when to stop accepting applications.

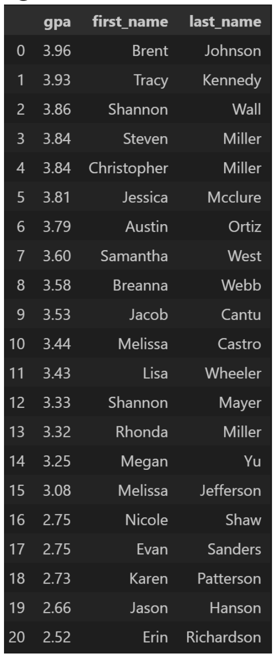


```sql
SELECT COUNT(*) as num_applications
FROM Positions p
JOIN Application a ON p.position_id = a.position_id
WHERE p.position_id = 10;
```

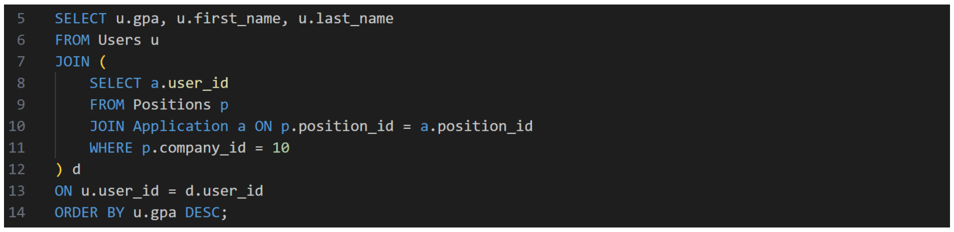


num_applications
21

Use Case 5: XYZ Company wants to know the GPA (DESC order) of students applying to their open positions.
This use case allows a company to sort through all their applicants to all positions by GPA. A company may want to vet candidates based on GPA, which is feasible through this method.

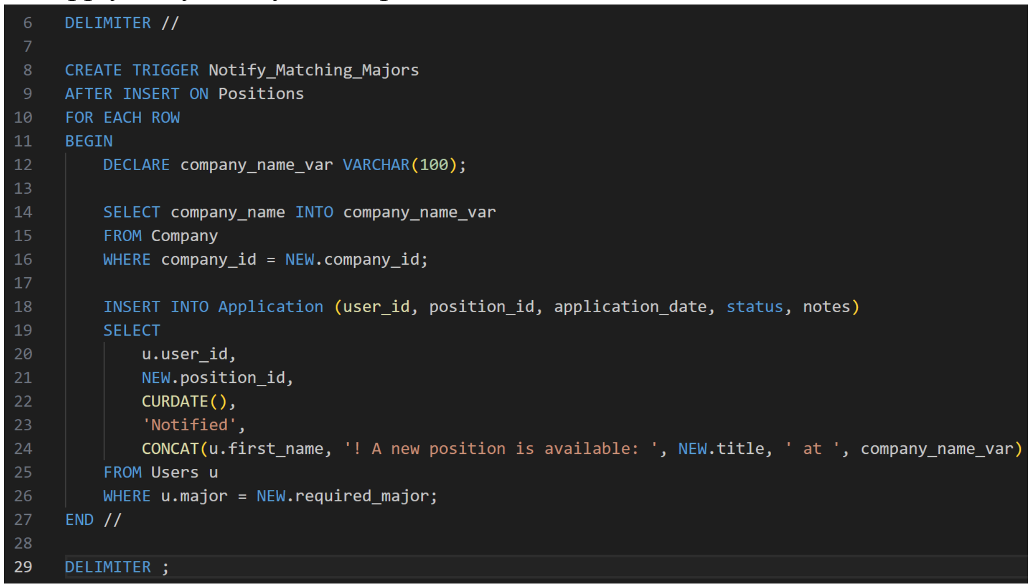


	gpa	first_name	last_name
0	3.96	Brent	Johnson
1	3.93	Tracy	Kennedy
2	3.86	Shannon	Wall
3	3.84	Steven	Miller
4	3.84	Christopher	Miller
5	3.81	Jessica	Mcclure
6	3.79	Austin	Ortiz
7	3.60	Samantha	West
8	3.58	Breanna	Webb
9	3.53	Jacob	Cantu
10	3.44	Melissa	Castro
11	3.43	Lisa	Wheeler
12	3.33	Shannon	Mayer
13	3.32	Rhonda	Miller
14	3.25	Megan	Yu
15	3.08	Melissa	Jefferson
16	2.75	Nicole	Shaw
17	2.75	Evan	Sanders
18	2.73	Karen	Patterson
19	2.66	Jason	Hanson
20	2.52	Erin	Richardson

