Trent University

Project Report: Database Design and Implementation for Job Tracking System

Anh Tuan Hoang

I. Introduction

Purpose

The primary goal of this project is to design and implement a relational database system to manage and track information related to students, alumni, and job opportunities. The database enables effective communication between departments, students, alumni, and recruiters while facilitating the recommendation of relevant job postings. This system aims to streamline the process of matching job opportunities with suitable candidates and maintain up-to-date information about all stakeholders.

Scope

The project addresses the challenge of bridging the gap between employers and potential candidates by automating job recommendations and maintaining comprehensive records of students, alumni, departments, and recruiters. The database is designed to:

- Track current students and their associated academic information.
- Manage alumni data, including career progress and interests.
- Allow recruiters to post jobs and associate them with relevant departments.
- Provide departments with tools to recommend jobs to their associated students and alumni.

The intended users of this database are:

- Students: To find job opportunities relevant to their field of study and preferences.
- Alumni: To receive job recommendations and maintain connections with the university.
- Recruiters: To post and manage job listings efficiently.

- Departments and Administrators: To manage records, generate insights, and facilitate job recommendations.

Overview

The database is built around key entities such as Students, Alumni, Departments, Jobs, Recruiters, and Companies. Relationships are implemented to ensure efficient data retrieval and consistency. Key features include:

- Comprehensive job recommendation functionality.
- Role-based access control (RBAC) for data security.
- Automation through triggers and stored procedures.
- High-performance queries optimized with indexes and views.
- Data logging and encryption for enhanced security.

II. Requirements Analysis

Functional Requirements

The database must fulfill the following functional requirements:

- Students can view job opportunities relevant to their department and preferences.
- Alumni can receive job recommendations based on their interests, years of experience, and department.
- Recruiters can post, update, and delete job postings.
- Departments can track students and alumni, recommending relevant job opportunities.

- The system must log sensitive actions such as updates and deletions of data.
- Users should be able to perform CRUD operations on all tables as per their roles.
- Administrators should have access to all data and logs for audit purposes.

Non-Functional Requirements

- **Performance:** The database should handle a large number of users and queries efficiently. Indexes must be used for high-performance queries.
- **Scalability:** The system should support future growth, including more entities, relationships, and higher data volumes.
- **Security:** Role-based access control must restrict users to permitted operations. Sensitive data must be encrypted, and audit logs maintained for accountability.
- **Reliability:** The database should ensure data integrity through constraints and relationships. Redundancy should be minimized using normalization (BCNF).
- **Usability:** Simplified access through views and functions to retrieve relevant data quickly.

Assumptions

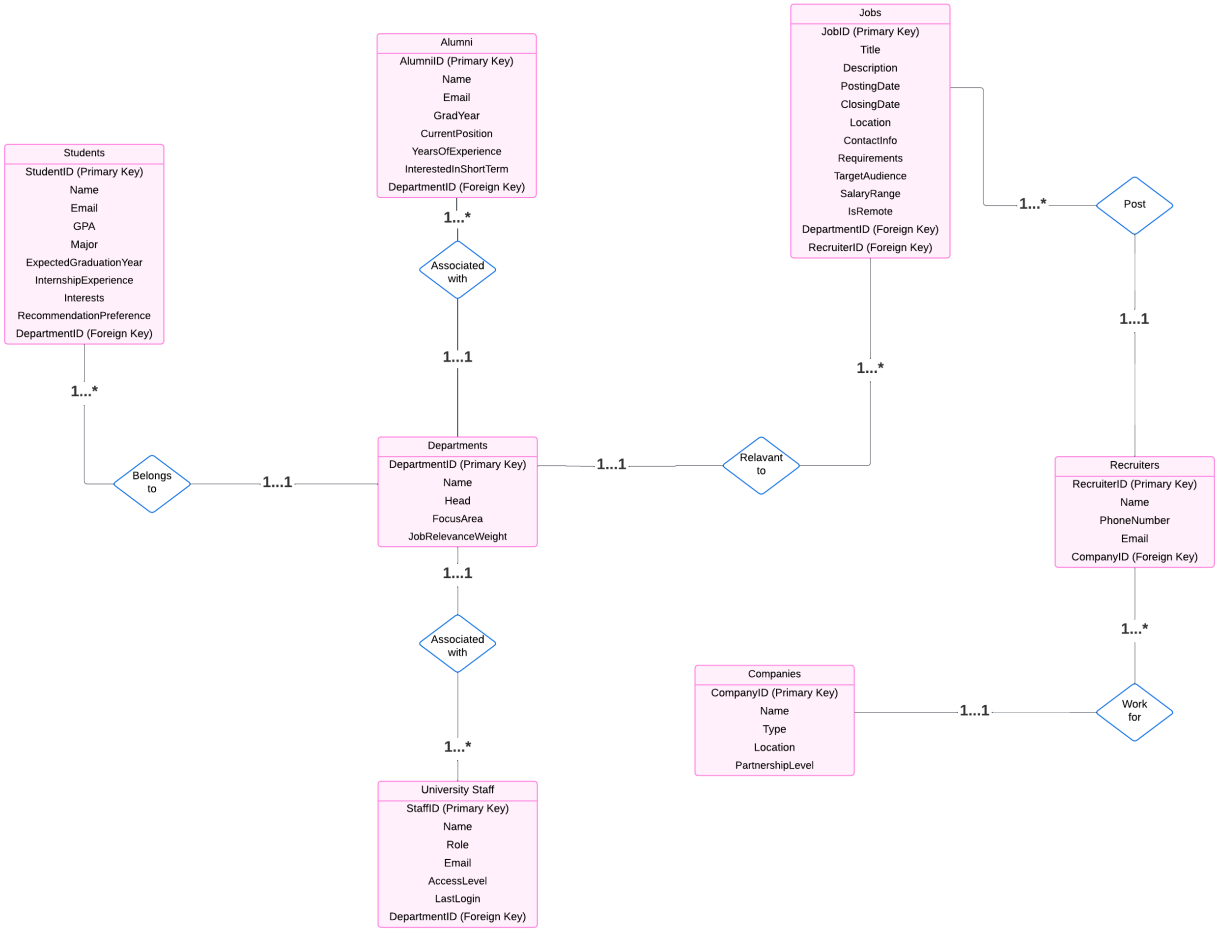
- Alumni may also be students, particularly for graduate or continuing education programs.
- Recruiters are associated with companies and can change affiliations over time.
- Job postings can be linked to one or more departments.

- Students and alumni may not always belong to a department, but they can still access general job opportunities.
- The database operates in a controlled environment with defined user roles (e.g., student, recruiter, administrator).

III. Database Design

Entity-Relationship (ER) Diagram

Based on the project requirements, I have created an ER diagram with 7 key entities in the database, ensuring all necessary relationships and attributes are effectively captured. capturing all necessary relationships and attributes to manage students, alumni, jobs, departments, and recruiters effectively. The ER diagram illustrates these components visually, serving as a foundational blueprint for the database. By mapping out the cardinality and associations, the ER diagram helps to identify redundant data and lays the groundwork for normalization.



Description of Components

1, Students: Represents students currently enrolled at the university.

2, Alumni: Captures information about graduates and their career progress.

3, Departments: Manages academic and professional focus areas.

4, Jobs: Represents job postings with detailed descriptions and associated metadata.

5, Recruiters: Represents individuals posting jobs on behalf of companies.

6, Companies: Stores details about organizations offering job opportunities.

7, University Staff: Represents university administrative staff who manage the system. Roles can include system administrators, department heads, and moderators.

Relationships and Cardinality:

1, Students → Departments (Many-to-One):

- **Explanation:** Each student belongs to one department, but a department can have many students.
- **Implementation:** This relationship is implemented using a Foreign Key DepartmentID in the Students table referencing Departments.DepartmentID.

2, Alumni → Departments (Many-to-One):

- **Explanation:** Each alumnus is associated with one department, but a department can have many alumni.
- **Implementation:** This relationship is implemented using a Foreign Key DepartmentID in the Alumni table referencing Departments.DepartmentID.

3, Jobs → Departments (Many-to-Many):

- **Explanation:** A job can be relevant to multiple departments, and a department can have many relevant jobs.
- **Implementation:** This is implemented through a junction table JobDepartment with composite keys JobID and DepartmentID referencing Jobs.JobID and Departments.DepartmentID.

4, Jobs → Recruiters (Many-to-One):

- **Explanation:** Each job is posted by one recruiter, but a recruiter can post many jobs.
- **Implementation:** This relationship is implemented using a Foreign Key RecruiterID in the Jobs table referencing Recruiters.RecruiterID.

5, Recruiters → Companies (Many-to-One):

- **Explanation:** A recruiter works for one company, but a company can have multiple recruiters.
- **Implementation:** This relationship is implemented using a Foreign Key CompanyID in the Recruiters table referencing Companies.CompanyID.

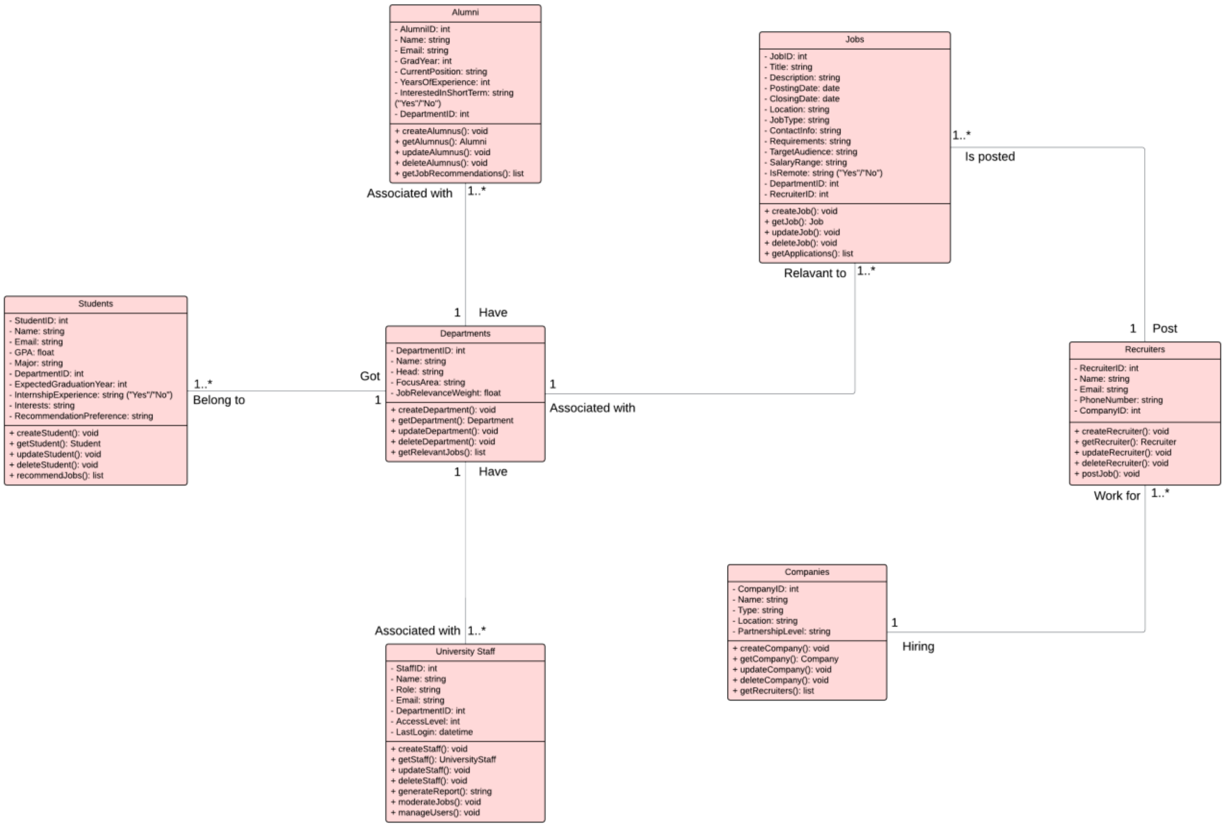
6, University Staff → Departments (Many-to-One):

- **Explanation:** Each university staff member is associated with one department, but a department can have multiple staff members.
- **Implementation:** This relationship is implemented using a Foreign Key DepartmentID in the UniversityStaff table referencing Departments.DepartmentID.

UML Diagram

From ERD, I have created a UML diagram. As UML diagram provides a detailed class-based representation of the database schema:

- UML diagrams are particularly useful for understanding complex database systems as they visually represent the structure of entities, their attributes, data types, and associated methods. This clarity helps developers and stakeholders identify relationships, dependencies, and potential design issues efficiently.
- It bridges the gap between database design and object-oriented programming by providing a clear structure for data interactions.
- Helps developers and stakeholders visualize how data flows and interacts in the system.
- UML diagrams are particularly useful for understanding complex database systems as they provide a high-level overview, making it easier to identify relationships, dependencies, and potential design flaws.



Normalization

Why the Database Design is in BCNF?

We can verify an example schema (Students Table):

Primary Key: StudentID

Attributes: Name, Email, GPA, Major, DepartmentID, ExpectedGraduationYear, InternshipExperience, Interests, RecommendationPreference

Determinants:

- **StudentID** is the **sole** determinant for all other attributes, ensuring no partial or transitive dependencies.
- No non-candidate key acts as a determinant for any attribute.
- The foreign keys DepartmentID is direct relationships and do not introduce transitive dependencies.

→ **Conclusion:** The Students table satisfies BCNF because all determinants (in this case, StudentID) are candidate keys, and there are no partial or transitive dependencies.

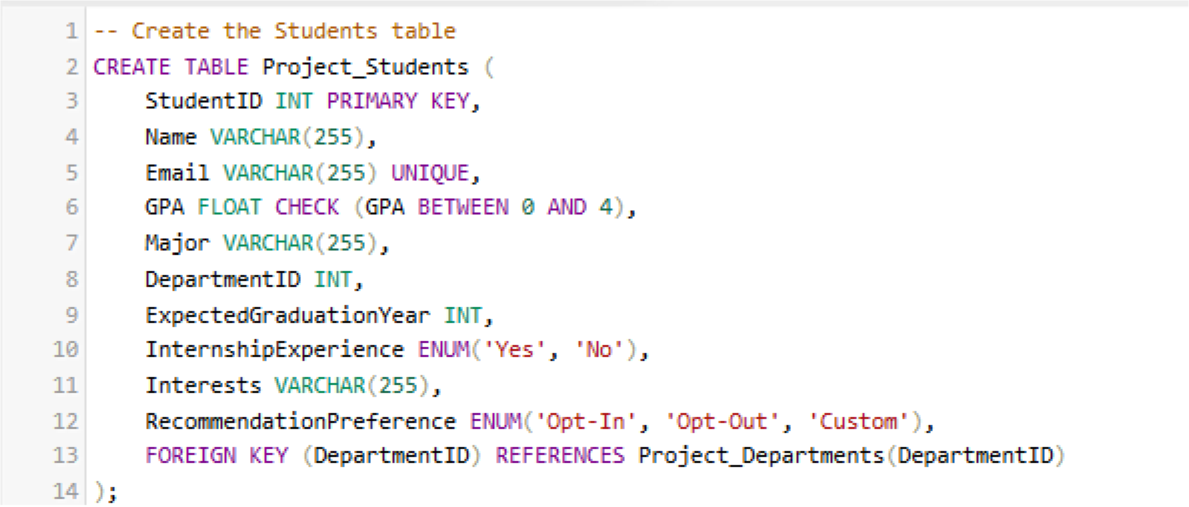
Other tables (Departments, Recruiters, Companies) follow the same principles.

→ Since BCNF is a stricter form of 3NF, a table that satisfies BCNF automatically satisfies 3NF.

IV. Implementation

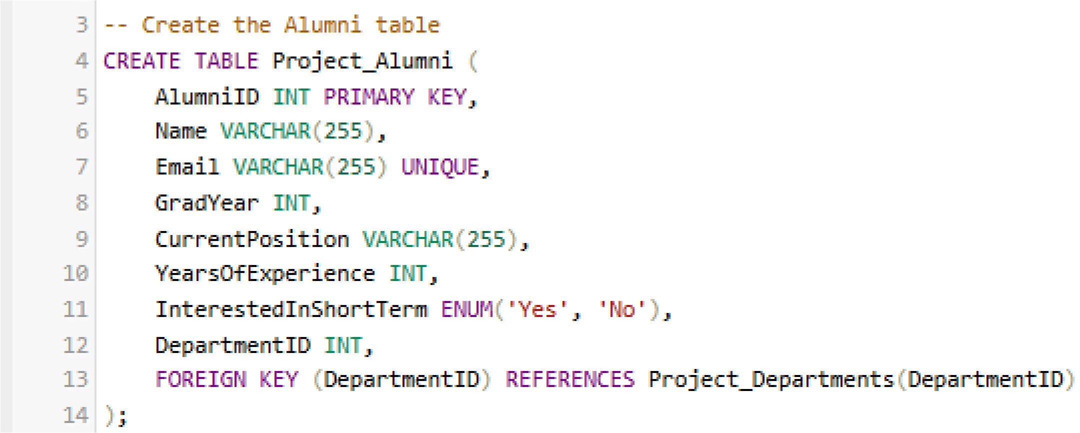
Schema Creation

Database schema was created based on our ERD. Each table includes primary keys, foreign keys, and constraints to ensure data integrity and normalization. Relationships were implemented based on project requirements. For project schema creation, I also add Project_ before name of all tables to distinguish it from others table from labs, assignments in our course.



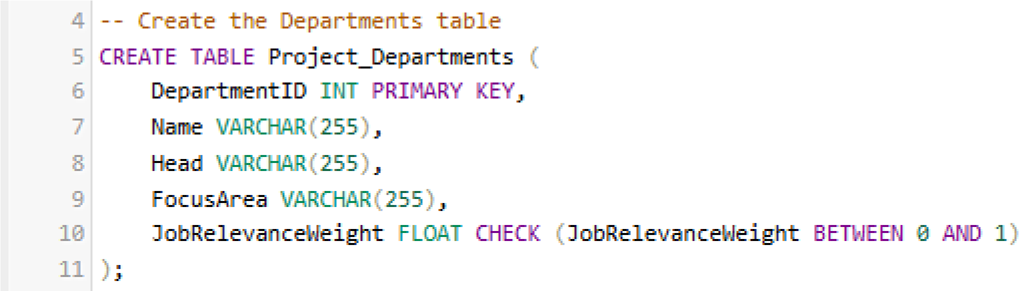
```
-- Create the Students table
CREATE TABLE Project_Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    GPA FLOAT CHECK (GPA BETWEEN 0 AND 4),
    Major VARCHAR(255),
    DepartmentID INT,
    ExpectedGraduationYear INT,
    InternshipExperience ENUM('Yes', 'No'),
    Interests VARCHAR(255),
    RecommendationPreference ENUM('Opt-In', 'Opt-Out', 'Custom'),
    FOREIGN KEY (DepartmentID) REFERENCES Project_Departments(DepartmentID)
);
```

Student schema



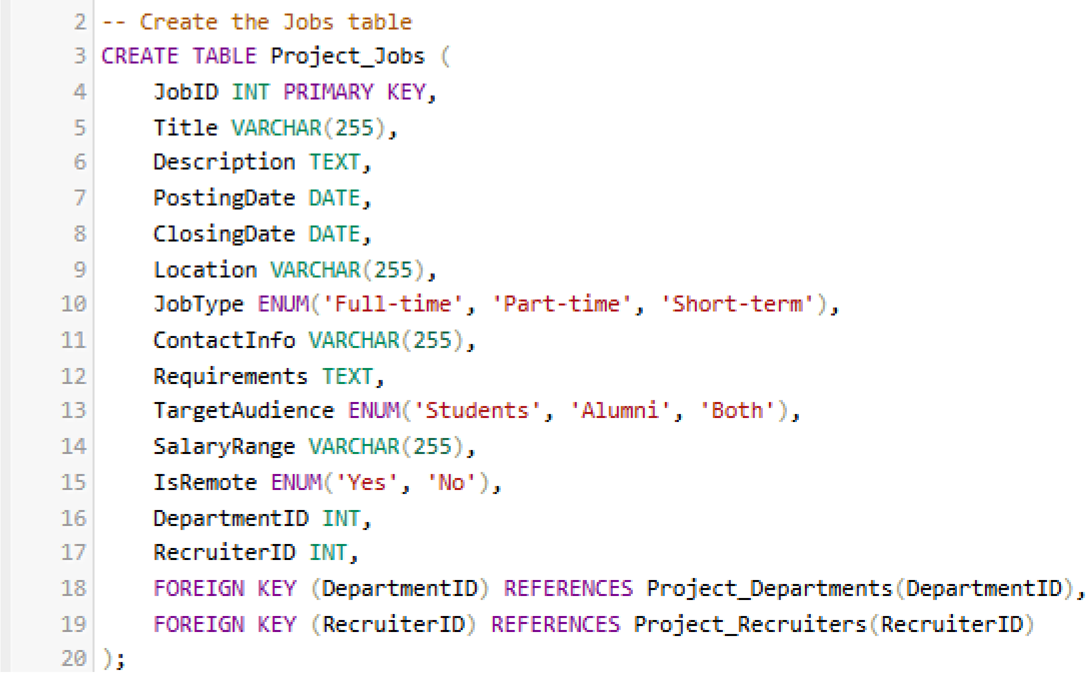
```sql
-- Create the Alumni table
CREATE TABLE Project_Alumni (
    AlumniID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    GradYear INT,
    CurrentPosition VARCHAR(255),
    YearsOfExperience INT,
    InterestedInShortTerm ENUM('Yes', 'No'),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Project_Departments(DepartmentID)
);
```

Alumni schema



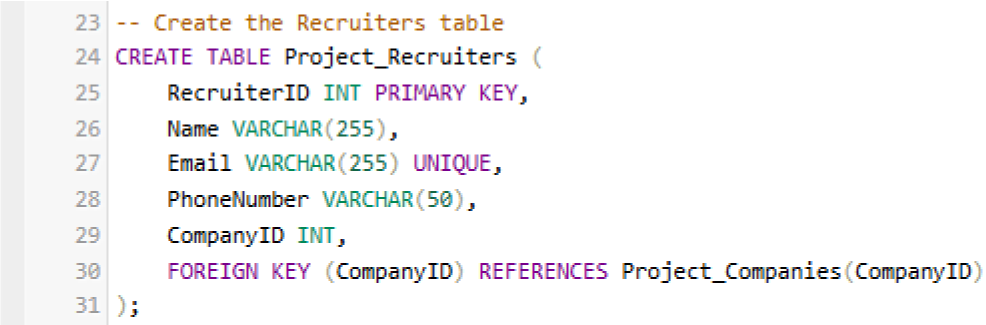
```sql
-- Create the Departments table
CREATE TABLE Project_Departments (
    DepartmentID INT PRIMARY KEY,
    Name VARCHAR(255),
    Head VARCHAR(255),
    FocusArea VARCHAR(255),
    JobRelevanceWeight FLOAT CHECK (JobRelevanceWeight BETWEEN 0 AND 1)
);
```

Departments schema



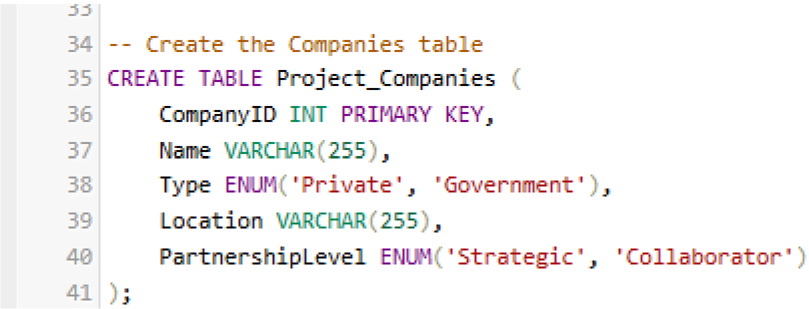
```sql
-- Create the Jobs table
CREATE TABLE Project_Jobs (
    JobID INT PRIMARY KEY,
    Title VARCHAR(255),
    Description TEXT,
    PostingDate DATE,
    ClosingDate DATE,
    Location VARCHAR(255),
    JobType ENUM('Full-time', 'Part-time', 'Short-term'),
    ContactInfo VARCHAR(255),
    Requirements TEXT,
    TargetAudience ENUM('Students', 'Alumni', 'Both'),
    SalaryRange VARCHAR(255),
    IsRemote ENUM('Yes', 'No'),
    DepartmentID INT,
    RecruiterID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Project_Departments(DepartmentID),
    FOREIGN KEY (RecruiterID) REFERENCES Project_Recruiters(RecruiterID)
);
```