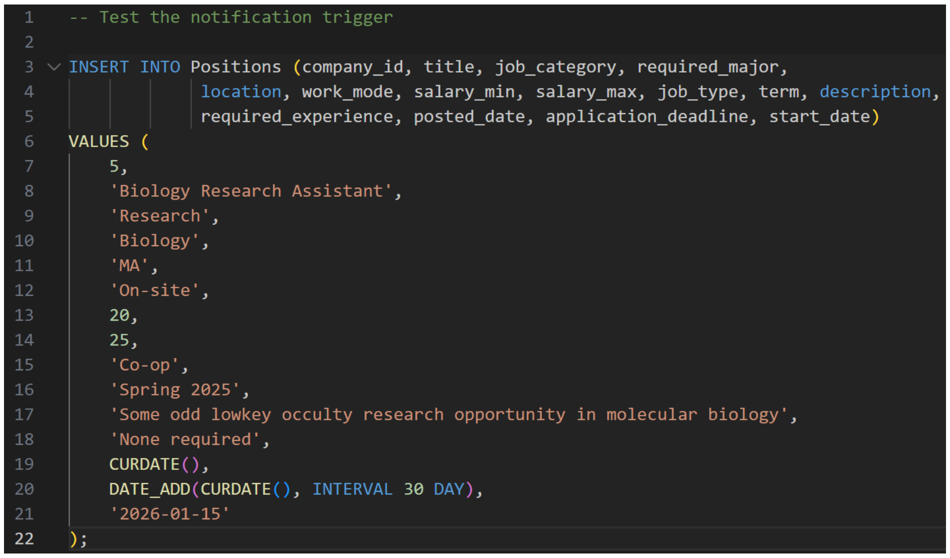


```sql
SELECT u.gpa, u.first_name, u.last_name
FROM Users u
JOIN (
    SELECT a.user_id
    FROM Positions p
    JOIN Application a ON p.position_id = a.position_id
    WHERE p.company_id = 10
) d
ON u.user_id = d.user_id
ORDER BY u.gpa DESC;
```

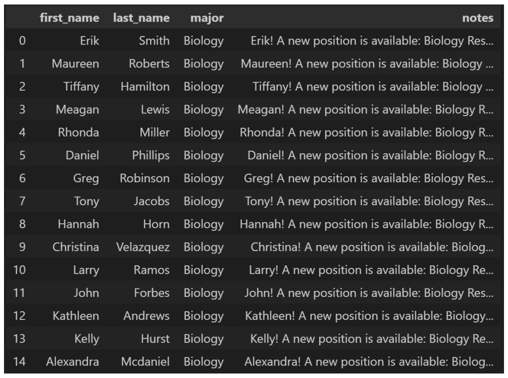
Use Case 6: ABC Company wants to create an application position for Biology majors. This notifies Sean, who is a Biology major looking for a co-op.
This use case is a trigger that activates when a company opens a new position, notifying all students who match the major requirement. This is important for students to be able to get ahead and apply early if they are responsive to their notifications.



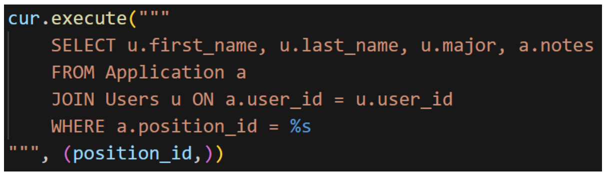
```sql
DELIMITER //

CREATE TRIGGER Notify_Matching_Majors
AFTER INSERT ON Positions
FOR EACH ROW
BEGIN
    DECLARE company_name_var VARCHAR(100);

    SELECT company_name INTO company_name_var
    FROM Company
    WHERE company_id = NEW.company_id;

    INSERT INTO Application (user_id, position_id, application_date, status, notes)
    SELECT
        u.user_id,
        NEW.position_id,
        CURDATE(),
        'Notified',
        CONCAT(u.first_name, '! A new position is available: ', NEW.title, ' at ', company_name_var)
    FROM Users u
    WHERE u.major = NEW.required_major;
END //

DELIMITER ;
```



```sql
-- Test the notification trigger

INSERT INTO Positions (company_id, title, job_category, required_major,
             location, work_mode, salary_min, salary_max, job_type, term, description,
             required_experience, posted_date, application_deadline, start_date)
VALUES (
    5,
    'Biology Research Assistant',
    'Research',
    'Biology',
    'MA',
    'On-site',
    20,
    25,
    'Co-op',
    'Spring 2025',
    'Some odd lowkey occulty research opportunity in molecular biology',
    'None required',
    CURDATE(),
    DATE_ADD(CURDATE(), INTERVAL 30 DAY),
    '2026-01-15'
);
```