Jobs schema



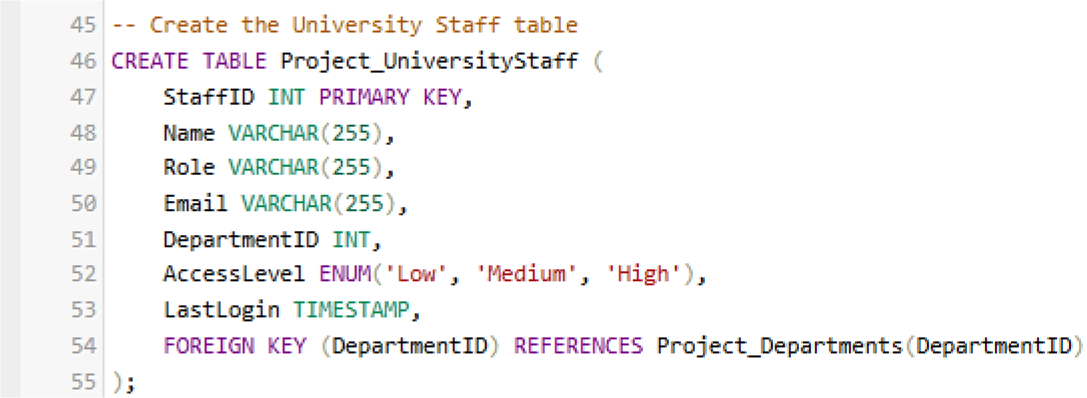
```sql
-- Create the Recruiters table
CREATE TABLE Project_Recruiters (
    RecruiterID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    PhoneNumber VARCHAR(50),
    CompanyID INT,
    FOREIGN KEY (CompanyID) REFERENCES Project_Companies(CompanyID)
);
```

Recruiter schema



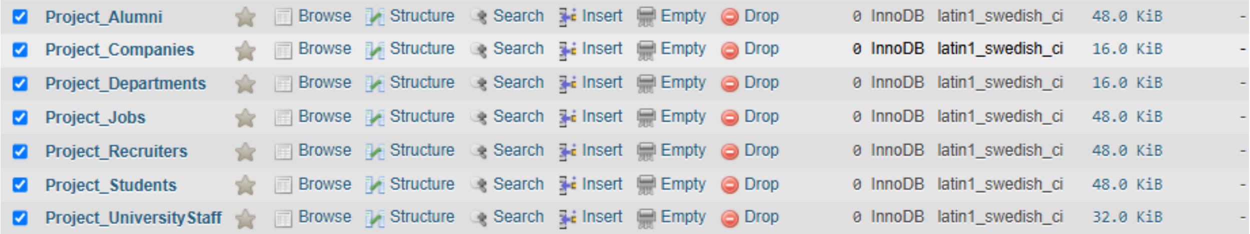
```sql
-- Create the Companies table
CREATE TABLE Project_Companies (
    CompanyID INT PRIMARY KEY,
    Name VARCHAR(255),
    Type ENUM('Private', 'Government'),
    Location VARCHAR(255),
    PartnershipLevel ENUM('Strategic', 'Collaborator')
);
```

Companies schema



```sql
-- Create the University Staff table
CREATE TABLE Project_UniversityStaff (
    StaffID INT PRIMARY KEY,
    Name VARCHAR(255),
    Role VARCHAR(255),
    Email VARCHAR(255),
    DepartmentID INT,
    AccessLevel ENUM('Low', 'Medium', 'High'),
    LastLogin TIMESTAMP,
    FOREIGN KEY (DepartmentID) REFERENCES Project_Departments(DepartmentID)
);
```

University Staff schema

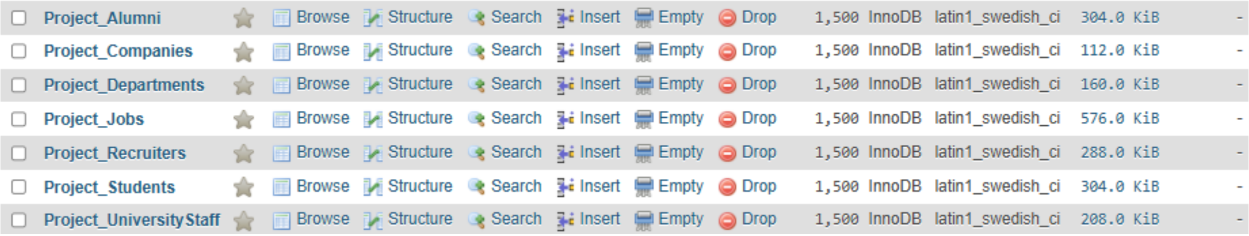


☑	Project_Alumni	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	48.0 KiB	-
☑	Project_Companies	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16.0 KiB	-
☑	Project_Departments	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16.0 KiB	-
☑	Project_Jobs	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	48.0 KiB	-
☑	Project_Recruiters	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	48.0 KiB	-
☑	Project_Students	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	48.0 KiB	-
☑	Project_UniversityStaff	☆	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	32.0 KiB	-

All initial tables

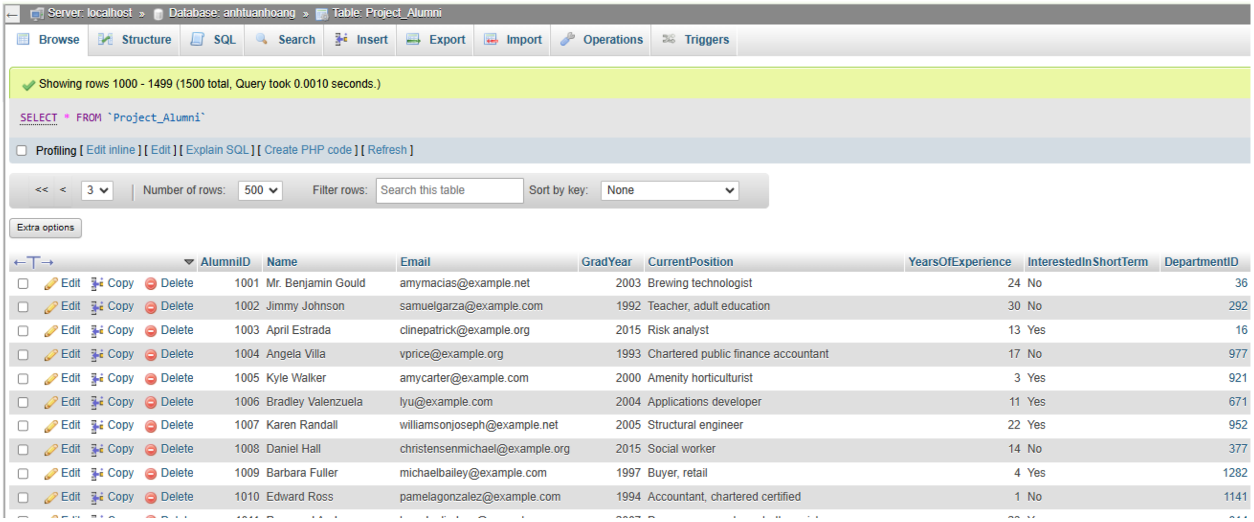
Test Data Population

Data for each table was generated using Python scripts with the Faker library and random data generation logic. A minimum of 1,500 rows were created for each, with realistic values that match schema constraints. For inserting fake data into database, I do it for table with no foreign keys first to avoid making errors. I will attach the fake data code source in pdf file.



□	Project_Alumni	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	304.0 KiB	-
□	Project_Companies	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	112.0 KiB	-
□	Project_Departments	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	160.0 KiB	-
□	Project_Jobs	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	576.0 KiB	-
□	Project_Recruiters	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	288.0 KiB	-
□	Project_Students	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	304.0 KiB	-
□	Project_UniversityStaff	☆	Browse	Structure	Search	Insert	Empty	Drop	1,500	InnoDB	latin1_swedish_ci	208.0 KiB	-

1500 rows data for each table


Server: localhost » Database: anhtuanhoang » Table: Project_Alumni
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers

✓ Showing rows 1000 - 1499 (1500 total, Query took 0.0010 seconds.)

```
SELECT * FROM `Project_Alumni`
```

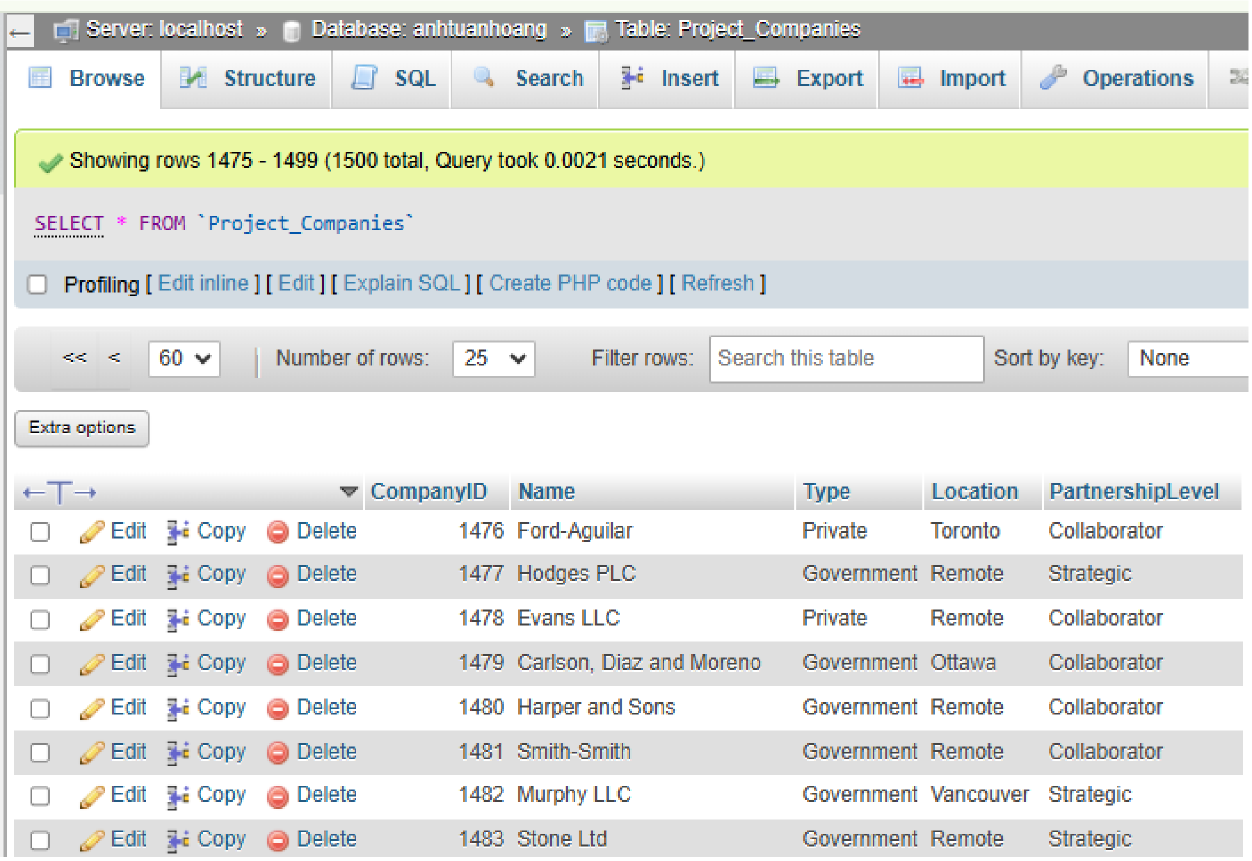
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

<< < 3 | Number of rows: 500 Filter rows: Search this table Sort by key: None

Extra options

←T→	AlumniID	Name	Email	GradYear	CurrentPosition	YearsOfExperience	InterestedInShortTerm	DepartmentID
□ Edit Copy Delete	1001	Mr. Benjamin Gould	amymacias@example.net	2003	Brewing technologist	24	No	36
□ Edit Copy Delete	1002	Jimmy Johnson	samuelgarza@example.com	1992	Teacher, adult education	30	No	292
□ Edit Copy Delete	1003	April Estrada	clinepatrick@example.org	2015	Risk analyst	13	Yes	16
□ Edit Copy Delete	1004	Angela Villa	vprice@example.org	1993	Chartered public finance accountant	17	No	977
□ Edit Copy Delete	1005	Kyle Walker	amycarter@example.com	2000	Amenity horticulturist	3	Yes	921
□ Edit Copy Delete	1006	Bradley Valenzuela	lyu@example.com	2004	Applications developer	11	Yes	671
□ Edit Copy Delete	1007	Karen Randall	williamsonjoseph@example.net	2005	Structural engineer	22	Yes	952
□ Edit Copy Delete	1008	Daniel Hall	christensenmichael@example.org	2015	Social worker	14	No	377
□ Edit Copy Delete	1009	Barbara Fuller	michaelbailey@example.com	1997	Buyer, retail	4	Yes	1282
□ Edit Copy Delete	1010	Edward Ross	pamelagonzalez@example.com	1994	Accountant, chartered certified	1	No	1141

Alumni data


Server: localhost » Database: anhtuanhoang » Table: Project_Companies
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

✓ Showing rows 1475 - 1499 (1500 total, Query took 0.0021 seconds.)

```
SELECT * FROM `Project_Companies`
```

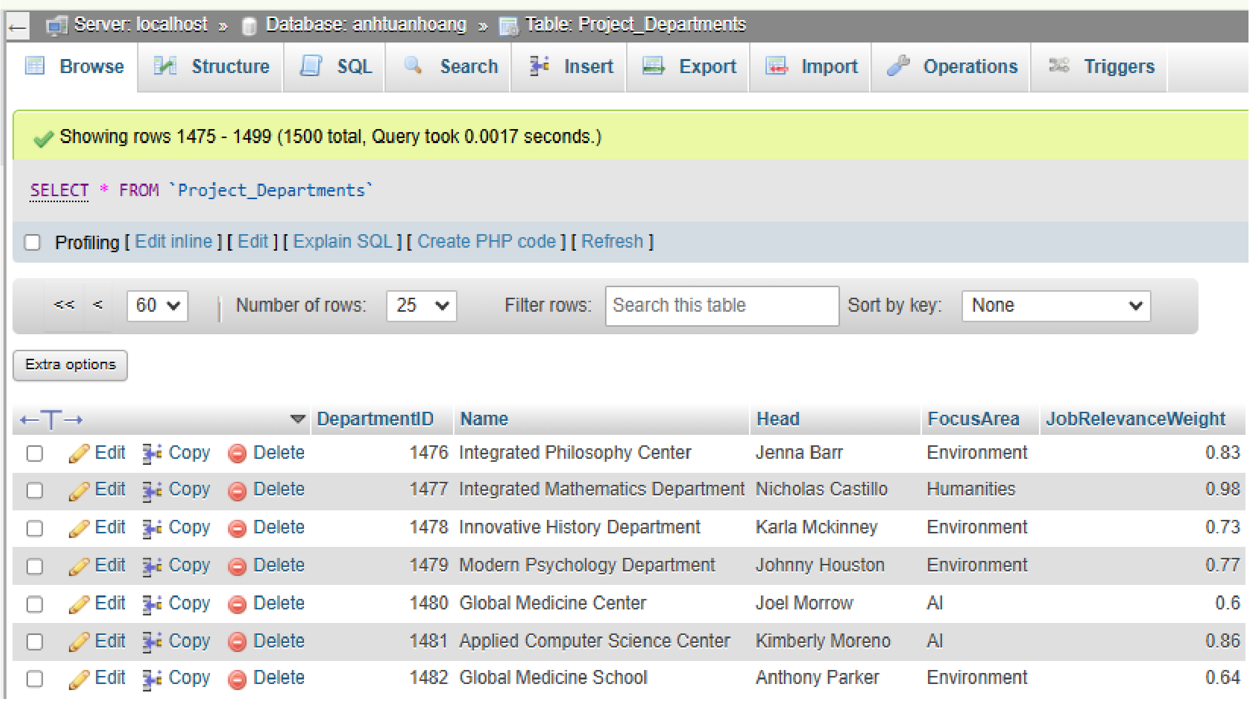
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

<< < 60 | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

←T→	CompanyID	Name	Type	Location	PartnershipLevel
□ Edit Copy Delete	1476	Ford-Aguilar	Private	Toronto	Collaborator
□ Edit Copy Delete	1477	Hodges PLC	Government	Remote	Strategic
□ Edit Copy Delete	1478	Evans LLC	Private	Remote	Collaborator
□ Edit Copy Delete	1479	Carlson, Diaz and Moreno	Government	Ottawa	Collaborator
□ Edit Copy Delete	1480	Harper and Sons	Government	Remote	Collaborator
□ Edit Copy Delete	1481	Smith-Smith	Government	Remote	Collaborator
□ Edit Copy Delete	1482	Murphy LLC	Government	Vancouver	Strategic
□ Edit Copy Delete	1483	Stone Ltd	Government	Remote	Strategic

Companies data



← Server: localhost » Database: anhtuanhoang » Table: Project_Departments

✓ Showing rows 1475 - 1499 (1500 total, Query took 0.0017 seconds.)

```
SELECT * FROM `Project_Departments`
```

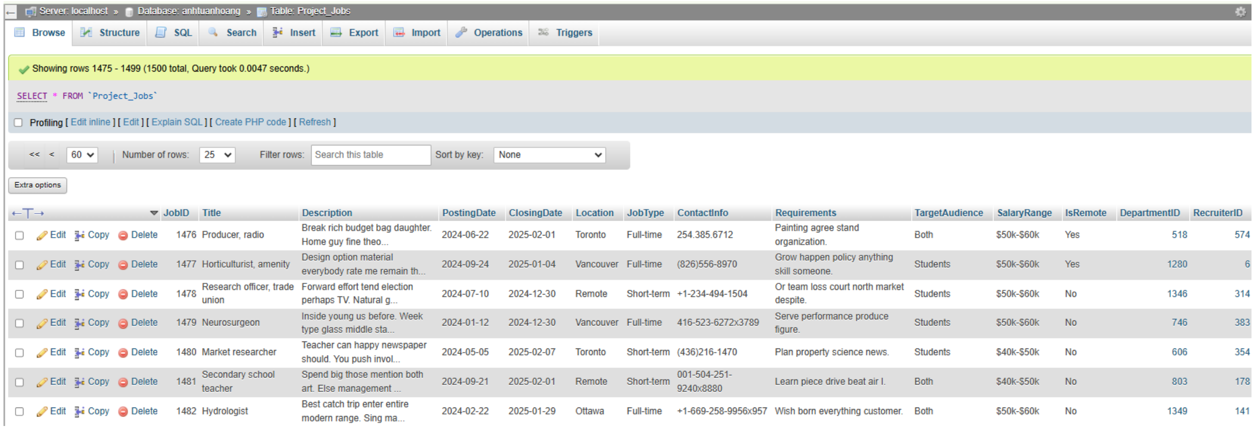
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

<< < 60 | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

←T→	DepartmentID	Name	Head	FocusArea	JobRelevanceWeight
□ Edit Copy Delete	1476	Integrated Philosophy Center	Jenna Barr	Environment	0.83
□ Edit Copy Delete	1477	Integrated Mathematics Department	Nicholas Castillo	Humanities	0.98
□ Edit Copy Delete	1478	Innovative History Department	Karla Mckinney	Environment	0.73
□ Edit Copy Delete	1479	Modern Psychology Department	Johnny Houston	Environment	0.77
□ Edit Copy Delete	1480	Global Medicine Center	Joel Morrow	AI	0.6
□ Edit Copy Delete	1481	Applied Computer Science Center	Kimberly Moreno	AI	0.86
□ Edit Copy Delete	1482	Global Medicine School	Anthony Parker	Environment	0.64

Department data



← Server: localhost » Database: anhtuanhoang » Table: Project_Jobs

✓ Showing rows 1475 - 1499 (1500 total, Query took 0.0047 seconds.)

```
SELECT * FROM `Project_Jobs`
```

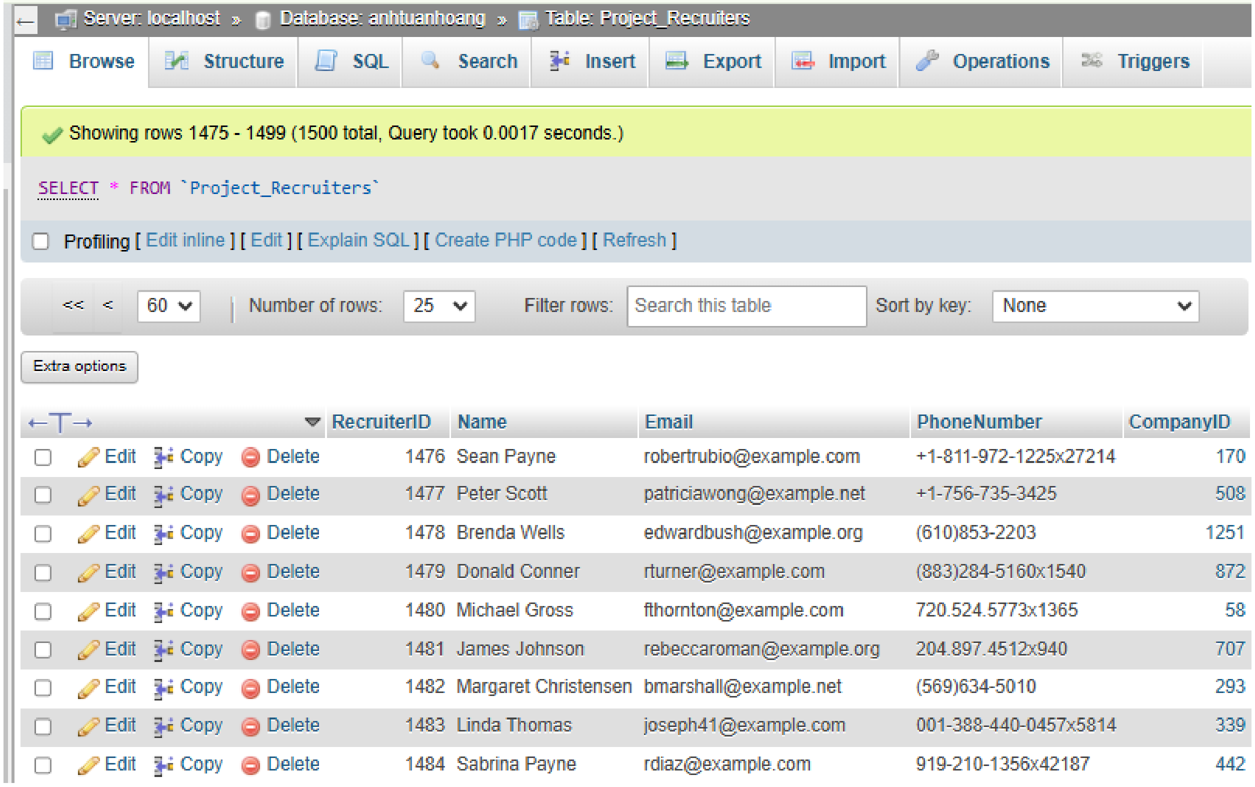
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

<< < 60 | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

←T→	JobID	Title	Description	PostingDate	ClosingDate	Location	JobType	ContactInfo	Requirements	TargetAudience	SalaryRange	IsRemote	DepartmentID	RecruiterID
□ Edit Copy Delete	1476	Producer, radio	Break rich budget bag daughter. Home guy fine theo...	2024-06-22	2025-02-01	Toronto	Full-time	254.385.6712	Painting agree stand organization.	Both	$50k-$60k	Yes	518	574
□ Edit Copy Delete	1477	Horticulturist, amenity	Design option material everybody rate me remain th...	2024-09-24	2025-01-04	Vancouver	Full-time	(826)556-8970	Grow happen policy anything skill someone.	Students	$50k-$60k	Yes	1280	6
□ Edit Copy Delete	1478	Research officer, trade union	Forward effort tend election perhaps TV. Natural g...	2024-07-10	2024-12-30	Remote	Short-term	+1-234-494-1504	Or team loss court north market despite.	Students	$50k-$60k	No	1346	314
□ Edit Copy Delete	1479	Neurosurgeon	Inside young us before. Week type glass middle sta...	2024-01-12	2024-12-30	Vancouver	Full-time	416-523-6272x3789	Serve performance produce figure.	Students	$50k-$60k	No	746	383
□ Edit Copy Delete	1480	Market researcher	Teacher can happy newspaper should. You push invol...	2024-05-05	2025-02-07	Toronto	Short-term	(436)216-1470	Plan property science news.	Students	$40k-$50k	No	606	354
□ Edit Copy Delete	1481	Secondary school teacher	Spend big those mention both art. Else management ...	2024-09-21	2025-02-01	Remote	Short-term	001-504-251-9240x8880	Learn piece drive beat air I.	Both	$40k-$50k	No	803	178
□ Edit Copy Delete	1482	Hydrologist	Best catch trip enter entire modern range. Sing ma...	2024-02-22	2025-01-29	Ottawa	Full-time	+1-669-258-9956x957	Wish born everything customer.	Both	$50k-$60k	No	1349	141

Jobs data


Server: localhost » Database: anhtuanhoang » Table: Project_Recruiters
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
<< < 60
Extra options

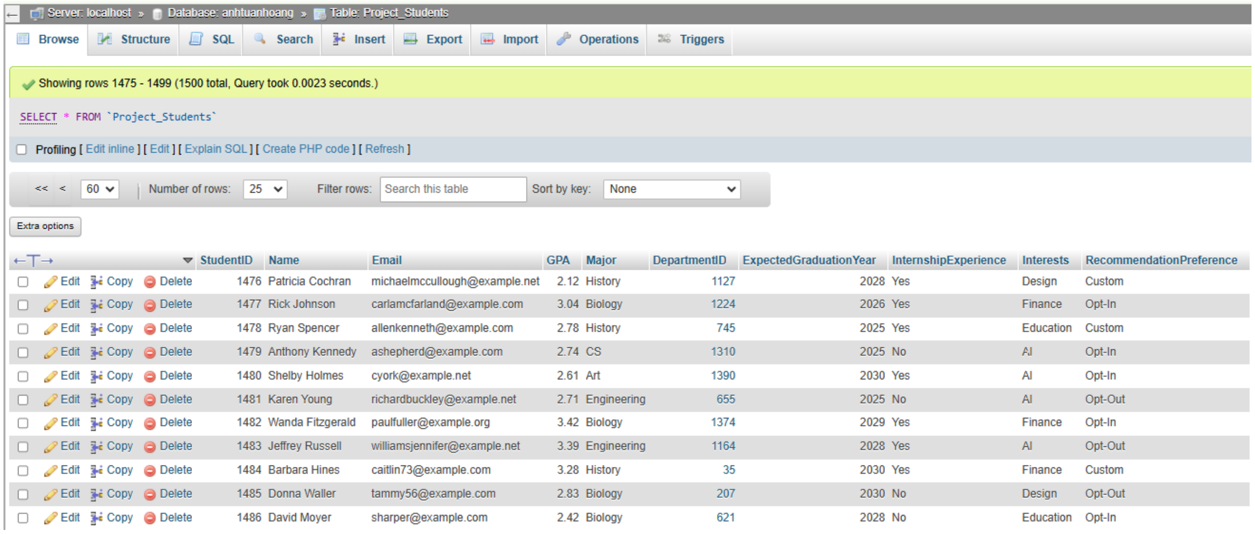
✓ Showing rows 1475 - 1499 (1500 total, Query took 0.0017 seconds.)

```
SELECT * FROM `Project_Recruiters`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

	RecruiterID	Name	Email	PhoneNumber	CompanyID
□ Edit Copy Delete	1476	Sean Payne	robertrubio@example.com	+1-811-972-1225x27214	170
□ Edit Copy Delete	1477	Peter Scott	patriciawong@example.net	+1-756-735-3425	508
□ Edit Copy Delete	1478	Brenda Wells	edwardbush@example.org	(610)853-2203	1251
□ Edit Copy Delete	1479	Donald Conner	rturner@example.com	(883)284-5160x1540	872
□ Edit Copy Delete	1480	Michael Gross	fthornton@example.com	720.524.5773x1365	58
□ Edit Copy Delete	1481	James Johnson	rebeccaroman@example.org	204.897.4512x940	707
□ Edit Copy Delete	1482	Margaret Christensen	bmarshall@example.net	(569)634-5010	293
□ Edit Copy Delete	1483	Linda Thomas	joseph41@example.com	001-388-440-0457x5814	339
□ Edit Copy Delete	1484	Sabrina Payne	rdiaz@example.com	919-210-1356x42187	442

Recruiters data


Server: localhost » Database: anhtuanhoang » Table: Project_Students
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
<< < 60
Extra options

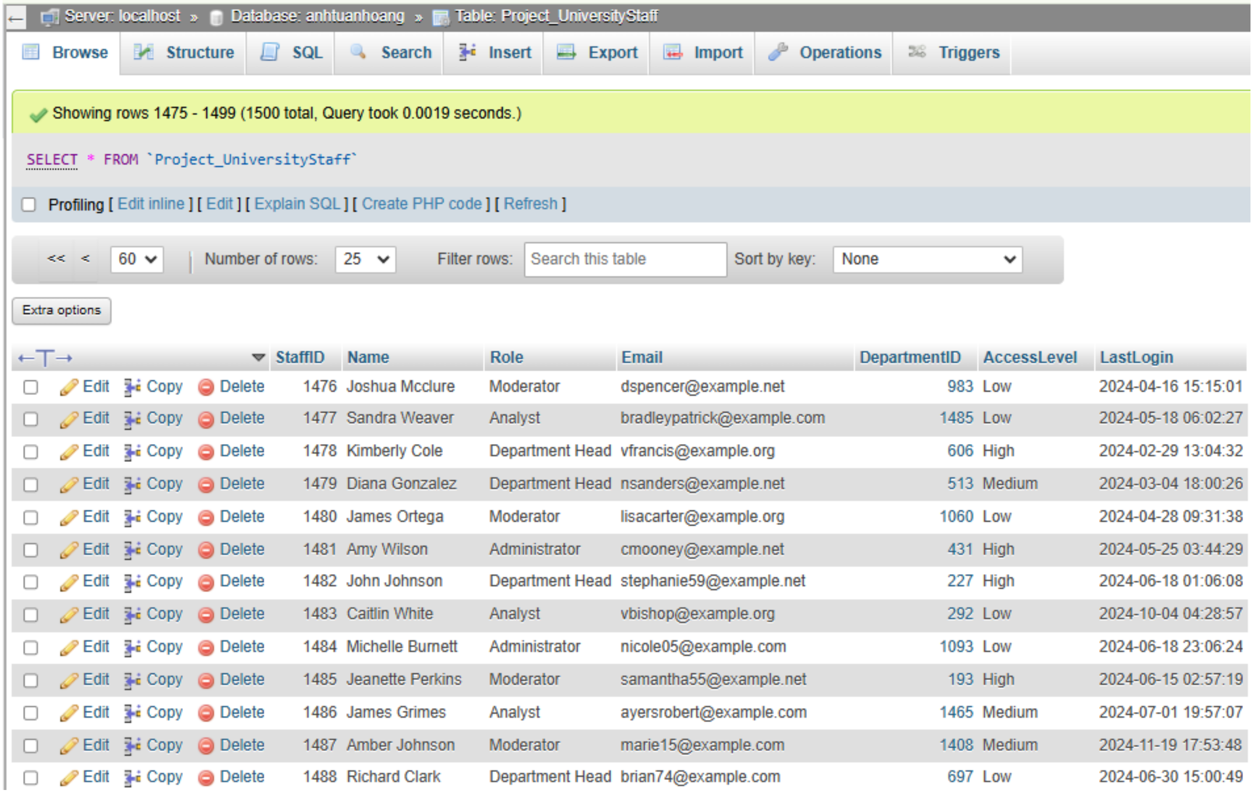
✓ Showing rows 1475 - 1499 (1500 total, Query took 0.0023 seconds.)

```
SELECT * FROM `Project_Students`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

	StudentID	Name	Email	GPA	Major	DepartmentID	ExpectedGraduationYear	InternshipExperience	Interests	RecommendationPreference
□ Edit Copy Delete	1476	Patricia Cochran	michaelmccullough@example.net	2.12	History	1127	2028	Yes	Design	Custom
□ Edit Copy Delete	1477	Rick Johnson	carlamcfarland@example.com	3.04	Biology	1224	2026	Yes	Finance	Opt-In
□ Edit Copy Delete	1478	Ryan Spencer	allenkenneth@example.com	2.78	History	745	2025	Yes	Education	Custom
□ Edit Copy Delete	1479	Anthony Kennedy	ashepherd@example.com	2.74	CS	1310	2025	No	AI	Opt-In
□ Edit Copy Delete	1480	Shelby Holmes	cyork@example.net	2.61	Art	1390	2030	Yes	AI	Opt-In
□ Edit Copy Delete	1481	Karen Young	richardbuckley@example.net	2.71	Engineering	655	2025	No	AI	Opt-Out
□ Edit Copy Delete	1482	Wanda Fitzgerald	paulfuller@example.org	3.42	Biology	1374	2029	Yes	Finance	Opt-In
□ Edit Copy Delete	1483	Jeffrey Russell	williamsjennifer@example.net	3.39	Engineering	1164	2028	Yes	AI	Opt-Out
□ Edit Copy Delete	1484	Barbara Hines	caitlin73@example.com	3.28	History	35	2030	Yes	Finance	Custom
□ Edit Copy Delete	1485	Donna Waller	tammy56@example.com	2.83	Biology	207	2030	No	Design	Opt-Out
□ Edit Copy Delete	1486	David Moyer	sharper@example.com	2.42	Biology	621	2028	No	Education	Opt-In

Students data


Server: localhost » Database: anhtuanhoang » Table: Project_UniversityStaff
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
<< < 60
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Extra options
Edit
Copy
Delete

Showing rows 1475 - 1499 (1500 total, Query took 0.0019 seconds.)

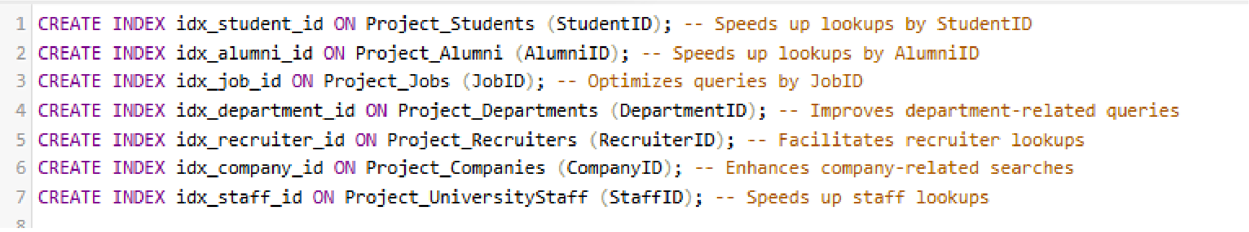
```
SELECT * FROM `Project_UniversityStaff`
```

StaffID	Name	Role	Email	DepartmentID	AccessLevel	LastLogin
1476	Joshua Mcclure	Moderator	dspencer@example.net	983	Low	2024-04-16 15:15:01
1477	Sandra Weaver	Analyst	bradleypatrick@example.com	1485	Low	2024-05-18 06:02:27
1478	Kimberly Cole	Department Head	vfrancis@example.org	606	High	2024-02-29 13:04:32
1479	Diana Gonzalez	Department Head	nsanders@example.net	513	Medium	2024-03-04 18:00:26
1480	James Ortega	Moderator	lisacarter@example.org	1060	Low	2024-04-28 09:31:38
1481	Amy Wilson	Administrator	cmooney@example.net	431	High	2024-05-25 03:44:29
1482	John Johnson	Department Head	stephanie59@example.net	227	High	2024-06-18 01:06:08
1483	Caitlin White	Analyst	vbishop@example.org	292	Low	2024-10-04 04:28:57
1484	Michelle Burnett	Administrator	nicole05@example.com	1093	Low	2024-06-18 23:06:24
1485	Jeanette Perkins	Moderator	samantha55@example.net	193	High	2024-06-15 02:57:19
1486	James Grimes	Analyst	ayersrobert@example.com	1465	Medium	2024-07-01 19:57:07
1487	Amber Johnson	Moderator	marie15@example.com	1408	Medium	2024-11-19 17:53:48
1488	Richard Clark	Department Head	brian74@example.com	697	Low	2024-06-30 15:00:49

University data

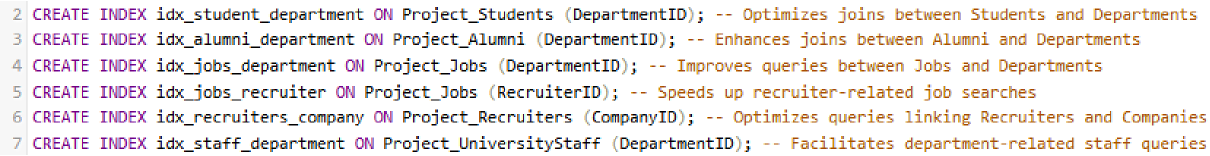
Indexes

Indexes are crucial for improving the performance of a database, especially when dealing with large datasets. With each table containing at least 1,500 rows, indexes help optimize query execution by reducing the time taken to search, filter, and sort records.



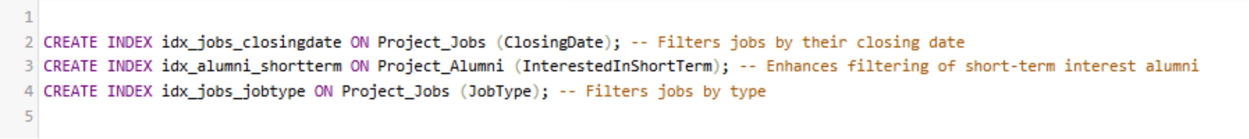
```sql
CREATE INDEX idx_student_id ON Project_Students (StudentID); -- Speeds up lookups by StudentID
CREATE INDEX idx_alumni_id ON Project_Alumni (AlumniID); -- Speeds up lookups by AlumniID
CREATE INDEX idx_job_id ON Project_Jobs (JobID); -- Optimizes queries by JobID
CREATE INDEX idx_department_id ON Project_Departments (DepartmentID); -- Improves department-related queries
CREATE INDEX idx_recruiter_id ON Project_Recruiters (RecruiterID); -- Facilitates recruiter lookups
CREATE INDEX idx_company_id ON Project_Companies (CompanyID); -- Enhances company-related searches
CREATE INDEX idx_staff_id ON Project_UniversityStaff (StaffID); -- Speeds up staff lookups
```

Index on primary keys



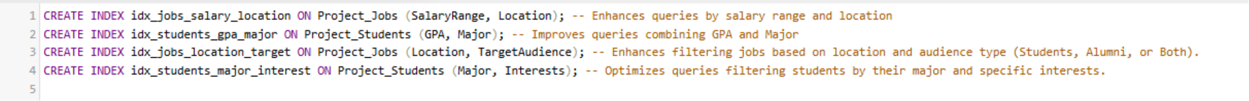
```sql
CREATE INDEX idx_student_department ON Project_Students (DepartmentID); -- Optimizes joins between Students and Departments
CREATE INDEX idx_alumni_department ON Project_Alumni (DepartmentID); -- Enhances joins between Alumni and Departments
CREATE INDEX idx_jobs_department ON Project_Jobs (DepartmentID); -- Improves queries between Jobs and Departments
CREATE INDEX idx_jobs_recruiter ON Project_Jobs (RecruiterID); -- Speeds up recruiter-related job searches
CREATE INDEX idx_recruiters_company ON Project_Recruiters (CompanyID); -- Optimizes queries linking Recruiters and Companies
CREATE INDEX idx_staff_department ON Project_UniversityStaff (DepartmentID); -- Facilitates department-related staff queries
```

Index on foreign keys



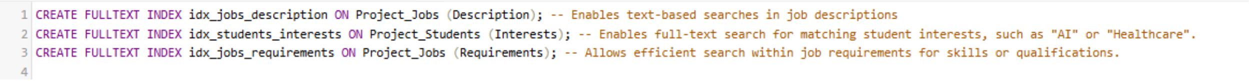
```sql

CREATE INDEX idx_jobs_closingdate ON Project_Jobs (ClosingDate); -- Filters jobs by their closing date
CREATE INDEX idx_alumni_shortterm ON Project_Alumni (InterestedInShortTerm); -- Enhances filtering of short-term interest alumni
CREATE INDEX idx_jobs_jobtype ON Project_Jobs (JobType); -- Filters jobs by type

```

Index on frequently queried fields



```sql
CREATE INDEX idx_jobs_salary_location ON Project_Jobs (SalaryRange, Location); -- Enhances queries by salary range and location
CREATE INDEX idx_students_gpa_major ON Project_Students (GPA, Major); -- Improves queries combining GPA and Major
CREATE INDEX idx_jobs_location_target ON Project_Jobs (Location, TargetAudience); -- Enhances filtering jobs based on location and audience type (Students, Alumni, or Both).
CREATE INDEX idx_students_major_interest ON Project_Students (Major, Interests); -- Optimizes queries filtering students by their major and specific interests.

```

Compound indexes



```sql
CREATE FULLTEXT INDEX idx_jobs_description ON Project_Jobs (Description); -- Enables text-based searches in job descriptions
CREATE FULLTEXT INDEX idx_students_interests ON Project_Students (Interests); -- Enables full-text search for matching student interests, such as "AI" or "Healthcare".
CREATE FULLTEXT INDEX idx_jobs_requirements ON Project_Jobs (Requirements); -- Allows efficient search within job requirements for skills or qualifications.

```

Full-Text Indexes

Backup Strategy

To ensure the reliability and integrity of the database, a robust backup strategy is essential. The following strategy has been designed to handle data recovery in case of system failures, data corruption, or accidental deletions:

1. **Full Backups:**

    - **Frequency:** A full backup of the database will be taken daily during off-peak hours to minimize performance impact.

- **Storage Location:** Full backups will be stored on a separate server dedicated to backups, with an additional copy saved in a secure cloud storage solution for redundancy.
- **Retention Policy:** Daily backups will be retained for 30 days, after which only weekly backups will be kept for a year.

2. **Incremental Backups:**
   - **Frequency:** Incremental backups will be performed hourly to capture changes made since the last full or incremental backup.
   - **Use Case:** These backups allow for precise recovery of data up to the most recent changes, minimizing data loss.
3. **Disaster Recovery:**
   - **Plan:** In the event of data loss, the recovery process involves restoring the latest full backup followed by applying all incremental backups in sequence.
   - **Testing:** Backup and recovery procedures will be tested monthly to ensure reliability.
4. **Special Considerations:**
   - **Batch Data Imports:** If large datasets are imported, a manual backup will be triggered beforehand to prevent loss during the process.
   - **Real-Time Updates:** For critical data, a write-ahead log (WAL) mechanism will be used to record changes before committing them to the database.

Replication and Hosting Strategy

To support the scalability and reliability of the database system, a replication and hosting strategy has been implemented. This strategy addresses the needs of up to 40,000 active alumni at Trent University and plans for potential scalability to handle a million users in the future.

1. **Replication:**
   - **Primary-Replica Model:**
     - A primary database server handles all write operations.
     - Two replica servers asynchronously replicate the primary server to handle read-heavy operations such as job searches and alumni queries.
   - **Benefits:**
     - Improves performance by distributing read queries across replicas.
     - Provides redundancy, ensuring high availability in case of a primary server failure.
   - **Geo-Replication:**
     - A secondary replica server is hosted in a geographically distant data center to ensure data availability during regional outages.
2. **Hosting Strategy:**
   - **Server Specifications:**

    - A high-performance database server is used for the primary instance with sufficient CPU, RAM, and SSD storage for rapid data processing.

    - Replica servers are optimized for read operations and use caching to improve response times.

- **Cloud Hosting:**

    - The system is hosted on a cloud platform (e.g., AWS RDS or Azure SQL) to leverage auto-scaling capabilities and distributed backups.

3. **Scalability Considerations:**

    - **Load Balancer:** A load balancer is used to direct read queries to replicas and write queries to the primary server.

    - **Connection Pooling:** Optimized database connections ensure that a large number of concurrent users can access the system without degrading performance.

    - **Shard-Based Partitioning (Future Consideration):**

        - If user numbers increase beyond current estimates, the database will be partitioned into shards based on departments or geographic locations.

4. **Monitoring and Maintenance:**

    - **Monitoring Tools:** Tools like Prometheus and Grafana will monitor server performance, replication lag, and query execution times.

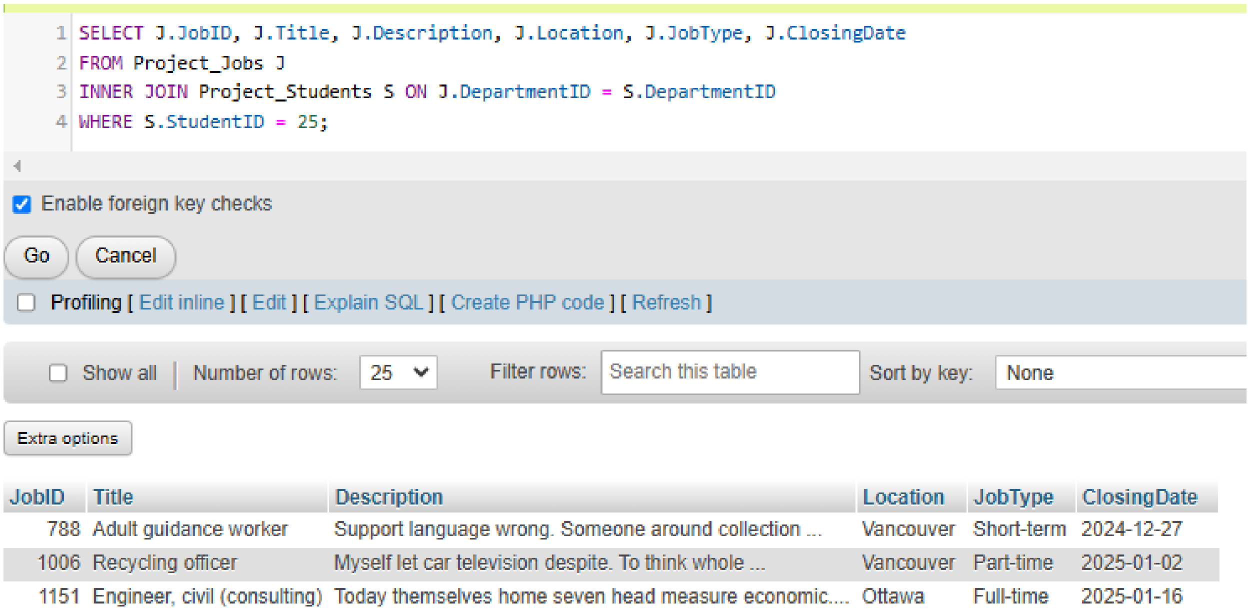
    - **Scheduled Maintenance:** Regular updates and optimizations will be performed during off-peak hours to minimize user disruption.

V. Queries, Views, and Security

#For students

Find Jobs Relevant to Students: Retrieves job postings relevant to a student's department.