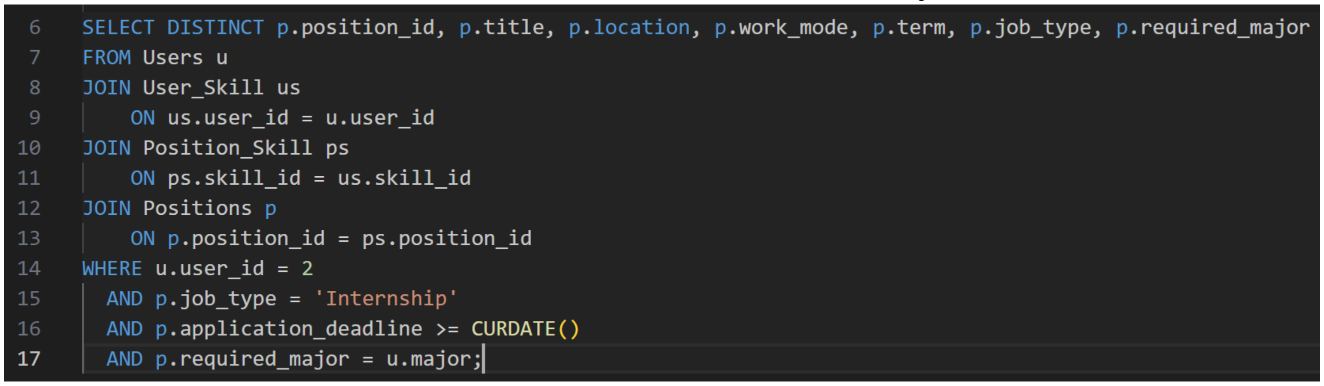


	first_name	last_name	major	notes
0	Erik	Smith	Biology	Erik! A new position is available: Biology Res...
1	Maureen	Roberts	Biology	Maureen! A new position is available: Biology ...
2	Tiffany	Hamilton	Biology	Tiffany! A new position is available: Biology ...
3	Meagan	Lewis	Biology	Meagan! A new position is available: Biology R...
4	Rhonda	Miller	Biology	Rhonda! A new position is available: Biology R...
5	Daniel	Phillips	Biology	Daniel! A new position is available: Biology R...
6	Greg	Robinson	Biology	Greg! A new position is available: Biology Res...
7	Tony	Jacobs	Biology	Tony! A new position is available: Biology Res...
8	Hannah	Horn	Biology	Hannah! A new position is available: Biology R...
9	Christina	Velazquez	Biology	Christina! A new position is available: Biolog...
10	Larry	Ramos	Biology	Larry! A new position is available: Biology Re...
11	John	Forbes	Biology	John! A new position is available: Biology Res...
12	Kathleen	Andrews	Biology	Kathleen! A new position is available: Biology...
13	Kelly	Hurst	Biology	Kelly! A new position is available: Biology Re...
14	Alexandra	Mcdaniel	Biology	Alexandra! A new position is available: Biolog...

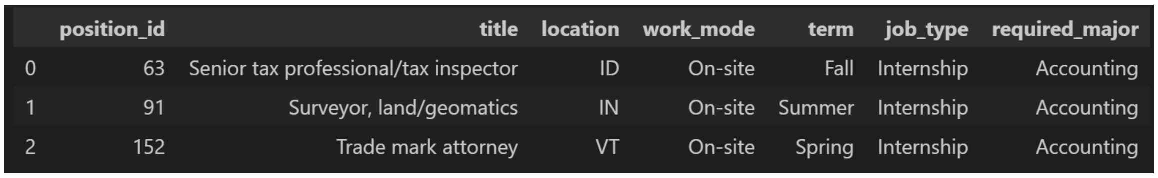


```python
cur.execute("""
    SELECT u.first_name, u.last_name, u.major, a.notes
    FROM Application a
    JOIN Users u ON a.user_id = u.user_id
    WHERE a.position_id = %s
""", (position_id,))
```