Enables students to discover jobs aligned with their field of study.



```
SELECT J.JobID, J.Title, J.Description, J.Location, J.JobType, J.ClosingDate
FROM Project_Jobs J
INNER JOIN Project_Students S ON J.DepartmentID = S.DepartmentID
WHERE S.StudentID = 25;
```

☑ Enable foreign key checks

Go
Cancel
☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
☐ Show all
Filter rows: Search this table
Extra options

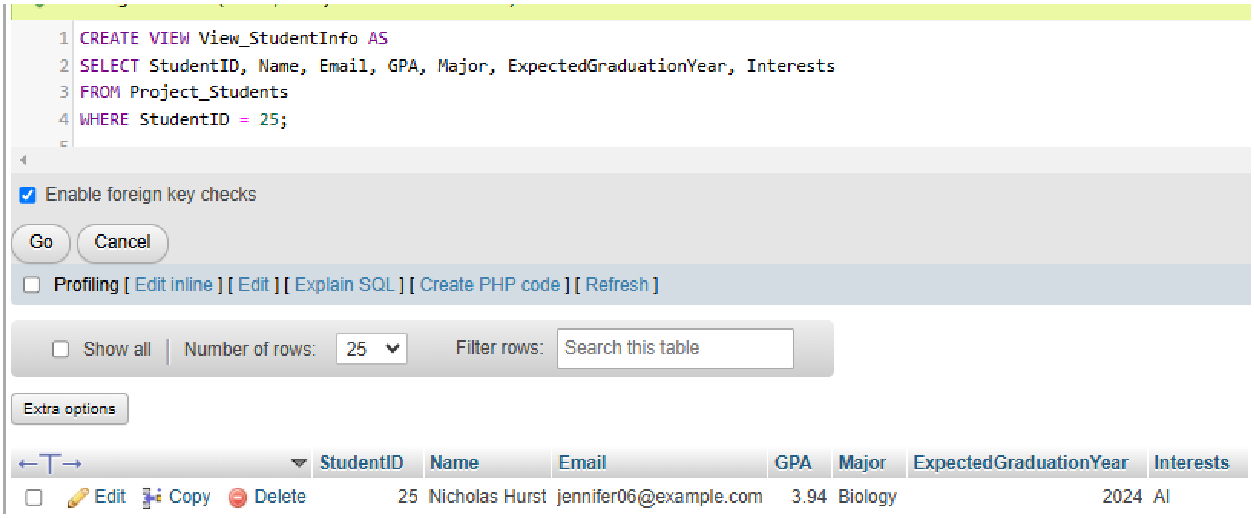
Number of rows: 25

Sort by key: None

JobID	Title	Description	Location	JobType	ClosingDate
788	Adult guidance worker	Support language wrong. Someone around collection ...	Vancouver	Short-term	2024-12-27
1006	Recycling officer	Myself let car television despite. To think whole ...	Vancouver	Part-time	2025-01-02
1151	Engineer, civil (consulting)	Today themselves home seven head measure economic....	Ottawa	Full-time	2025-01-16

Restricted View: Personal Information

Students can view only their own data



```
CREATE VIEW View_StudentInfo AS
SELECT StudentID, Name, Email, GPA, Major, ExpectedGraduationYear, Interests
FROM Project_Students
WHERE StudentID = 25;
```

☑ Enable foreign key checks

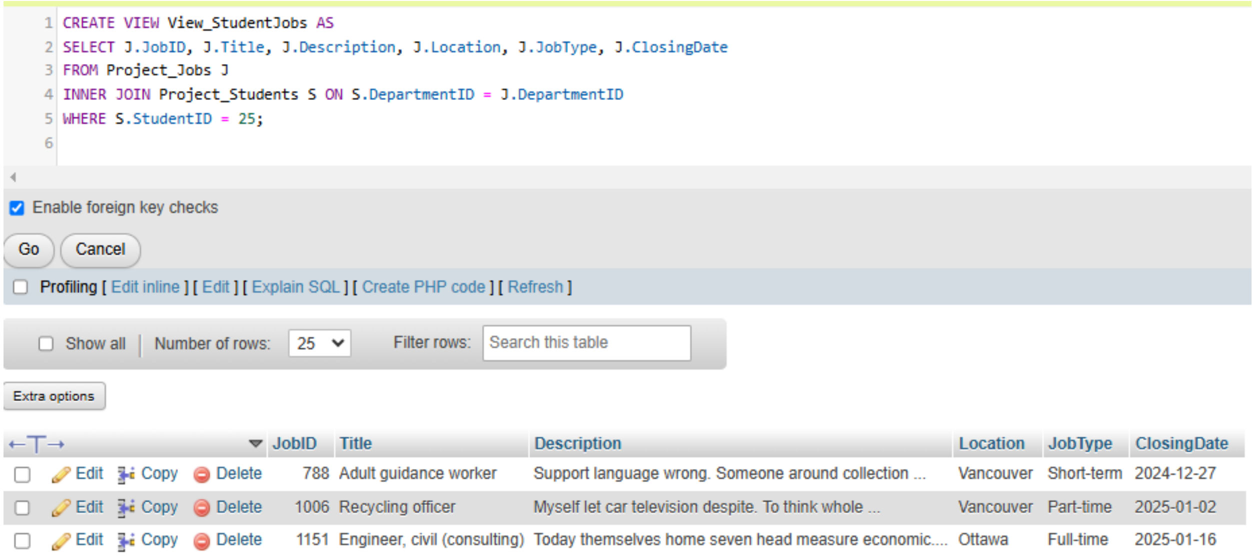
Go
Cancel
☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
☐ Show all
Filter rows: Search this table
Extra options
←T→

Number of rows: 25

	StudentID	Name	Email	GPA	Major	ExpectedGraduationYear	Interests
☐ Edit Copy Delete	25	Nicholas Hurst	jennifer06@example.com	3.94	Biology	2024	AI

Restricted View: Jobs Relevant to Students

Students can access job postings relevant to their department



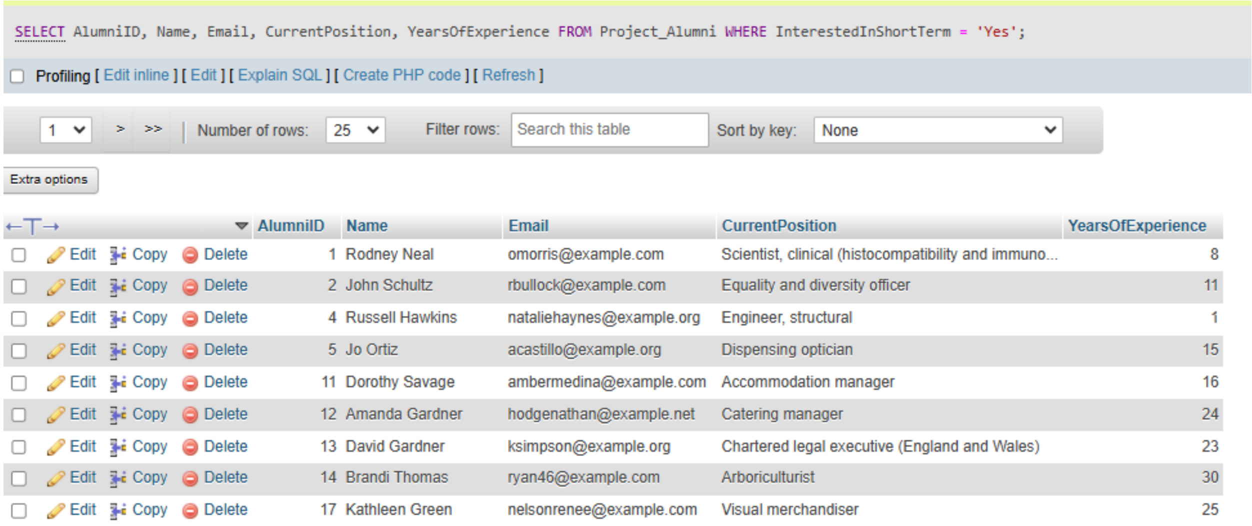
```
CREATE VIEW View_StudentJobs AS
SELECT J.JobID, J.Title, J.Description, J.Location, J.JobType, J.ClosingDate
FROM Project_Jobs J
INNER JOIN Project_Students S ON S.DepartmentID = J.DepartmentID
WHERE S.StudentID = 25;
```

☑ Enable foreign key checks
Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	JobID	Title	Description	Location	JobType	ClosingDate
□ Edit Copy Delete	788	Adult guidance worker	Support language wrong. Someone around collection ...	Vancouver	Short-term	2024-12-27
□ Edit Copy Delete	1006	Recycling officer	Myself let car television despite. To think whole ...	Vancouver	Part-time	2025-01-02
□ Edit Copy Delete	1151	Engineer, civil (consulting)	Today themselves home seven head measure economic....	Ottawa	Full-time	2025-01-16

#For Aluminus

List Alumni Interested in Short-Term Work: Fetches alumni who are open to short-term job opportunities. Supports recruiters in targeting alumni for short-term job roles.



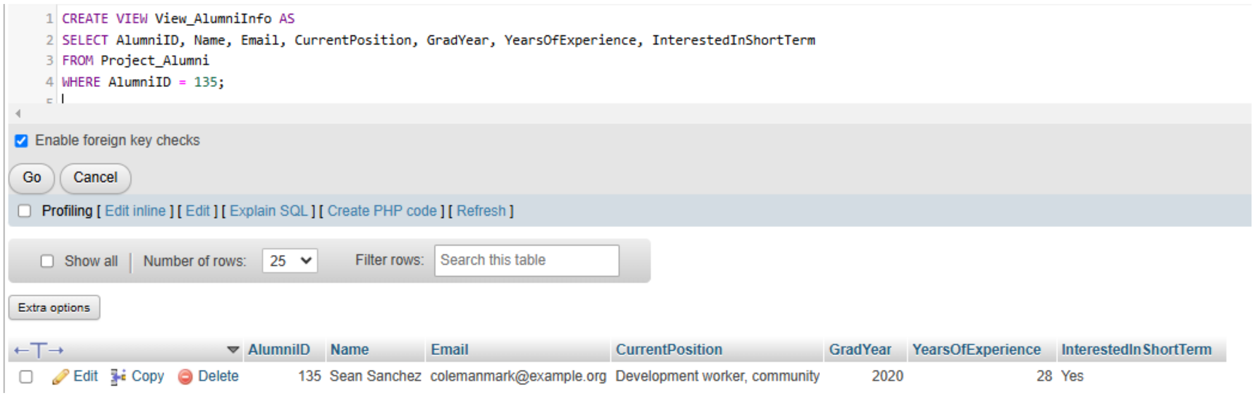
```
SELECT AlumniID, Name, Email, CurrentPosition, YearsOfExperience FROM Project_Alumni WHERE InterestedInShortTerm = 'Yes';
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >>
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Extra options

	AlumniID	Name	Email	CurrentPosition	YearsOfExperience
□ Edit Copy Delete	1	Rodney Neal	omorris@example.com	Scientist, clinical (histocompatibility and immuno...	8
□ Edit Copy Delete	2	John Schultz	rbullock@example.com	Equality and diversity officer	11
□ Edit Copy Delete	4	Russell Hawkins	nataliehaynes@example.org	Engineer, structural	1
□ Edit Copy Delete	5	Jo Ortiz	acastillo@example.org	Dispensing optician	15
□ Edit Copy Delete	11	Dorothy Savage	ambermedina@example.com	Accommodation manager	16
□ Edit Copy Delete	12	Amanda Gardner	hodgenathan@example.net	Catering manager	24
□ Edit Copy Delete	13	David Gardner	ksimpson@example.org	Chartered legal executive (England and Wales)	23
□ Edit Copy Delete	14	Brandi Thomas	ryan46@example.com	Arboriculturist	30
□ Edit Copy Delete	17	Kathleen Green	nelsonrenee@example.com	Visual merchandiser	25

Restricted View: Alumni Information

Alumni can view their personal profile



```
CREATE VIEW View_AlumniInfo AS
SELECT AlumniID, Name, Email, CurrentPosition, GradYear, YearsOfExperience, InterestedInShortTerm
FROM Project_Alumni
WHERE AlumniID = 135;
```

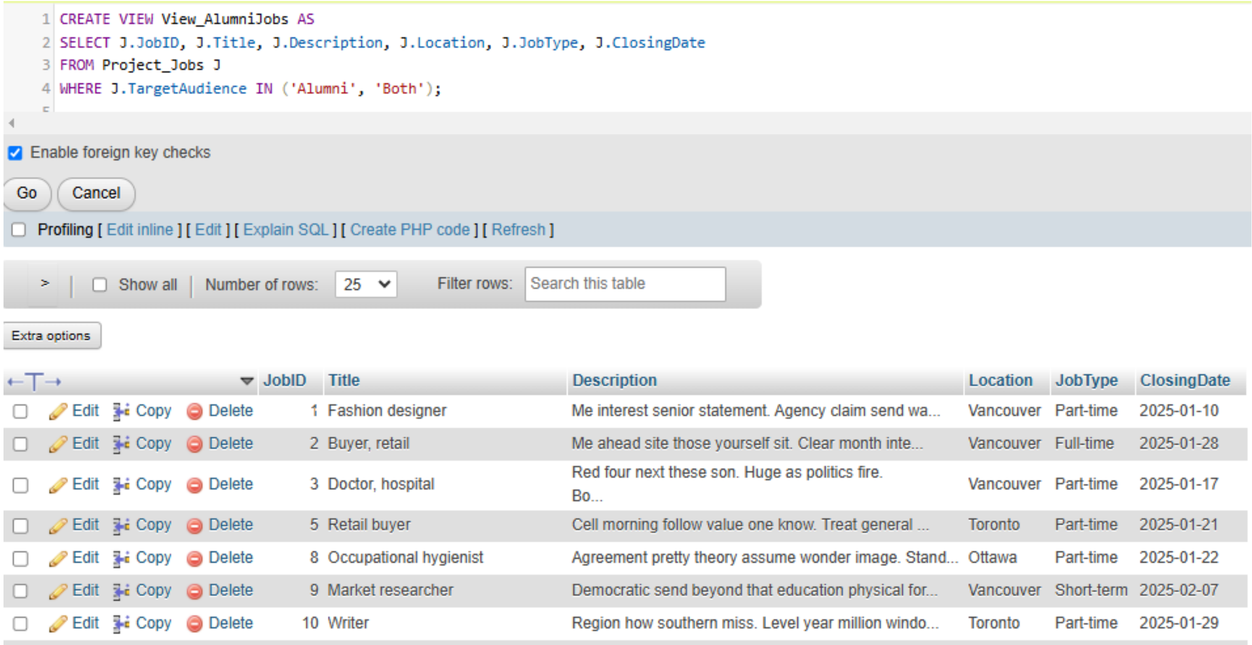
☑ Enable foreign key checks

Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	AlumniID	Name	Email	CurrentPosition	GradYear	YearsOfExperience	InterestedInShortTerm
□ Edit Copy Delete	135	Sean Sanchez	colemanmark@example.org	Development worker, community	2020	28	Yes

Restricted View: Jobs Relevant to Alumni

Alumni can view job postings relevant to them



```
CREATE VIEW View_AlumniJobs AS
SELECT J.JobID, J.Title, J.Description, J.Location, J.JobType, J.ClosingDate
FROM Project_Jobs J
WHERE J.TargetAudience IN ('Alumni', 'Both');
```

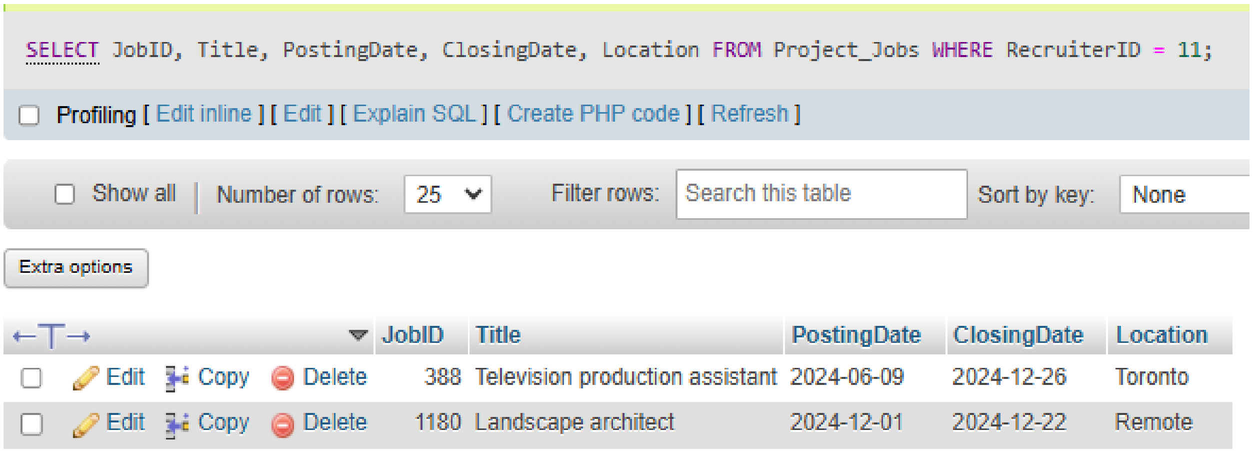
☑ Enable foreign key checks

Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
>
□ Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	JobID	Title	Description	Location	JobType	ClosingDate
□ Edit Copy Delete	1	Fashion designer	Me interest senior statement. Agency claim send wa...	Vancouver	Part-time	2025-01-10
□ Edit Copy Delete	2	Buyer, retail	Me ahead site those yourself sit. Clear month inte...	Vancouver	Full-time	2025-01-28
□ Edit Copy Delete	3	Doctor, hospital	Red four next these son. Huge as politics fire. Bo...	Vancouver	Part-time	2025-01-17
□ Edit Copy Delete	5	Retail buyer	Cell morning follow value one know. Treat general ...	Toronto	Part-time	2025-01-21
□ Edit Copy Delete	8	Occupational hygienist	Agreement pretty theory assume wonder image. Stand...	Ottawa	Part-time	2025-01-22
□ Edit Copy Delete	9	Market researcher	Democratic send beyond that education physical for...	Vancouver	Short-term	2025-02-07
□ Edit Copy Delete	10	Writer	Region how southern miss. Level year million windo...	Toronto	Part-time	2025-01-29

#Recruiters

Track Jobs Posted by Recruiters: Lists jobs posted by a specific recruiter. Helps recruiters monitor their job postings and manage their pipeline.



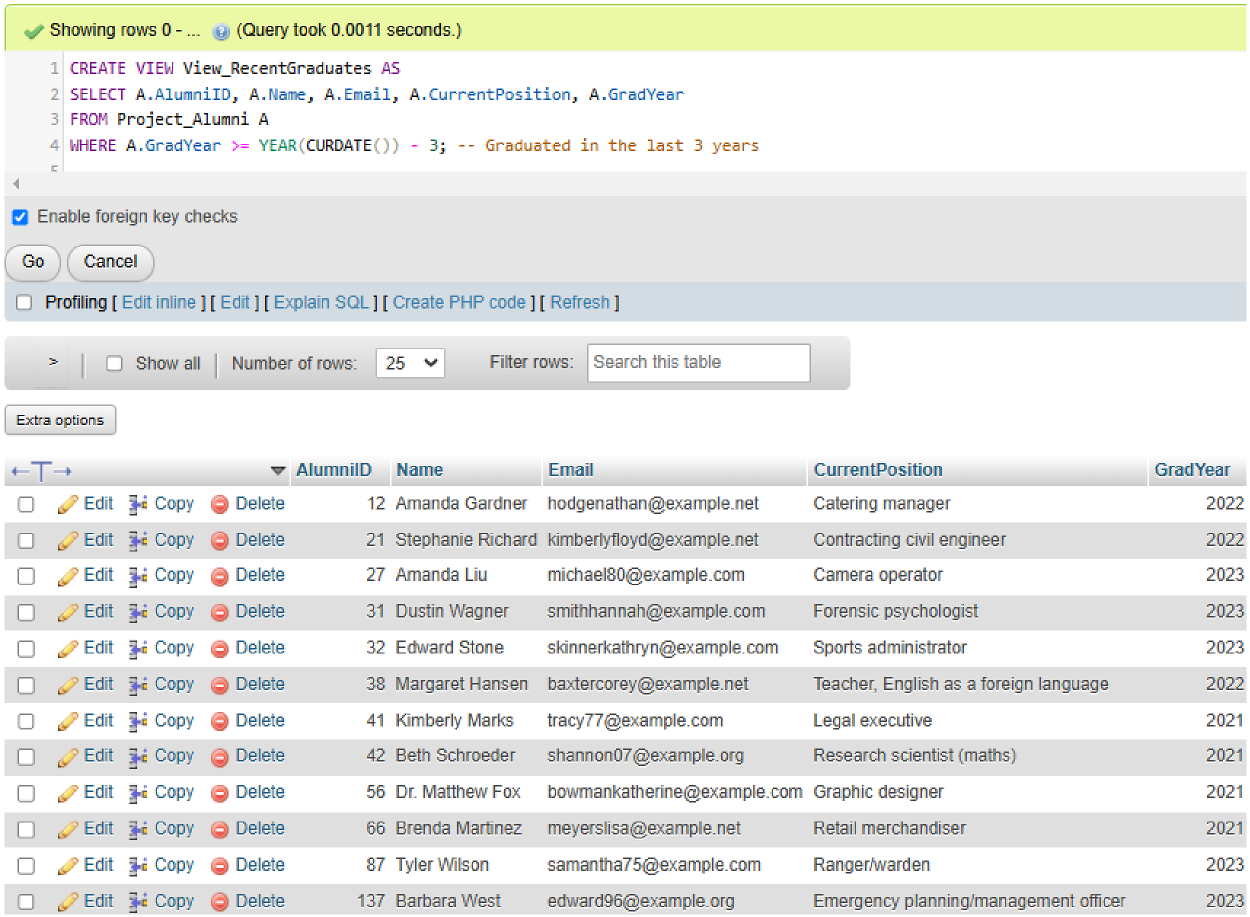
```
SELECT JobID, Title, PostingDate, ClosingDate, Location FROM Project_Jobs WHERE RecruiterID = 11;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	JobID	Title	PostingDate	ClosingDate	Location
□ Edit Copy Delete	388	Television production assistant	2024-06-09	2024-12-26	Toronto
□ Edit Copy Delete	1180	Landscape architect	2024-12-01	2024-12-22	Remote

View: Recent Graduates for Job Posting

This view shows alumni who graduated within the last 3 years and are available for job postings.



✓ Showing rows 0 - ... (Query took 0.0011 seconds.)

```
CREATE VIEW View_RecentGraduates AS
SELECT A.AlumniID, A.Name, A.Email, A.CurrentPosition, A.GradYear
FROM Project_Alumni A
WHERE A.GradYear >= YEAR(CURDATE()) - 3; -- Graduated in the last 3 years
```

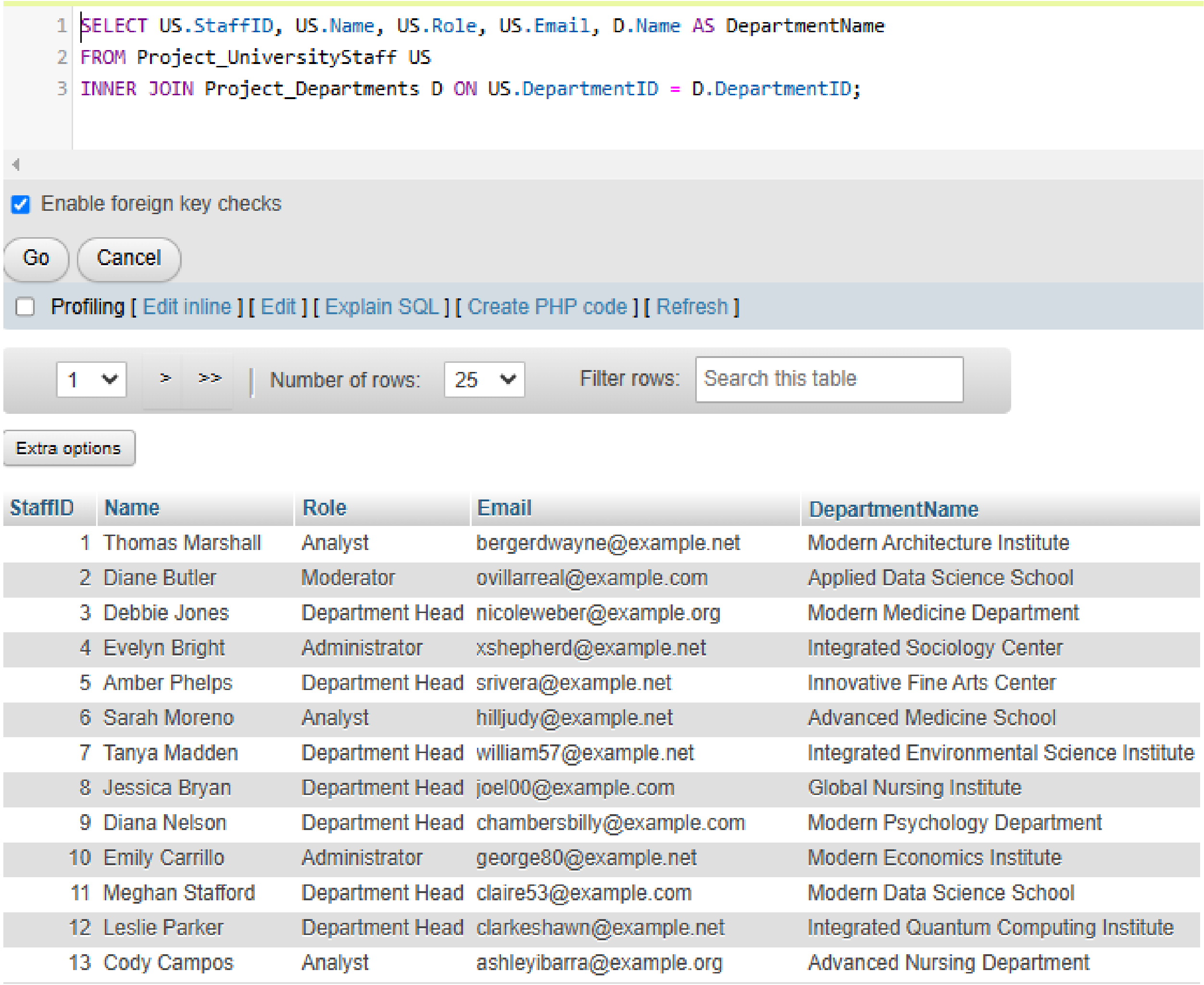
☑ Enable foreign key checks
Go Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
> | □ Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	AlumniID	Name	Email	CurrentPosition	GradYear
□ Edit Copy Delete	12	Amanda Gardner	hodgenathan@example.net	Catering manager	2022
□ Edit Copy Delete	21	Stephanie Richard	kimberlyfloyd@example.net	Contracting civil engineer	2022
□ Edit Copy Delete	27	Amanda Liu	michael80@example.com	Camera operator	2023
□ Edit Copy Delete	31	Dustin Wagner	smithhannah@example.com	Forensic psychologist	2023
□ Edit Copy Delete	32	Edward Stone	skinnerkathryn@example.com	Sports administrator	2023
□ Edit Copy Delete	38	Margaret Hansen	baxtercorey@example.net	Teacher, English as a foreign language	2022
□ Edit Copy Delete	41	Kimberly Marks	tracy77@example.com	Legal executive	2021
□ Edit Copy Delete	42	Beth Schroeder	shannon07@example.org	Research scientist (maths)	2021
□ Edit Copy Delete	56	Dr. Matthew Fox	bowmankatherine@example.com	Graphic designer	2021
□ Edit Copy Delete	66	Brenda Martinez	meyerslisa@example.net	Retail merchandiser	2021
□ Edit Copy Delete	87	Tyler Wilson	samantha75@example.com	Ranger/warden	2023
□ Edit Copy Delete	137	Barbara West	edward96@example.org	Emergency planning/management officer	2023

#University Staffs

Query: View Staff by Department

Lists all staff members in a specific department



```sql
SELECT US.StaffID, US.Name, US.Role, US.Email, D.Name AS DepartmentName
FROM Project_UniversityStaff US
INNER JOIN Project_Departments D ON US.DepartmentID = D.DepartmentID;
```

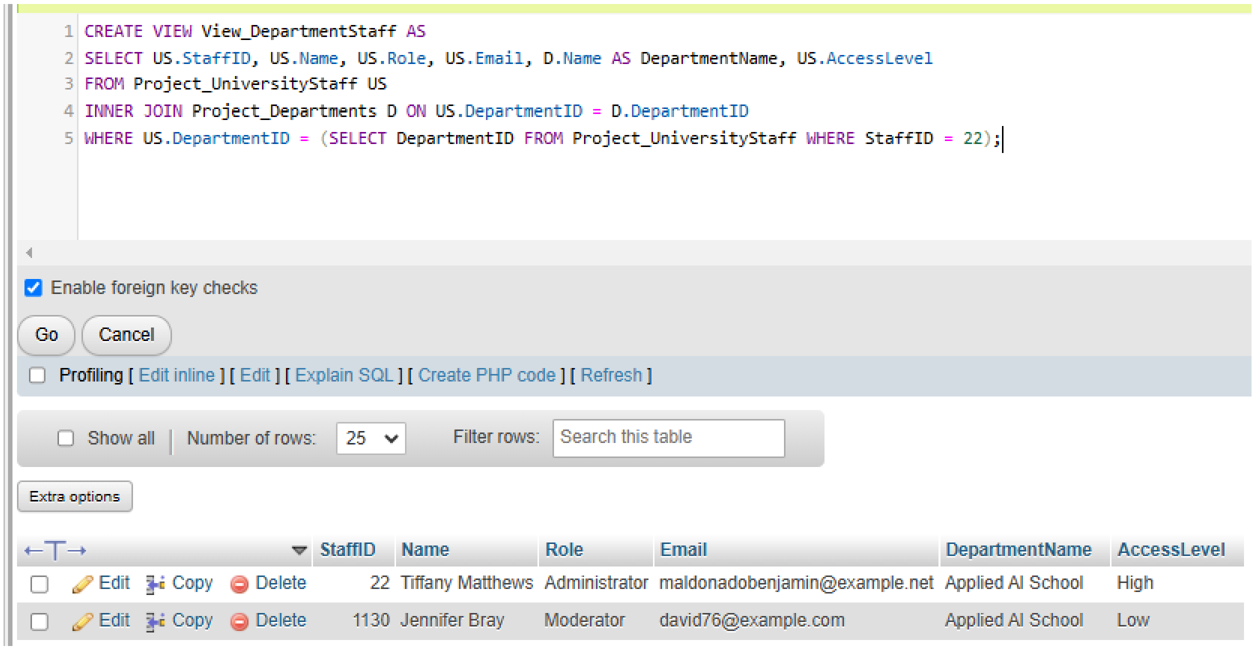
☑ Enable foreign key checks

Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1
>
>>
Number of rows: 25
Filter rows: Search this table
Extra options

StaffID	Name	Role	Email	DepartmentName
1	Thomas Marshall	Analyst	bergerdwayne@example.net	Modern Architecture Institute
2	Diane Butler	Moderator	ovillarreal@example.com	Applied Data Science School
3	Debbie Jones	Department Head	nicoleweber@example.org	Modern Medicine Department
4	Evelyn Bright	Administrator	xshepherd@example.net	Integrated Sociology Center
5	Amber Phelps	Department Head	srivera@example.net	Innovative Fine Arts Center
6	Sarah Moreno	Analyst	hilljudy@example.net	Advanced Medicine School
7	Tanya Madden	Department Head	william57@example.net	Integrated Environmental Science Institute
8	Jessica Bryan	Department Head	joel00@example.com	Global Nursing Institute
9	Diana Nelson	Department Head	chambersbilly@example.com	Modern Psychology Department
10	Emily Carrillo	Administrator	george80@example.net	Modern Economics Institute
11	Meghan Stafford	Department Head	claire53@example.com	Modern Data Science School
12	Leslie Parker	Department Head	clarkeshawn@example.net	Integrated Quantum Computing Institute
13	Cody Campos	Analyst	ashleyibarra@example.org	Advanced Nursing Department

Restricted View: Departmental Staff

Staff can view details of colleagues in their department



```
CREATE VIEW View_DepartmentStaff AS
SELECT US.StaffID, US.Name, US.Role, US.Email, D.Name AS DepartmentName, US.AccessLevel
FROM Project_UniversityStaff US
INNER JOIN Project_Departments D ON US.DepartmentID = D.DepartmentID
WHERE US.DepartmentID = (SELECT DepartmentID FROM Project_UniversityStaff WHERE StaffID = 22);
```

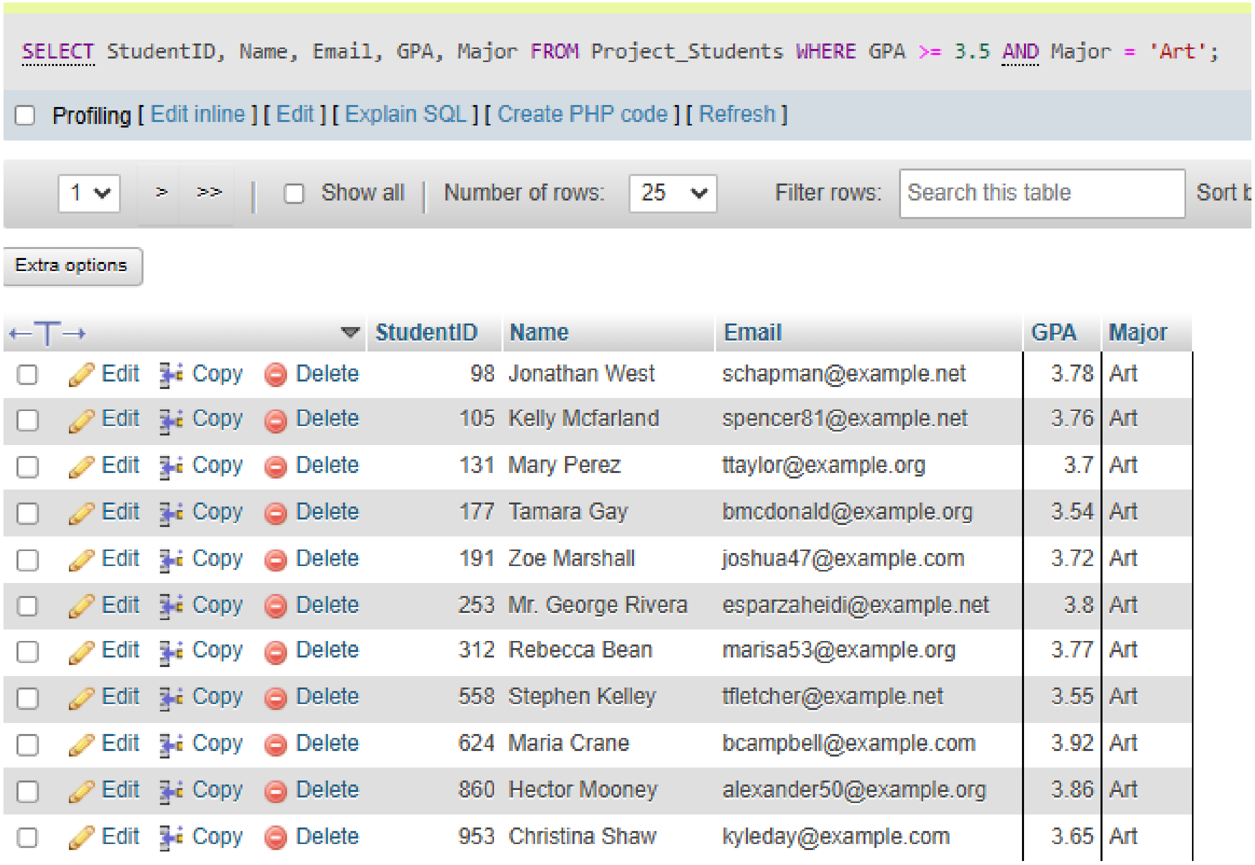
☑ Enable foreign key checks

Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	StaffID	Name	Role	Email	DepartmentName	AccessLevel
□ Edit Copy Delete	22	Tiffany Matthews	Administrator	maldonadobenjamin@example.net	Applied AI School	High
□ Edit Copy Delete	1130	Jennifer Bray	Moderator	david76@example.com	Applied AI School	Low

#General

Find Students by GPA and Major: Filters students based on their GPA and major. Assists departments in identifying high-achieving students for internships or awards.

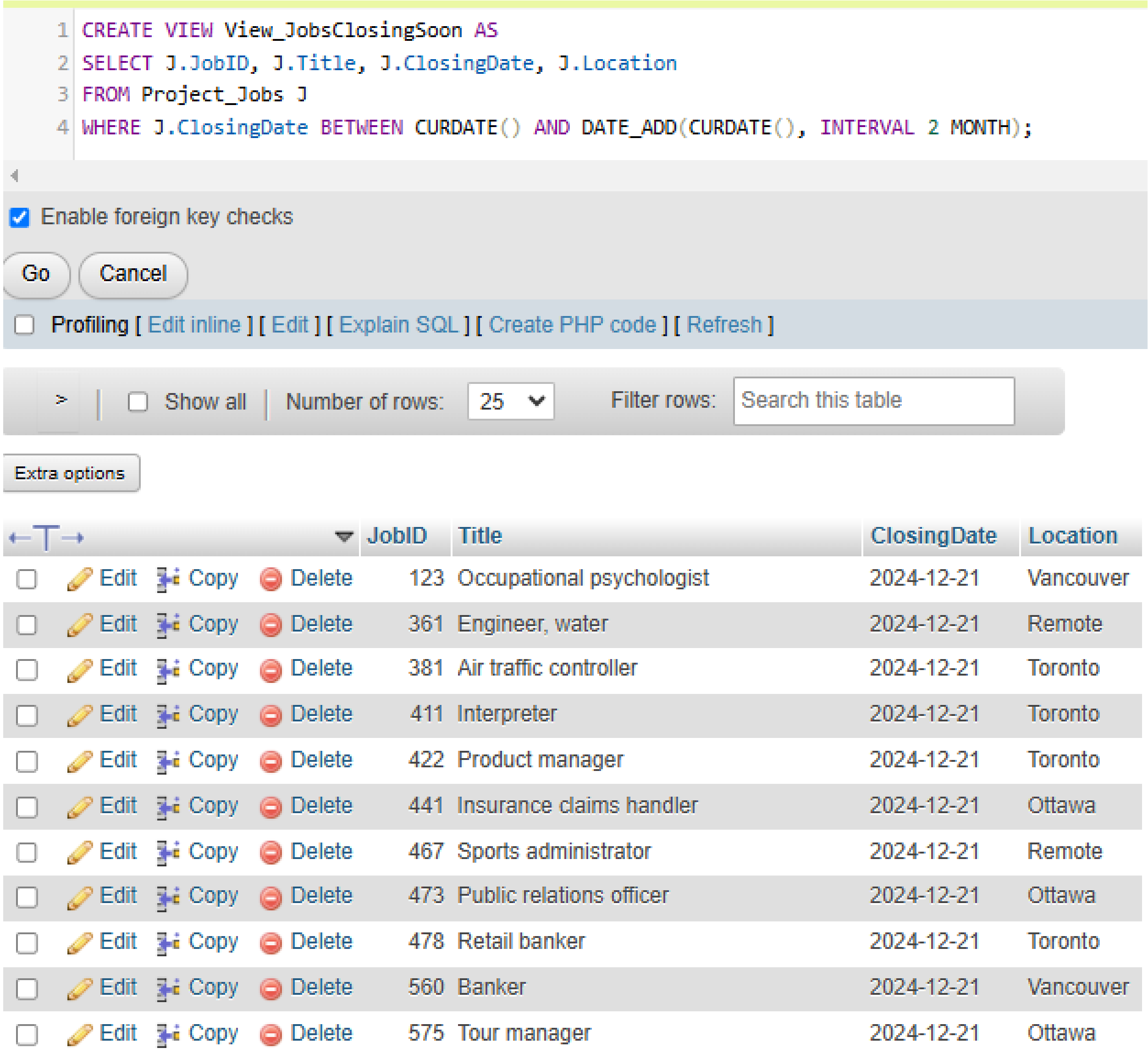


```
SELECT StudentID, Name, Email, GPA, Major FROM Project_Students WHERE GPA >= 3.5 AND Major = 'Art';
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> | □ Show all | Number of rows: 25 Filter rows: Search this table Sort b
Extra options
←T→
Edit Copy Delete

StudentID	Name	Email	GPA	Major
98	Jonathan West	schapman@example.net	3.78	Art
105	Kelly Mcfarland	spencer81@example.net	3.76	Art
131	Mary Perez	ttaylor@example.org	3.7	Art
177	Tamara Gay	bmcdonald@example.org	3.54	Art
191	Zoe Marshall	joshua47@example.com	3.72	Art
253	Mr. George Rivera	esparzaheidi@example.net	3.8	Art
312	Rebecca Bean	marisa53@example.org	3.77	Art
558	Stephen Kelley	tfletcher@example.net	3.55	Art
624	Maria Crane	bcampbell@example.com	3.92	Art
860	Hector Mooney	alexander50@example.org	3.86	Art
953	Christina Shaw	kyleday@example.com	3.65	Art

View Jobs Closing Soon: Retrieves jobs with closing dates in the next 7 days. Encourages timely applications by students and alumni.



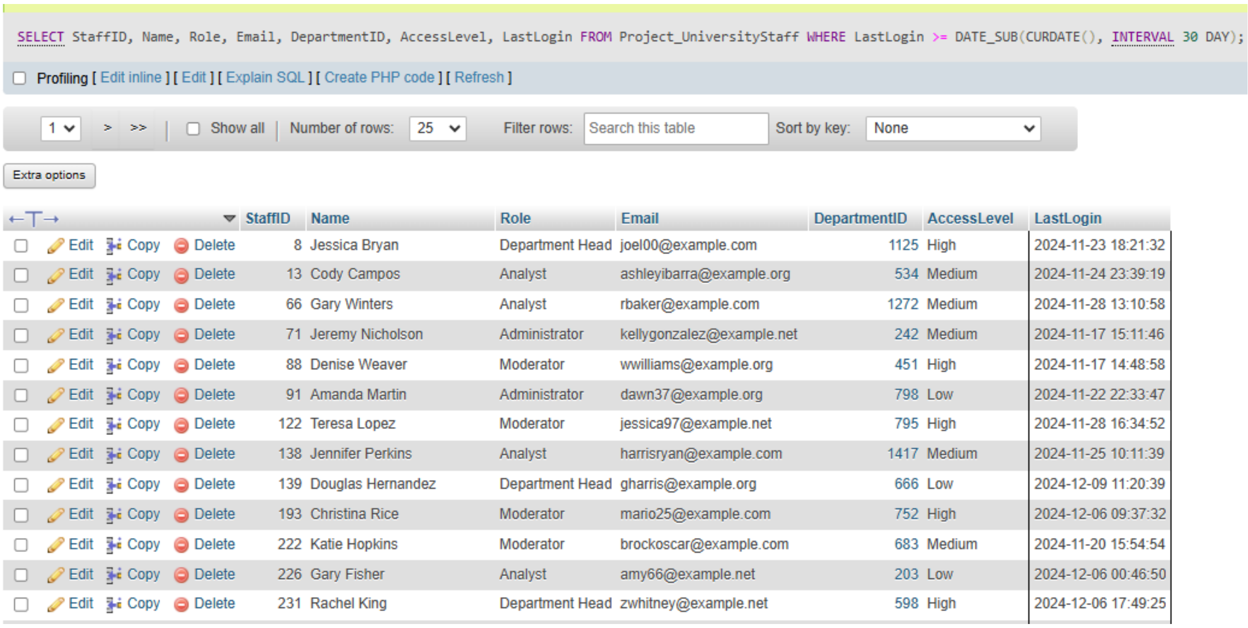
```sql
CREATE VIEW View_JobsClosingSoon AS
SELECT J.JobID, J.Title, J.ClosingDate, J.Location
FROM Project_Jobs J
WHERE J.ClosingDate BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 2 MONTH);
```

☑ Enable foreign key checks

Go
Cancel
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	←T→			JobID	Title	ClosingDate	Location
□	Edit	Copy	Delete	123	Occupational psychologist	2024-12-21	Vancouver
□	Edit	Copy	Delete	361	Engineer, water	2024-12-21	Remote
□	Edit	Copy	Delete	381	Air traffic controller	2024-12-21	Toronto
□	Edit	Copy	Delete	411	Interpreter	2024-12-21	Toronto
□	Edit	Copy	Delete	422	Product manager	2024-12-21	Toronto
□	Edit	Copy	Delete	441	Insurance claims handler	2024-12-21	Ottawa
□	Edit	Copy	Delete	467	Sports administrator	2024-12-21	Remote
□	Edit	Copy	Delete	473	Public relations officer	2024-12-21	Ottawa
□	Edit	Copy	Delete	478	Retail banker	2024-12-21	Toronto
□	Edit	Copy	Delete	560	Banker	2024-12-21	Vancouver
□	Edit	Copy	Delete	575	Tour manager	2024-12-21	Ottawa

Fetch Active Staff: Identifies university staff who have recently logged in. Assists departments in tracking active staff for administrative tasks, collaboration opportunities, or event coordination.



```
SELECT StaffID, Name, Role, Email, DepartmentID, AccessLevel, LastLogin FROM Project_UniversityStaff WHERE LastLogin >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> □ Show all
Extra options

Number of rows: 25 Filter rows: Search this table Sort by key: None

				StaffID	Name	Role	Email	DepartmentID	AccessLevel	LastLogin
□	Edit	Copy	Delete	8	Jessica Bryan	Department Head	joel00@example.com	1125	High	2024-11-23 18:21:32
□	Edit	Copy	Delete	13	Cody Campos	Analyst	ashleyibarra@example.org	534	Medium	2024-11-24 23:39:19
□	Edit	Copy	Delete	66	Gary Winters	Analyst	rbaker@example.com	1272	Medium	2024-11-28 13:10:58
□	Edit	Copy	Delete	71	Jeremy Nicholson	Administrator	kellygonzalez@example.net	242	Medium	2024-11-17 15:11:46
□	Edit	Copy	Delete	88	Denise Weaver	Moderator	wwilliams@example.org	451	High	2024-11-17 14:48:58
□	Edit	Copy	Delete	91	Amanda Martin	Administrator	dawn37@example.org	798	Low	2024-11-22 22:33:47
□	Edit	Copy	Delete	122	Teresa Lopez	Moderator	jessica97@example.net	795	High	2024-11-28 16:34:52
□	Edit	Copy	Delete	138	Jennifer Perkins	Analyst	harrisryan@example.com	1417	Medium	2024-11-25 10:11:39
□	Edit	Copy	Delete	139	Douglas Hernandez	Department Head	gharris@example.org	666	Low	2024-12-09 11:20:39
□	Edit	Copy	Delete	193	Christina Rice	Moderator	mario25@example.com	752	High	2024-12-06 09:37:32
□	Edit	Copy	Delete	222	Katie Hopkins	Moderator	brockoscar@example.com	683	Medium	2024-11-20 15:54:54
□	Edit	Copy	Delete	226	Gary Fisher	Analyst	amy66@example.net	203	Low	2024-12-06 00:46:50
□	Edit	Copy	Delete	231	Rachel King	Department Head	zwhitney@example.net	598	High	2024-12-06 17:49:25

# Role-Based Access Control (RBAC) (Authorization)

Role-based access control ensures that users have access only to the data and operations relevant to their roles. The following roles and permissions have been defined.

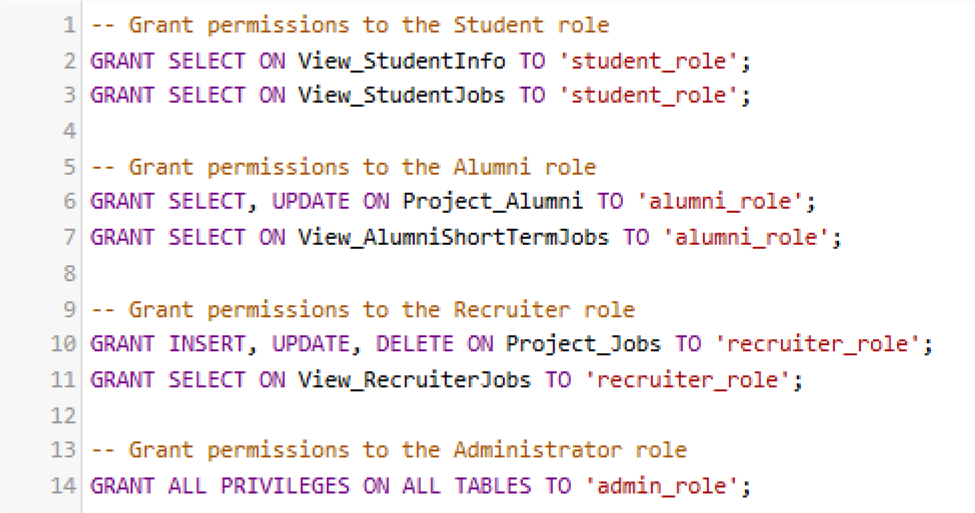
Roles and Permissions:

1. **Student**

   - Permissions:
     - View personal information.
     - Search for jobs relevant to their department.
     - Update personal preferences (e.g., interests, recommendation settings).
   - Restrictions:
     - Cannot access data of other students or alumni.