Use Case 7: Craig is a third-year student majoring in Accounting. He has skills in Python, SQL, and Tableau. He wants to see open internship positions where his major matches the position's major requirement, and at least one of his skills matches the position's required skills.
This use case adds deadline and skill restrictions to better aid the user's job search.



```sql
SELECT DISTINCT p.position_id, p.title, p.location, p.work_mode, p.term, p.job_type, p.required_major
FROM Users u
JOIN User_Skill us
    ON us.user_id = u.user_id
JOIN Position_Skill ps
    ON ps.skill_id = us.skill_id
JOIN Positions p
    ON p.position_id = ps.position_id
WHERE u.user_id = 2
  AND p.job_type = 'Internship'
  AND p.application_deadline >= CURDATE()
  AND p.required_major = u.major;
```



	position_id	title	location	work_mode	term	job_type	required_major
0	63	Senior tax professional/tax inspector	ID	On-site	Fall	Internship	Accounting
1	91	Surveyor, land/geomatics	IN	On-site	Summer	Internship	Accounting
2	152	Trade mark attorney	VT	On-site	Spring	Internship	Accounting

Conclusions

Overall, our project shows that a well-designed relational database can support a more transparent and flexible job-matching experience for NU students. On the accomplishments side, we were able to store detailed information for every position, including attributes such as location and salary, so users can search and filter roles in a more targeted way. We also implemented automatic logic that checks applications against basic requirements: when a new position is posted that a user can apply to, the system can flag it, and if a student does not meet the requirements, the database will raise an error and reject the application, similar to how NUworks enforces eligibility rules.

At the same time, there are several limitations in our current design and data. Because of relationship constraints and the difficulty of generating more complex application patterns, we assumed that each user applies to only one position in our dataset, which is not realistic in a real job search. Our data is also generated with the Faker library rather than using real student or employer records, so it is less representative, even though it is sufficient for testing our use cases. In addition, we only modeled a limited set of majors and areas, and our automatic matching functions focus mainly on the student's major instead of capturing a broader set of preferences (such as skills, location preferences, or work authorization). These limitations suggest clear directions for future work, but within our current scope, the project successfully demonstrates how a relational database can power a job-matching platform similar to NUworks.

Author Contributions

Our group basically uses an equal division approach to ensure that everyone contributes evenly to our final development of the database system. Initially, we selected a topic together and created a timeline for our project milestones and goals in a shared Google Doc. Everyone designs 3 user cases independently to make sure everyone's idea and expectation will be enabled in the project. We learned from this process that everyone has a different perspective, and makes meaningful inputs for the database. For example, Xuanhao and Peisong are thinking about the user, helping students to find matching positions. In contrast, Nicholas stands at the employer's position to find the valuables of the database.

Although works were divided evenly in many aspects, every team member contributed a specific strength to shape different parts of the project. Xuanhao led the database design process, including creating the initial tables and fundamental structures in MySQL. His work makes sure the database is logical and reasonable, supporting queries and functions we run later. Peisong plays a key role in optimizing table variables, ensuring the model design aligns with the user cases and function requirements. Xuanhao and Peison also work together to organize, design, and finalize the ER diagram that is generated from MySQL Workbench's reverse engineering tool. Nicholas contributes to the project through his Python experiences. He was responsible for generating data using the Faker and Random library and inserting it to our database, making sure every query of the user case can return meaningful results. Nicholas also created and maintained a GitHub repository, integrating and managing our code.

GitHub Repository

The complete source code, data generation scripts, and setup instructions are available at: https://github.com/Quantalope/Job-Search-Project

References

1. Faraglia, D. (n.d.). *Welcome to Faker's documentation!* Welcome to Faker's documentation! - Faker 38.2.0 documentation. https://faker.readthedocs.io/en/master/
2. Python Software Foundation. (n.d.). *Random - generate pseudo-random numbers*. Python documentation. https://docs.python.org/3/library/random.html
3. Northeastern University. (n.d.). NUworks. https://nuworks.northeastern.edu
4. LinkedIn Corporation. (n.d.). *Jobs on LinkedIn*. https://www.linkedin.com/jobs
5. Introducing How You Match on LinkedIn Jobs. (2018). Linkedin.com. https://www.linkedin.com/blog/member/career/introducing-how-you-match-on-linkedin-jobs