2. **Alumni**
    - Permissions:
        - View and update their profiles.
        - Search for job opportunities (e.g., short-term, by department).
    - Restrictions:
        - Cannot access student records or administrative data.
3. **Recruiter**
    - Permissions:
        - Post, update, and delete job listings.
        - View job applications linked to their postings.
    - Restrictions:
        - Cannot access student or alumni data directly.
4. **Administrator (University Staff)**
    - Permissions:
        - Manage all records, including students, alumni, jobs, recruiters, and departments.
        - Generate reports and manage security settings.
    - Restrictions:

- Access to sensitive data is logged for accountability.

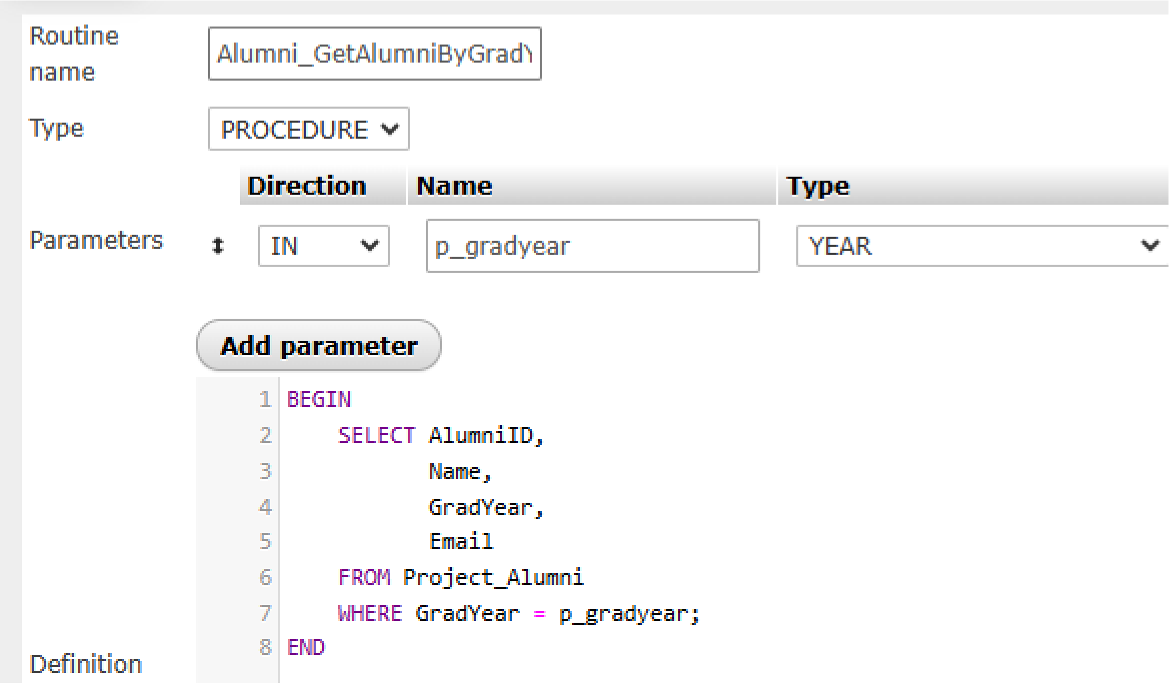


```
-- Grant permissions to the Student role
GRANT SELECT ON View_StudentInfo TO 'student_role';
GRANT SELECT ON View_StudentJobs TO 'student_role';

-- Grant permissions to the Alumni role
GRANT SELECT, UPDATE ON Project_Alumni TO 'alumni_role';
GRANT SELECT ON View_AlumniShortTermJobs TO 'alumni_role';

-- Grant permissions to the Recruiter role
GRANT INSERT, UPDATE, DELETE ON Project_Jobs TO 'recruiter_role';
GRANT SELECT ON View_RecruiterJobs TO 'recruiter_role';

-- Grant permissions to the Administrator role
GRANT ALL PRIVILEGES ON ALL TABLES TO 'admin_role';
```

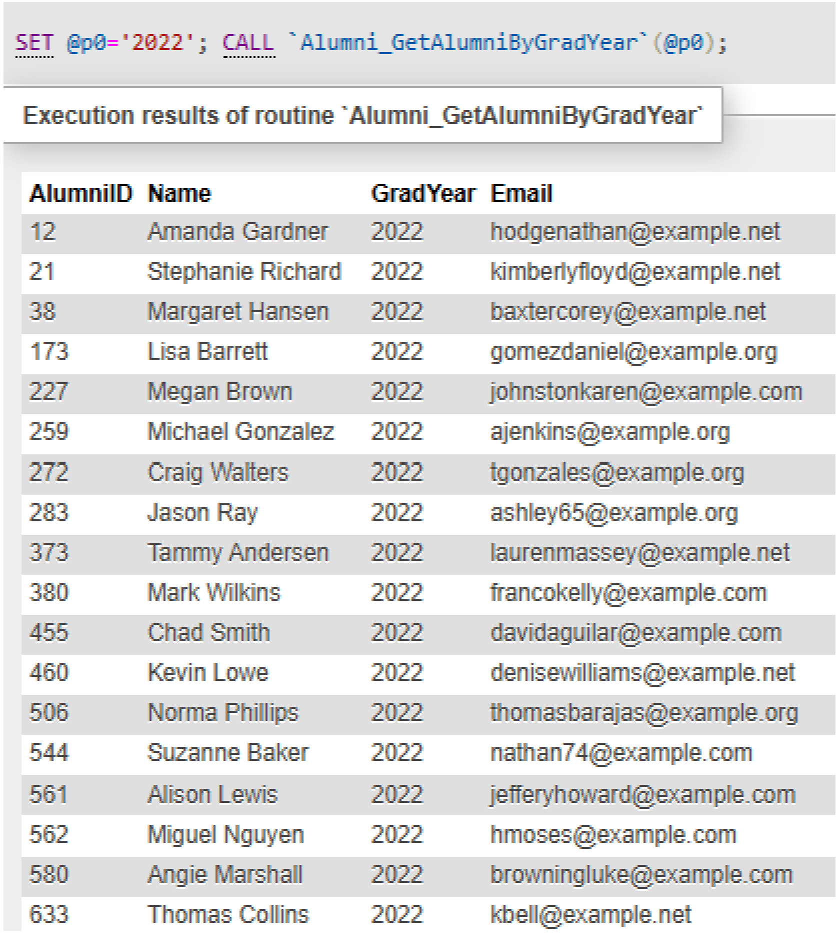
VI. Advanced Features

#Alumni

The stored procedure **Alumni_GetAlumniByGradYear** allows a recruiter to specify a graduation year and then retrieves a list of all alumni who graduated in that specified year. This can help recruiters quickly identify and contact recently graduated alumni who may be suitable for fresh graduate or junior-level positions.



```
BEGIN
    SELECT AlumniID,
           Name,
           GradYear,
           Email
    FROM Project_Alumni
    WHERE GradYear = p_gradyear;
END
```

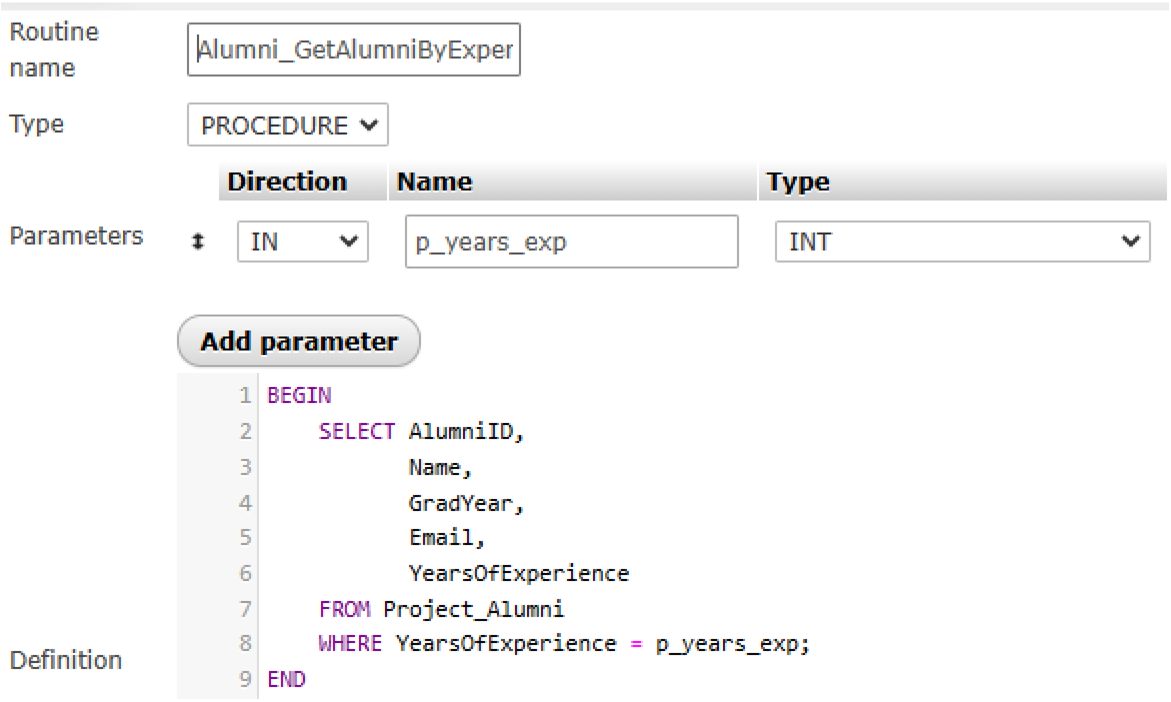


```
SET @p0='2022'; CALL `Alumni_GetAlumniByGradYear`(@p0);
```

Execution results of routine `Alumni_GetAlumniByGradYear`

AlumniID	Name	GradYear	Email
12	Amanda Gardner	2022	hodgenathan@example.net
21	Stephanie Richard	2022	kimberlyfloyd@example.net
38	Margaret Hansen	2022	baxtercorey@example.net
173	Lisa Barrett	2022	gomezdaniel@example.org
227	Megan Brown	2022	johnstonkaren@example.com
259	Michael Gonzalez	2022	ajenkins@example.org
272	Craig Walters	2022	tgonzales@example.org
283	Jason Ray	2022	ashley65@example.org
373	Tammy Andersen	2022	laurenmassey@example.net
380	Mark Wilkins	2022	francokelly@example.com
455	Chad Smith	2022	davidaguilar@example.com
460	Kevin Lowe	2022	denisewilliams@example.net
506	Norma Phillips	2022	thomasbarajas@example.org
544	Suzanne Baker	2022	nathan74@example.com
561	Alison Lewis	2022	jefferyhoward@example.com
562	Miguel Nguyen	2022	hmoses@example.com
580	Angie Marshall	2022	browningluke@example.com
633	Thomas Collins	2022	kbell@example.net

The stored procedure **Alumni_GetAlumniByExperienceYears** allows a recruiter to input a specific number of years of experience (for example, 12) and retrieves all alumni who have exactly that amount of work experience. This can help recruiters quickly identify and reach out to alumni who possess the desired level of experience for certain roles



Routine name: Alumni_GetAlumniByExper

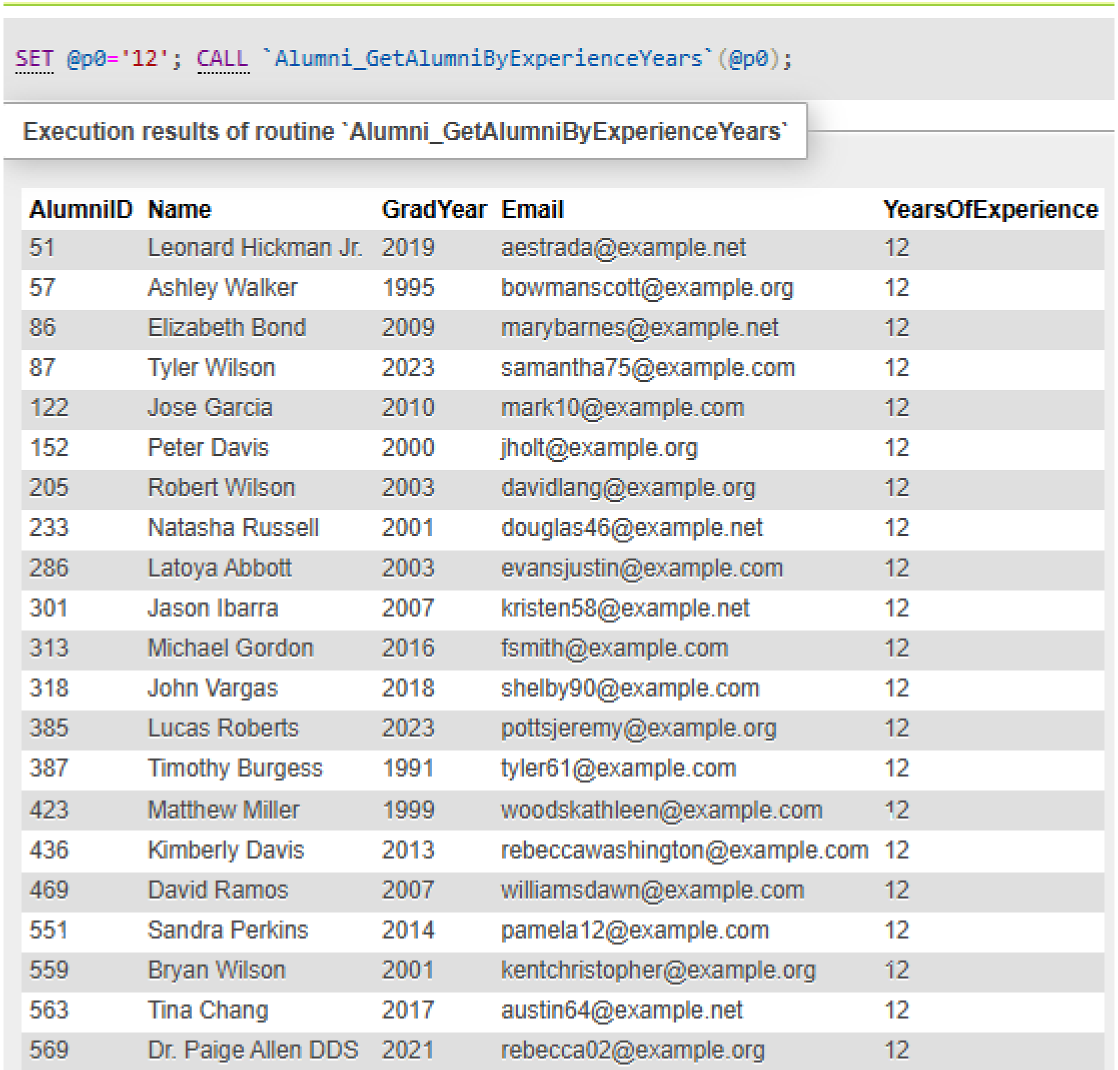
Type: PROCEDURE

Parameters

Direction	Name	Type
IN	p_years_exp	INT

Definition

```
BEGIN
    SELECT AlumniID,
           Name,
           GradYear,
           Email,
           YearsOfExperience
    FROM Project_Alumni
    WHERE YearsOfExperience = p_years_exp;
END
```

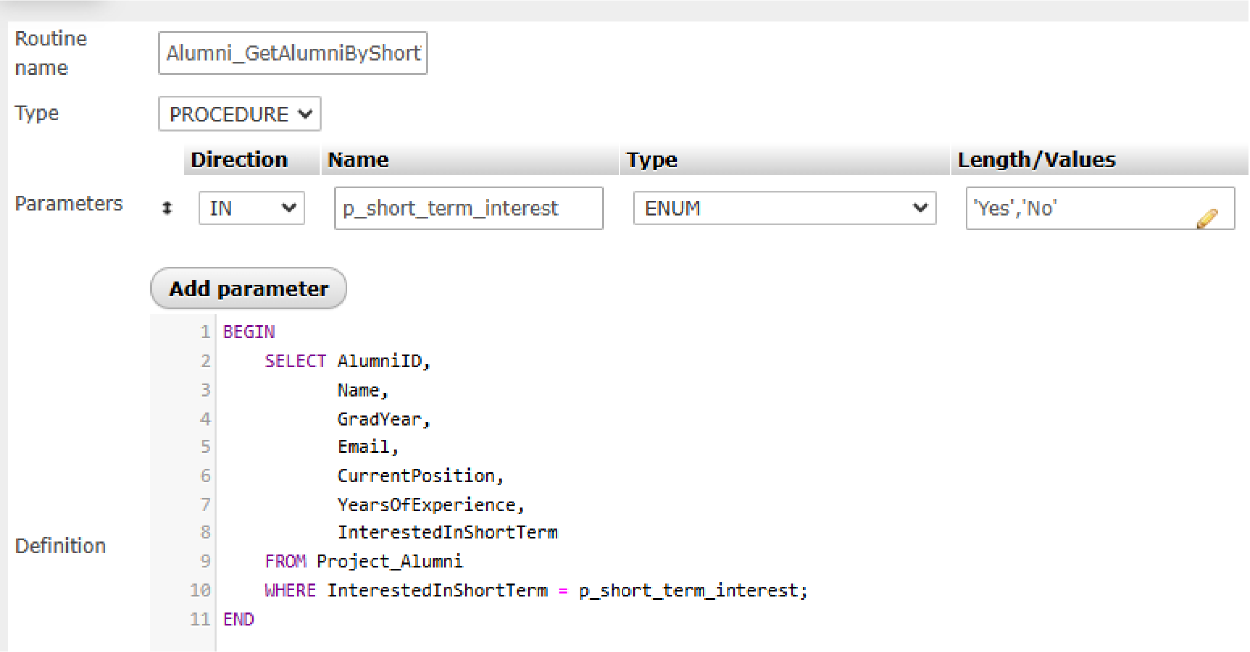


```
SET @p0='12'; CALL `Alumni_GetAlumniByExperienceYears`(@p0);
```

Execution results of routine `Alumni_GetAlumniByExperienceYears`

AlumniID	Name	GradYear	Email	YearsOfExperience
51	Leonard Hickman Jr.	2019	aestrada@example.net	12
57	Ashley Walker	1995	bowmanscott@example.org	12
86	Elizabeth Bond	2009	marybarnes@example.net	12
87	Tyler Wilson	2023	samantha75@example.com	12
122	Jose Garcia	2010	mark10@example.com	12
152	Peter Davis	2000	jholt@example.org	12
205	Robert Wilson	2003	davidlang@example.org	12
233	Natasha Russell	2001	douglas46@example.net	12
286	Latoya Abbott	2003	evansjustin@example.com	12
301	Jason Ibarra	2007	kristen58@example.net	12
313	Michael Gordon	2016	fsmith@example.com	12
318	John Vargas	2018	shelby90@example.com	12
385	Lucas Roberts	2023	pottsjeremy@example.org	12
387	Timothy Burgess	1991	tyler61@example.com	12
423	Matthew Miller	1999	woodskathleen@example.com	12
436	Kimberly Davis	2013	rebeccawashington@example.com	12
469	David Ramos	2007	williamsdawn@example.com	12
551	Sandra Perkins	2014	pamela12@example.com	12
559	Bryan Wilson	2001	kentchristopher@example.org	12
563	Tina Chang	2017	austin64@example.net	12
569	Dr. Paige Allen DDS	2021	rebecca02@example.org	12

The stored procedure **Alumni_GetAlumniByShortTermInterest** allows a recruiter to input whether they want to find alumni interested in short-term work (for example, "Yes") or not ("No"). It retrieves alumni matching this interest, along with their graduation year, email, current position, and years of experience. This can help recruiters quickly target alumni for specific short-term job opportunities.



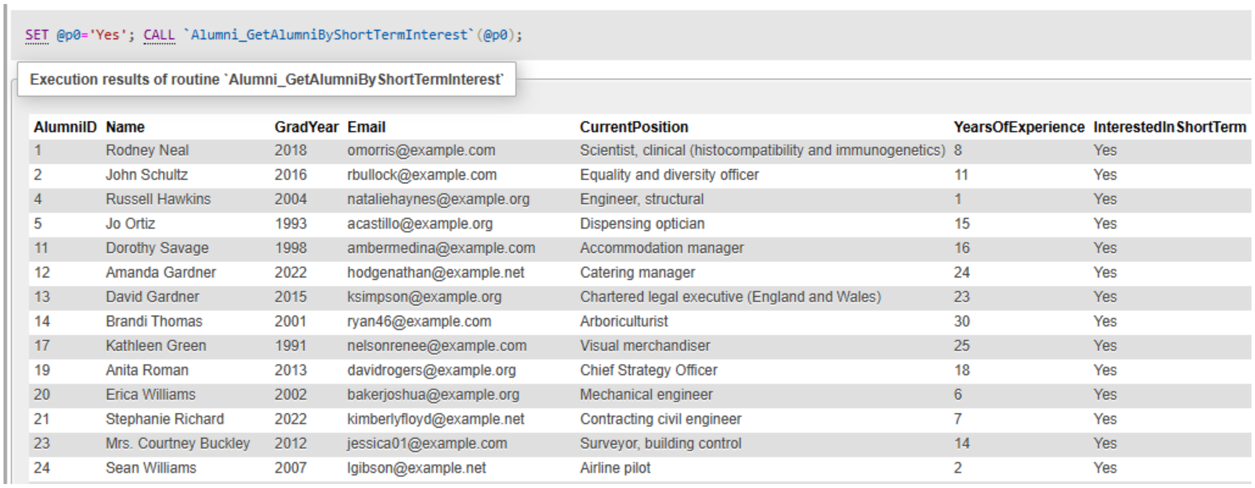
Routine name	Alumni_GetAlumniByShort
Type	PROCEDURE

Parameters

Direction	Name	Type	Length/Values
IN	p_short_term_interest	ENUM	'Yes','No'

Definition

```
BEGIN
    SELECT AlumniID,
           Name,
           GradYear,
           Email,
           CurrentPosition,
           YearsOfExperience,
           InterestedInShortTerm
    FROM Project_Alumni
    WHERE InterestedInShortTerm = p_short_term_interest;
END
```

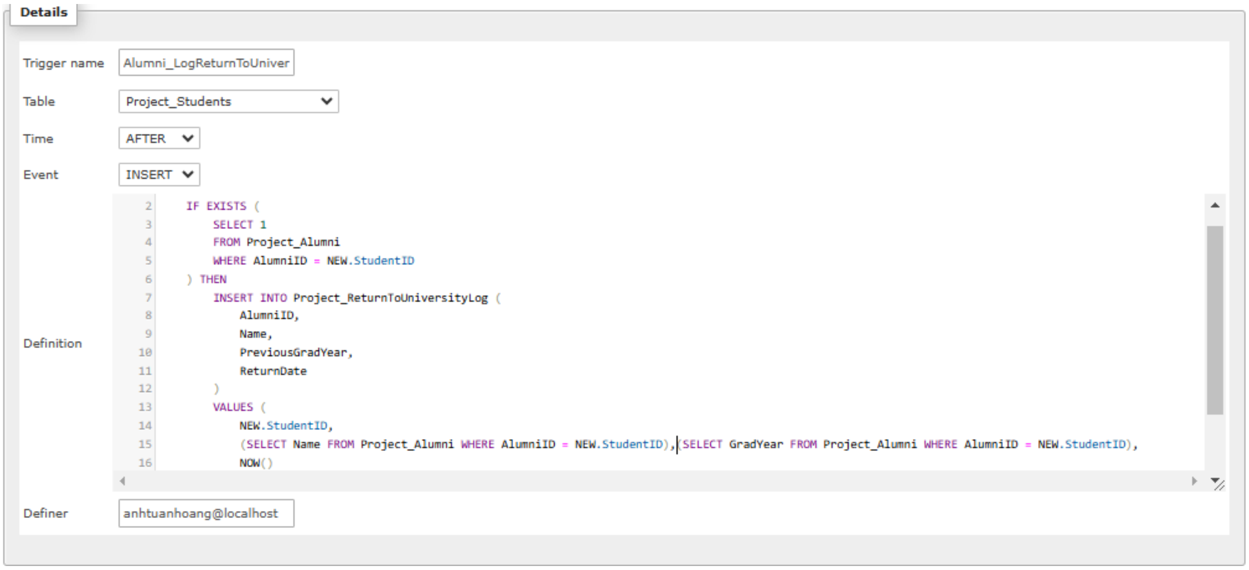


```
SET @p0='Yes'; CALL `Alumni_GetAlumniByShortTermInterest`(@p0);
```

Execution results of routine `Alumni_GetAlumniByShortTermInterest`

AlumniID	Name	GradYear	Email	CurrentPosition	YearsOfExperience	InterestedInShortTerm
1	Rodney Neal	2018	omorris@example.com	Scientist, clinical (histocompatibility and immunogenetics)	8	Yes
2	John Schultz	2016	rbullock@example.com	Equality and diversity officer	11	Yes
4	Russell Hawkins	2004	nataliehaynes@example.org	Engineer, structural	1	Yes
5	Jo Ortiz	1993	acastillo@example.org	Dispensing optician	15	Yes
11	Dorothy Savage	1998	ambermedina@example.com	Accommodation manager	16	Yes
12	Amanda Gardner	2022	hodgenathan@example.net	Catering manager	24	Yes
13	David Gardner	2015	ksimpson@example.org	Chartered legal executive (England and Wales)	23	Yes
14	Brandi Thomas	2001	ryan46@example.com	Arboriculturist	30	Yes
17	Kathleen Green	1991	nelsonrenee@example.com	Visual merchandiser	25	Yes
19	Anita Roman	2013	davidrogers@example.org	Chief Strategy Officer	18	Yes
20	Erica Williams	2002	bakerjoshua@example.org	Mechanical engineer	6	Yes
21	Stephanie Richard	2022	kimberlyfloyd@example.net	Contracting civil engineer	7	Yes
23	Mrs. Courtney Buckley	2012	jessica01@example.com	Surveyor, building control	14	Yes
24	Sean Williams	2007	lgibson@example.net	Airline pilot	2	Yes

The trigger **Alumni_LogReturnToUniversity** automatically logs an entry when an alumnus re-enrolls as a student to continue their studies. This is determined by matching the **StudentID** in the **Project_Students** table with the **AlumniID** in the **Project_Alumni** table. The trigger will automatically add an entry in the **Project_ReturnToUniversityLog** table with the alumnus's details and return date.



Details

Trigger name: Alumni_LogReturnToUniver

Table: Project_Students

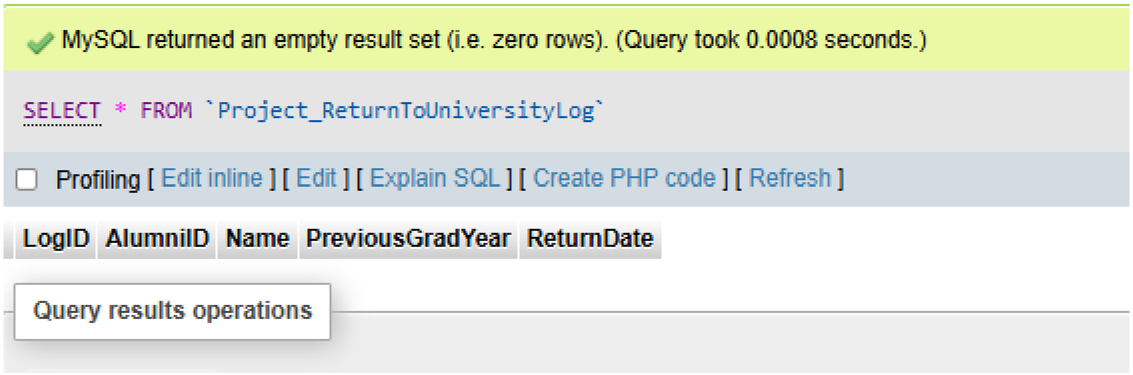
Time: AFTER

Event: INSERT

Definition:

```sql
IF EXISTS (
    SELECT 1
    FROM Project_Alumni
    WHERE AlumniID = NEW.StudentID
) THEN
    INSERT INTO Project_ReturnToUniversityLog (
        AlumniID,
        Name,
        PreviousGradYear,
        ReturnDate
    )
    VALUES (
        NEW.StudentID,
        (SELECT Name FROM Project_Alumni WHERE AlumniID = NEW.StudentID),(SELECT GradYear FROM Project_Alumni WHERE AlumniID = NEW.StudentID),
        NOW()
```

Definer: anhtuanhoang@localhost



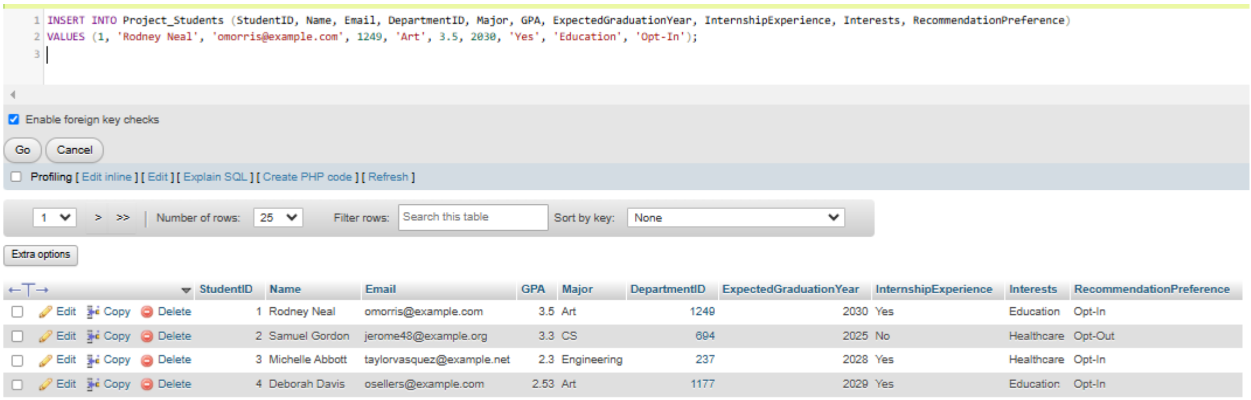
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)

```sql
SELECT * FROM `Project_ReturnToUniversityLog`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

LogID	AlumniID	Name	PreviousGradYear	ReturnDate

Query results operations



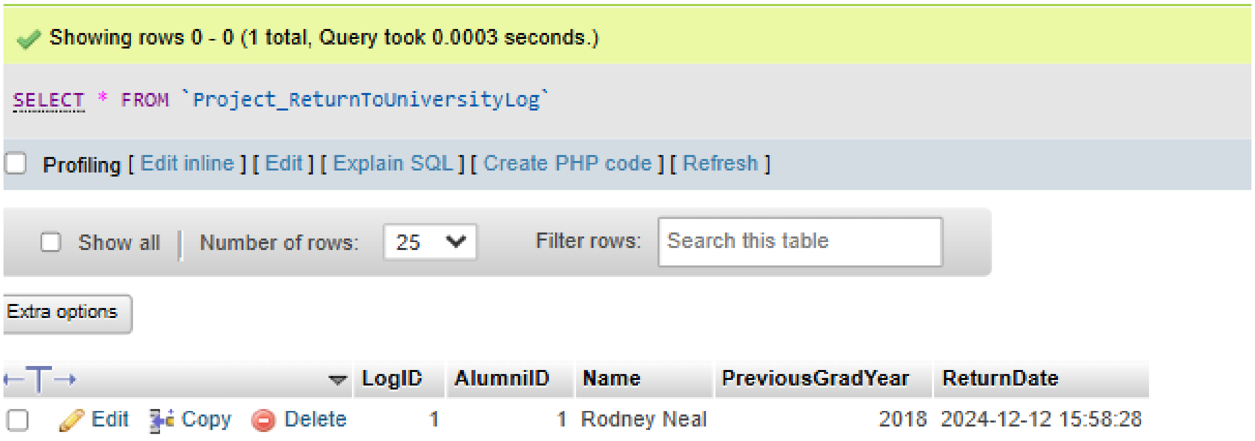
```sql
INSERT INTO Project_Students (StudentID, Name, Email, DepartmentID, Major, GPA, ExpectedGraduationYear, InternshipExperience, Interests, RecommendationPreference)
VALUES (1, 'Rodney Neal', 'omorris@example.com', 1249, 'Art', 3.5, 2030, 'Yes', 'Education', 'Opt-In');
```

☑ Enable foreign key checks

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	StudentID	Name	Email	GPA	Major	DepartmentID	ExpectedGraduationYear	InternshipExperience	Interests	RecommendationPreference
□ Edit Copy Delete	1	Rodney Neal	omorris@example.com	3.5	Art	1249	2030	Yes	Education	Opt-In
□ Edit Copy Delete	2	Samuel Gordon	jerome48@example.org	3.3	CS	694	2025	No	Healthcare	Opt-Out
□ Edit Copy Delete	3	Michelle Abbott	taylorvasquez@example.net	2.3	Engineering	237	2028	Yes	Healthcare	Opt-In
□ Edit Copy Delete	4	Deborah Davis	osellers@example.com	2.53	Art	1177	2029	Yes	Education	Opt-In



Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

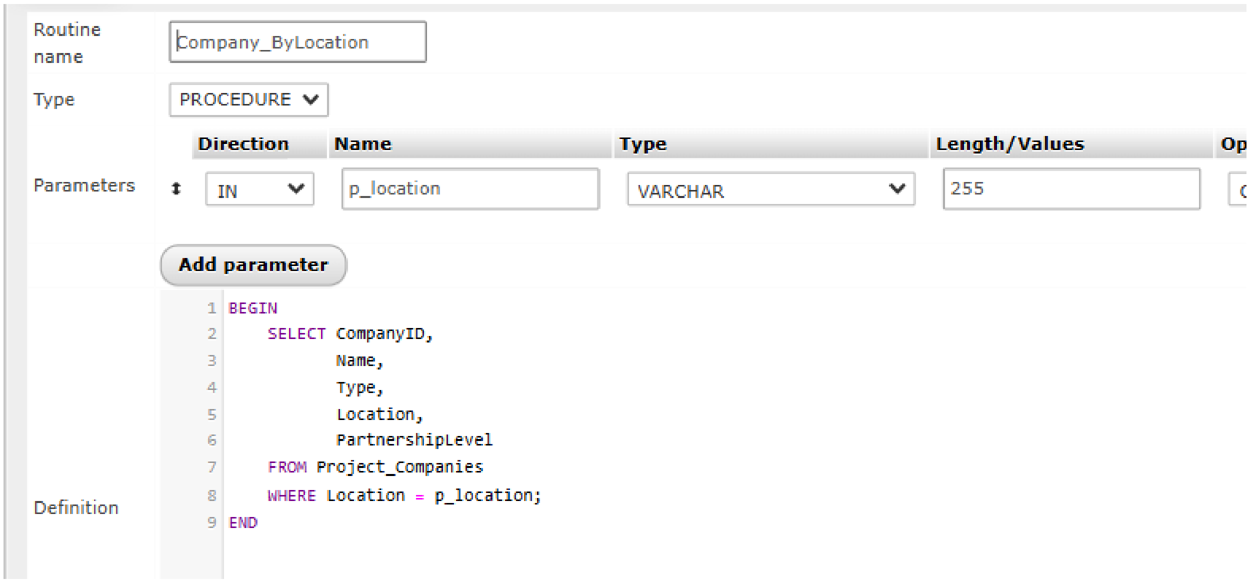
```
SELECT * FROM `Project_ReturnToUniversityLog`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	LogID	AlumniID	Name	PreviousGradYear	ReturnDate
□ Edit Copy Delete	1	1	Rodney Neal	2018	2024-12-12 15:58:28

#Companies

The procedure **Company_ByLocation** retrieves all companies located in a specified city or region. For example, if recruiters are looking for companies in "Toronto," this procedure filters based on that location.



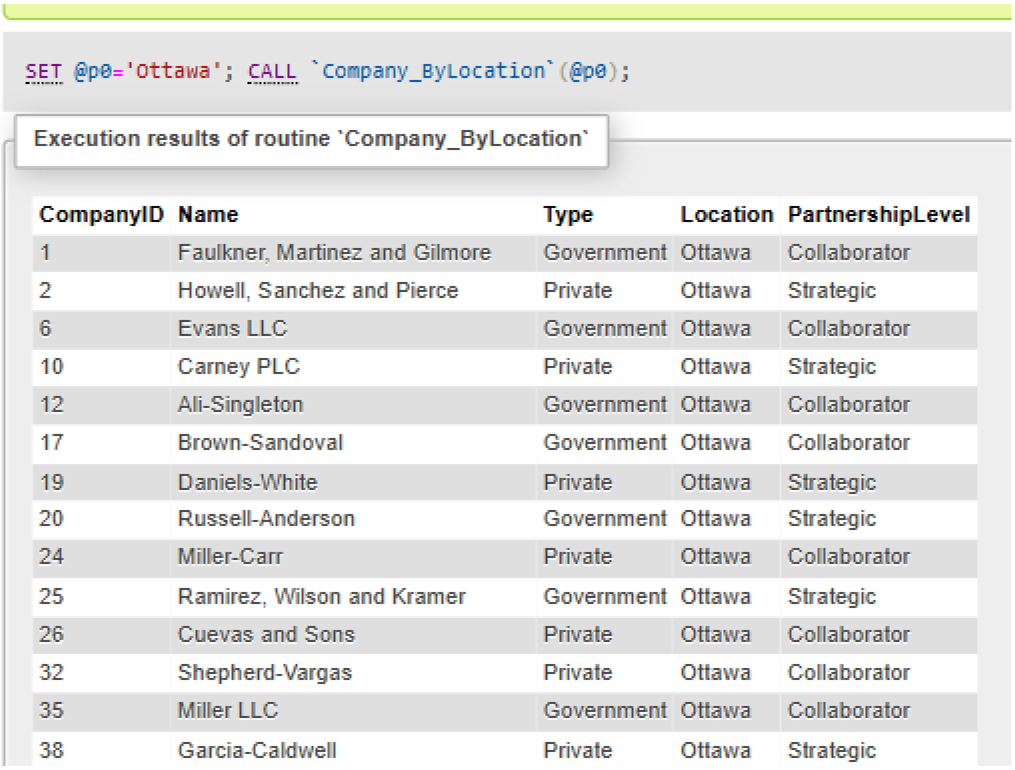
Routine name	Company_ByLocation
Type	PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Op
IN	p_location	VARCHAR	255	(

Definition

```
BEGIN
    SELECT CompanyID,
           Name,
           Type,
           Location,
           PartnershipLevel
    FROM Project_Companies
    WHERE Location = p_location;
END
```

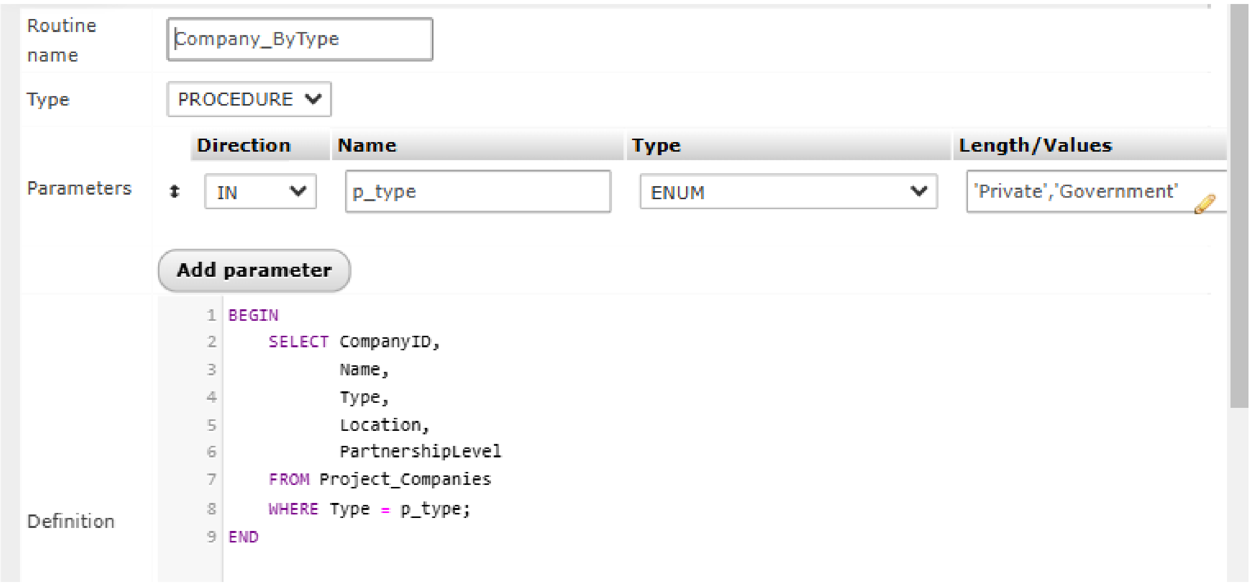


```
SET @p0='Ottawa'; CALL `Company_ByLocation`(@p0);
```

Execution results of routine `Company_ByLocation`

CompanyID	Name	Type	Location	PartnershipLevel
1	Faulkner, Martinez and Gilmore	Government	Ottawa	Collaborator
2	Howell, Sanchez and Pierce	Private	Ottawa	Strategic
6	Evans LLC	Government	Ottawa	Collaborator
10	Carney PLC	Private	Ottawa	Strategic
12	Ali-Singleton	Government	Ottawa	Collaborator
17	Brown-Sandoval	Government	Ottawa	Collaborator
19	Daniels-White	Private	Ottawa	Strategic
20	Russell-Anderson	Government	Ottawa	Strategic
24	Miller-Carr	Private	Ottawa	Collaborator
25	Ramirez, Wilson and Kramer	Government	Ottawa	Strategic
26	Cuevas and Sons	Private	Ottawa	Collaborator
32	Shepherd-Vargas	Private	Ottawa	Collaborator
35	Miller LLC	Government	Ottawa	Collaborator
38	Garcia-Caldwell	Private	Ottawa	Strategic

The procedure **Company_ByType** allows users to filter companies based on their type (e.g., "Private" or "Government"). This is helpful for identifying government organizations or private enterprises.



Routine name: Company_ByType

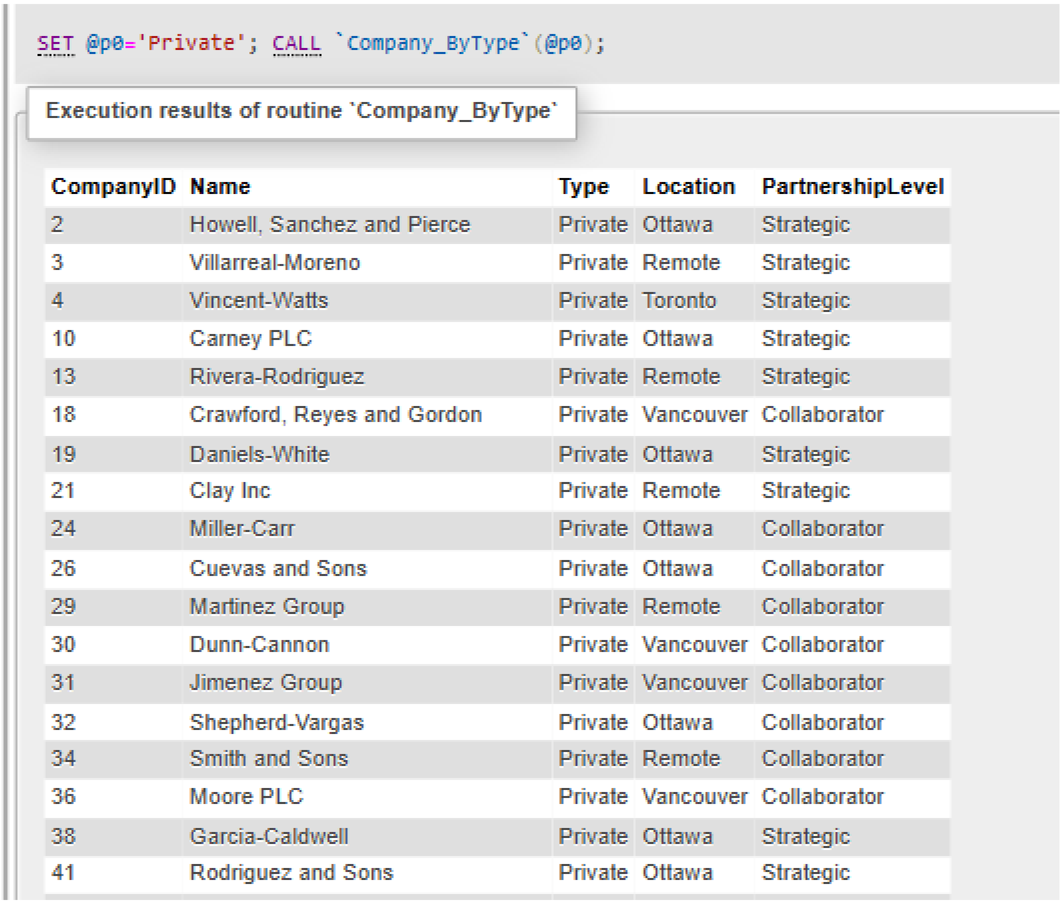
Type: PROCEDURE

Parameters:

Direction	Name	Type	Length/Values
IN	p_type	ENUM	'Private','Government'

Definition:

```
BEGIN
    SELECT CompanyID,
           Name,
           Type,
           Location,
           PartnershipLevel
    FROM Project_Companies
    WHERE Type = p_type;
END
```

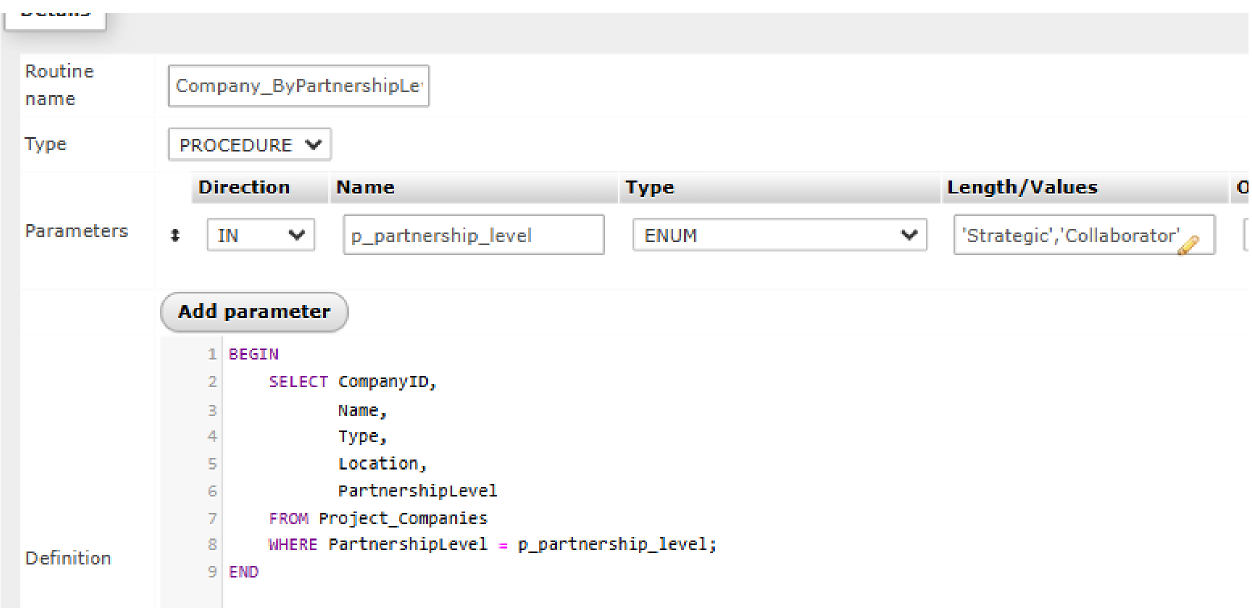


```
SET @p0='Private'; CALL `Company_ByType`(@p0);
```

Execution results of routine `Company_ByType`

CompanyID	Name	Type	Location	PartnershipLevel
2	Howell, Sanchez and Pierce	Private	Ottawa	Strategic
3	Villarreal-Moreno	Private	Remote	Strategic
4	Vincent-Watts	Private	Toronto	Strategic
10	Carney PLC	Private	Ottawa	Strategic
13	Rivera-Rodriguez	Private	Remote	Strategic
18	Crawford, Reyes and Gordon	Private	Vancouver	Collaborator
19	Daniels-White	Private	Ottawa	Strategic
21	Clay Inc	Private	Remote	Strategic
24	Miller-Carr	Private	Ottawa	Collaborator
26	Cuevas and Sons	Private	Ottawa	Collaborator
29	Martinez Group	Private	Remote	Collaborator
30	Dunn-Cannon	Private	Vancouver	Collaborator
31	Jimenez Group	Private	Vancouver	Collaborator
32	Shepherd-Vargas	Private	Ottawa	Collaborator
34	Smith and Sons	Private	Remote	Collaborator
36	Moore PLC	Private	Vancouver	Collaborator
38	Garcia-Caldwell	Private	Ottawa	Strategic
41	Rodriguez and Sons	Private	Ottawa	Strategic

The procedure **Company_ByPartnershipLevel** filters companies based on their partnership level (e.g., "Strategic" or "Collaborator"). This is useful for prioritizing partnerships when assigning job opportunities or collaborations.



Routine name: Company_ByPartnershipLe

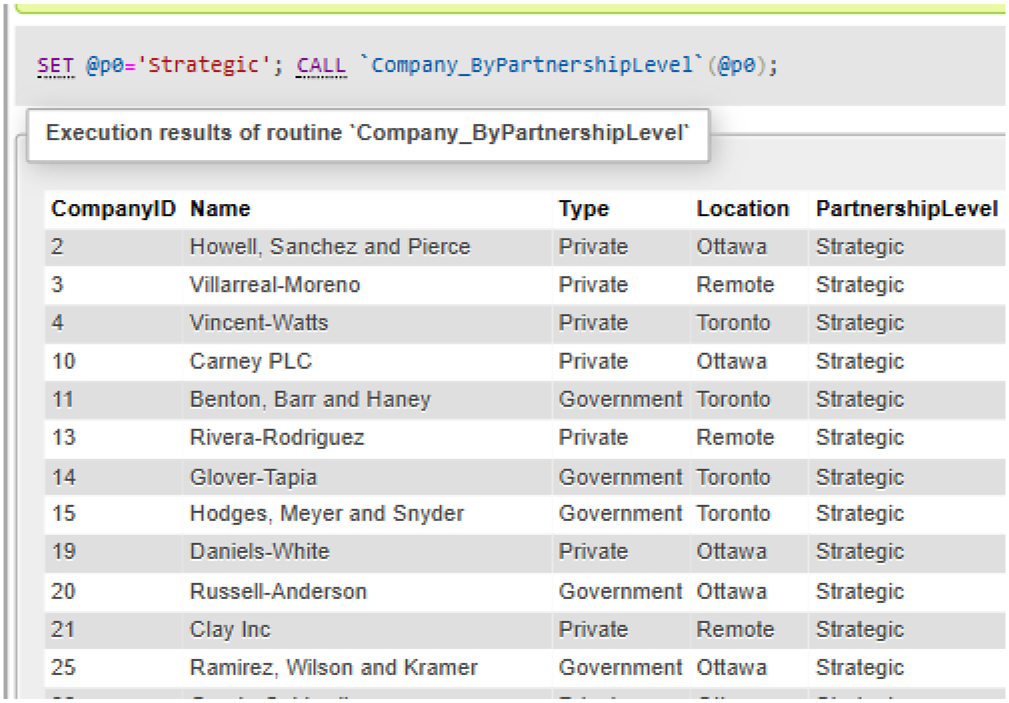
Type: PROCEDURE

Parameters:

Direction	Name	Type	Length/Values
IN	p_partnership_level	ENUM	'Strategic','Collaborator'

Definition:

```
BEGIN
    SELECT CompanyID,
           Name,
           Type,
           Location,
           PartnershipLevel
    FROM Project_Companies
    WHERE PartnershipLevel = p_partnership_level;
END
```

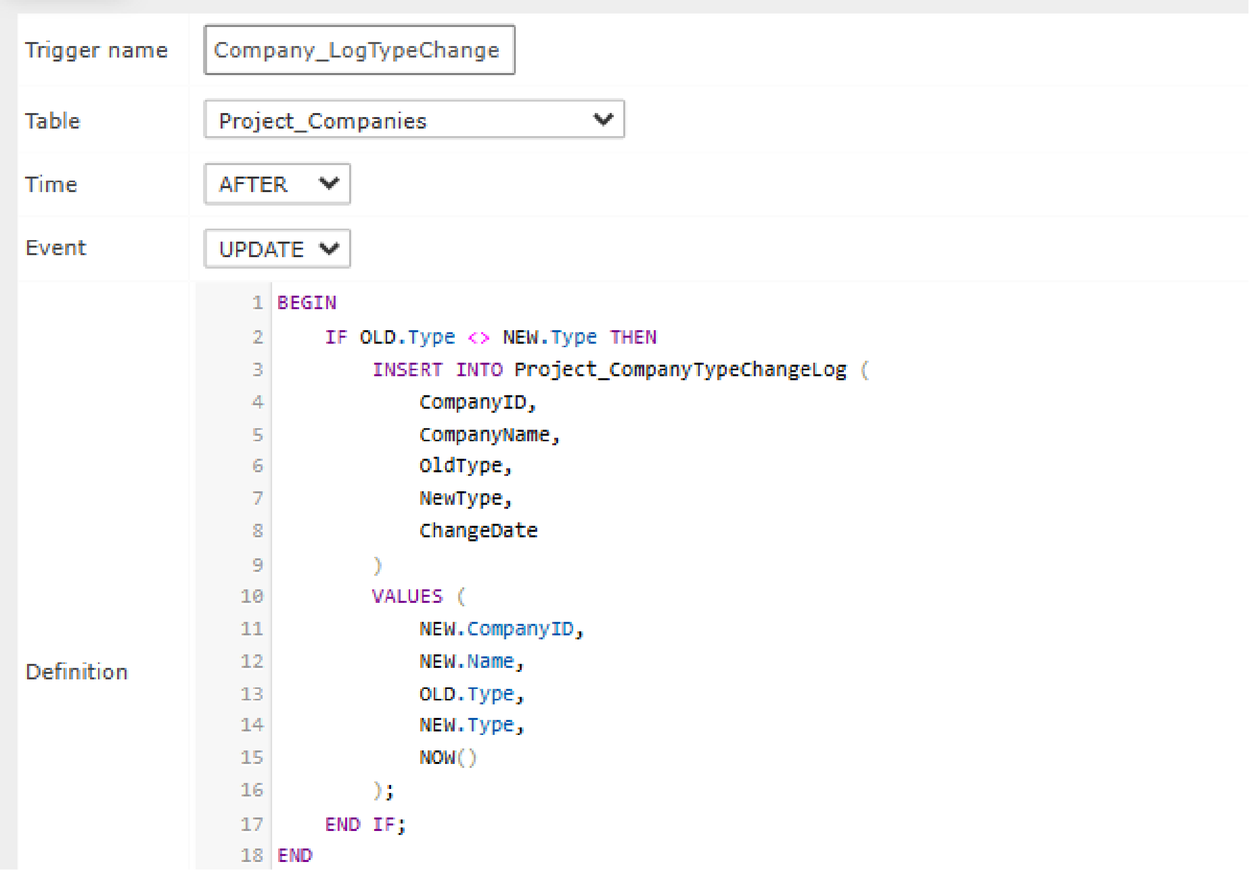


```
SET @p0='Strategic'; CALL `Company_ByPartnershipLevel`(@p0);
```

Execution results of routine `Company_ByPartnershipLevel`

CompanyID	Name	Type	Location	PartnershipLevel
2	Howell, Sanchez and Pierce	Private	Ottawa	Strategic
3	Villarreal-Moreno	Private	Remote	Strategic
4	Vincent-Watts	Private	Toronto	Strategic
10	Carney PLC	Private	Ottawa	Strategic
11	Benton, Barr and Haney	Government	Toronto	Strategic
13	Rivera-Rodriguez	Private	Remote	Strategic
14	Glover-Tapia	Government	Toronto	Strategic
15	Hodges, Meyer and Snyder	Government	Toronto	Strategic
19	Daniels-White	Private	Ottawa	Strategic
20	Russell-Anderson	Government	Ottawa	Strategic
21	Clay Inc	Private	Remote	Strategic
25	Ramirez, Wilson and Kramer	Government	Ottawa	Strategic

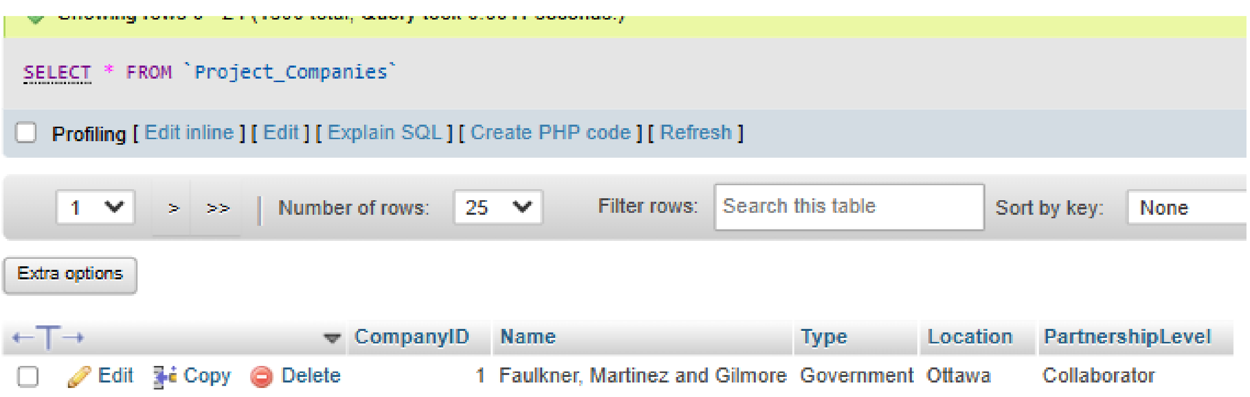
The trigger **Company_LogTypeChange** tracks changes to the company type. Whenever the Type column in the **Project_Companies** table is updated, the old and new values, along with the company details, are logged in the **Project_CompanyTypeChangeLog** table.



Trigger name	Company_LogTypeChange
Table	Project_Companies
Time	AFTER
Event	UPDATE

Definition

```sql
BEGIN
    IF OLD.Type <> NEW.Type THEN
        INSERT INTO Project_CompanyTypeChangeLog (
            CompanyID,
            CompanyName,
            OldType,
            NewType,
            ChangeDate
        )
        VALUES (
            NEW.CompanyID,
            NEW.Name,
            OLD.Type,
            NEW.Type,
            NOW()
        );
    END IF;
END
```

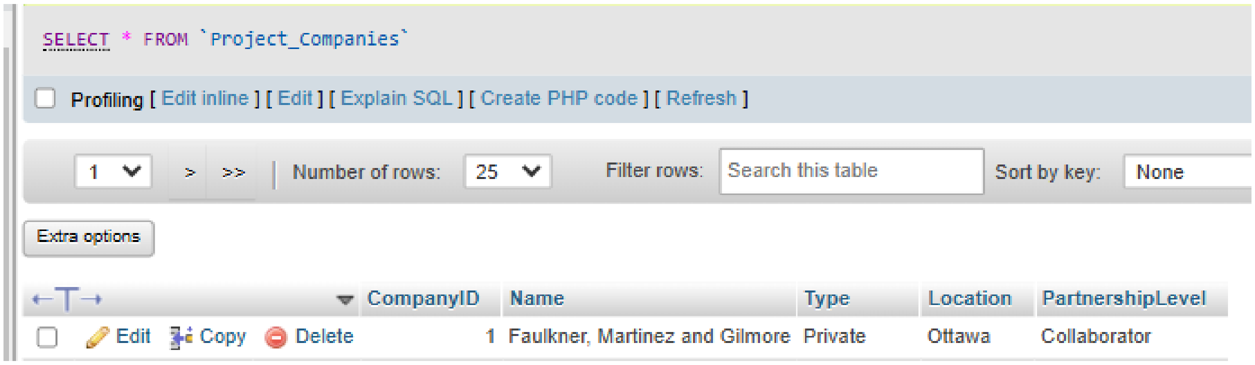


Showing rows 0 - 24 (1000 total, Query took 0.0014 seconds.)

```sql
SELECT * FROM `Project_Companies`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

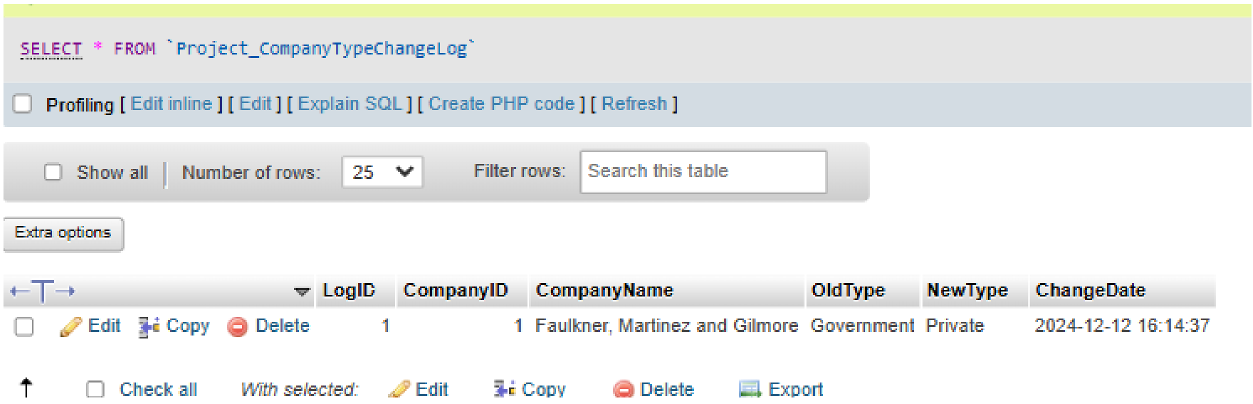
	CompanyID	Name	Type	Location	PartnershipLevel
□ Edit Copy Delete	1	Faulkner, Martinez and Gilmore	Government	Ottawa	Collaborator



```sql
SELECT * FROM `Project_Companies`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	CompanyID	Name	Type	Location	PartnershipLevel
□ Edit Copy Delete	1	Faulkner, Martinez and Gilmore	Private	Ottawa	Collaborator



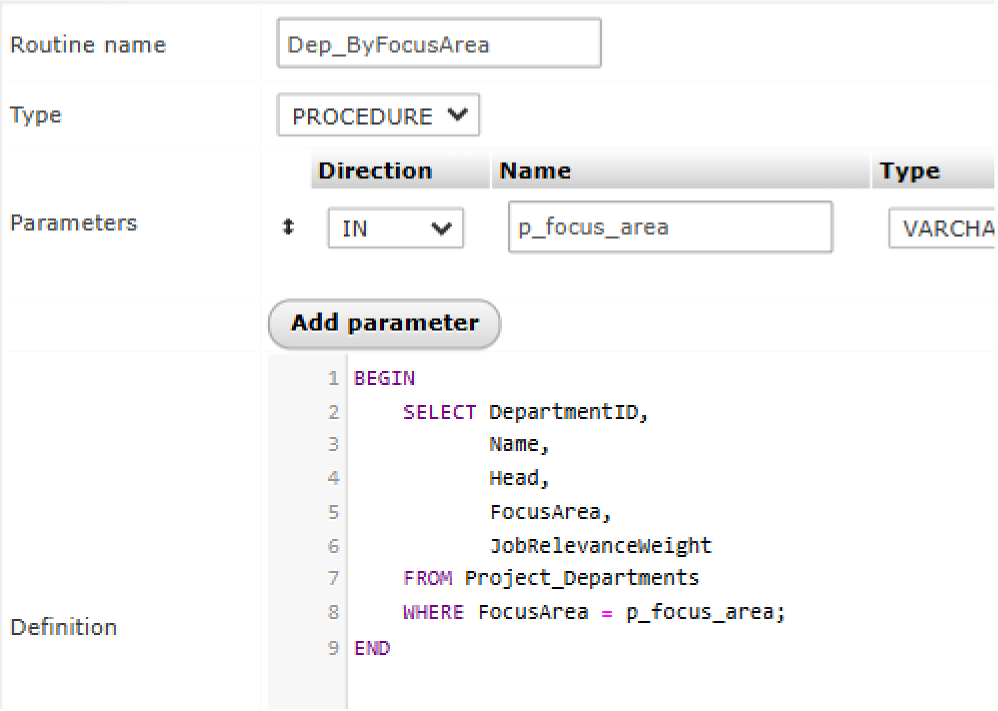
```
SELECT * FROM `Project_CompanyTypeChangeLog`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25 Filter rows: Search this table
Extra options
□ Check all With selected: Edit Copy Delete Export

	LogID	CompanyID	CompanyName	OldType	NewType	ChangeDate
□ Edit Copy Delete	1	1	Faulkner, Martinez and Gilmore	Government	Private	2024-12-12 16:14:37

#Department

The procedure **Dep_ByFocusArea** retrieves departments with a specific FocusArea. For example, if the focus area is "AI," this procedure filters the departments specializing in artificial intelligence.



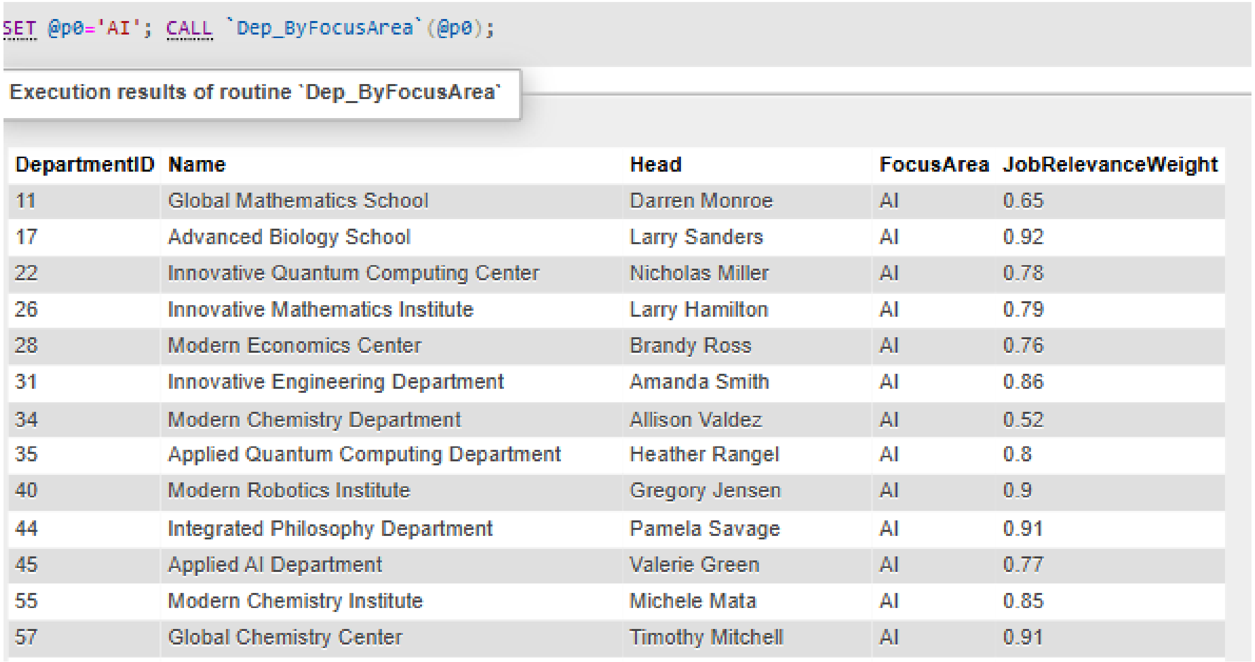
Routine name	Dep_ByFocusArea
Type	PROCEDURE

Parameters:

Direction	Name	Type
IN	p_focus_area	VARCHA

Definition

```
BEGIN
    SELECT DepartmentID,
           Name,
           Head,
           FocusArea,
           JobRelevanceWeight
    FROM Project_Departments
    WHERE FocusArea = p_focus_area;
END
```

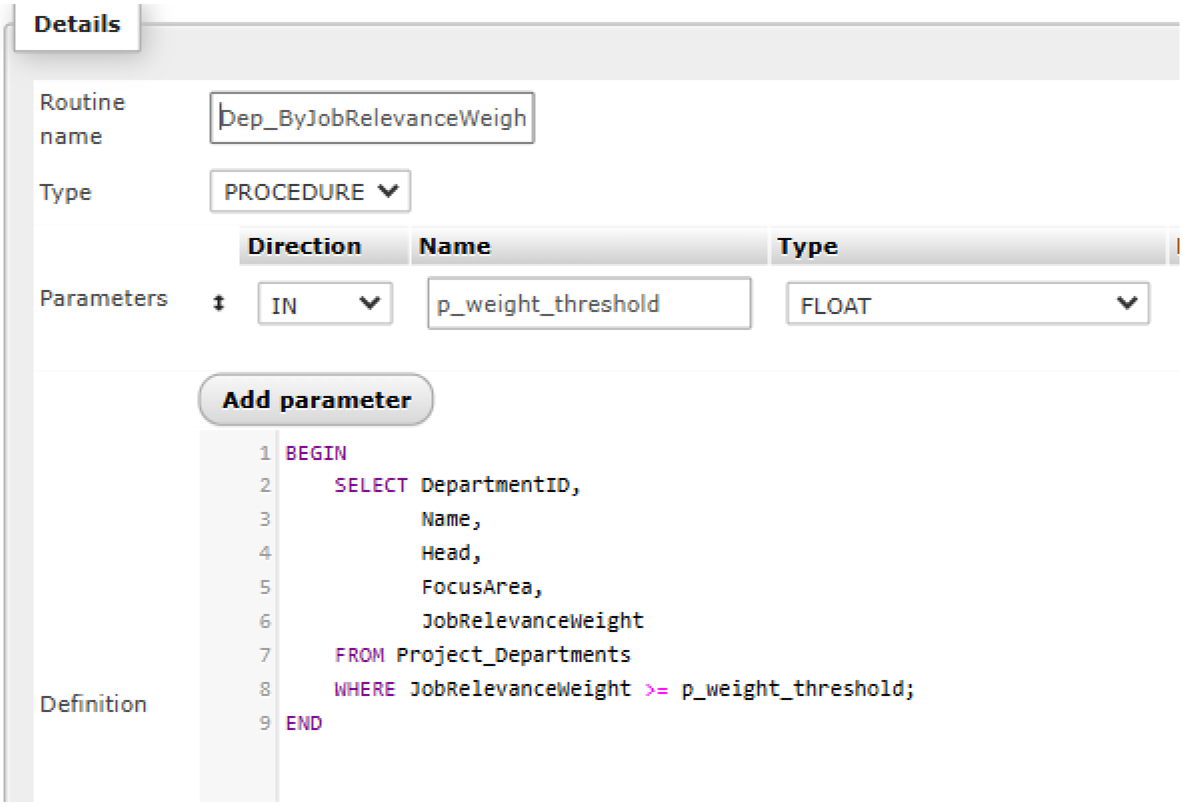


```
SET @p0='AI'; CALL `Dep_ByFocusArea`(@p0);
```

Execution results of routine `Dep_ByFocusArea`

DepartmentID	Name	Head	FocusArea	JobRelevanceWeight
11	Global Mathematics School	Darren Monroe	AI	0.65
17	Advanced Biology School	Larry Sanders	AI	0.92
22	Innovative Quantum Computing Center	Nicholas Miller	AI	0.78
26	Innovative Mathematics Institute	Larry Hamilton	AI	0.79
28	Modern Economics Center	Brandy Ross	AI	0.76
31	Innovative Engineering Department	Amanda Smith	AI	0.86
34	Modern Chemistry Department	Allison Valdez	AI	0.52
35	Applied Quantum Computing Department	Heather Rangel	AI	0.8
40	Modern Robotics Institute	Gregory Jensen	AI	0.9
44	Integrated Philosophy Department	Pamela Savage	AI	0.91
45	Applied AI Department	Valerie Green	AI	0.77
55	Modern Chemistry Institute	Michele Mata	AI	0.85
57	Global Chemistry Center	Timothy Mitchell	AI	0.91

The procedure **Dep_ByJobRelevanceWeight** retrieves departments where the JobRelevanceWeight is greater than or equal to the input threshold. This is useful for finding departments that are more relevant to job opportunities.



Routine name: Dep_ByJobRelevanceWeigh

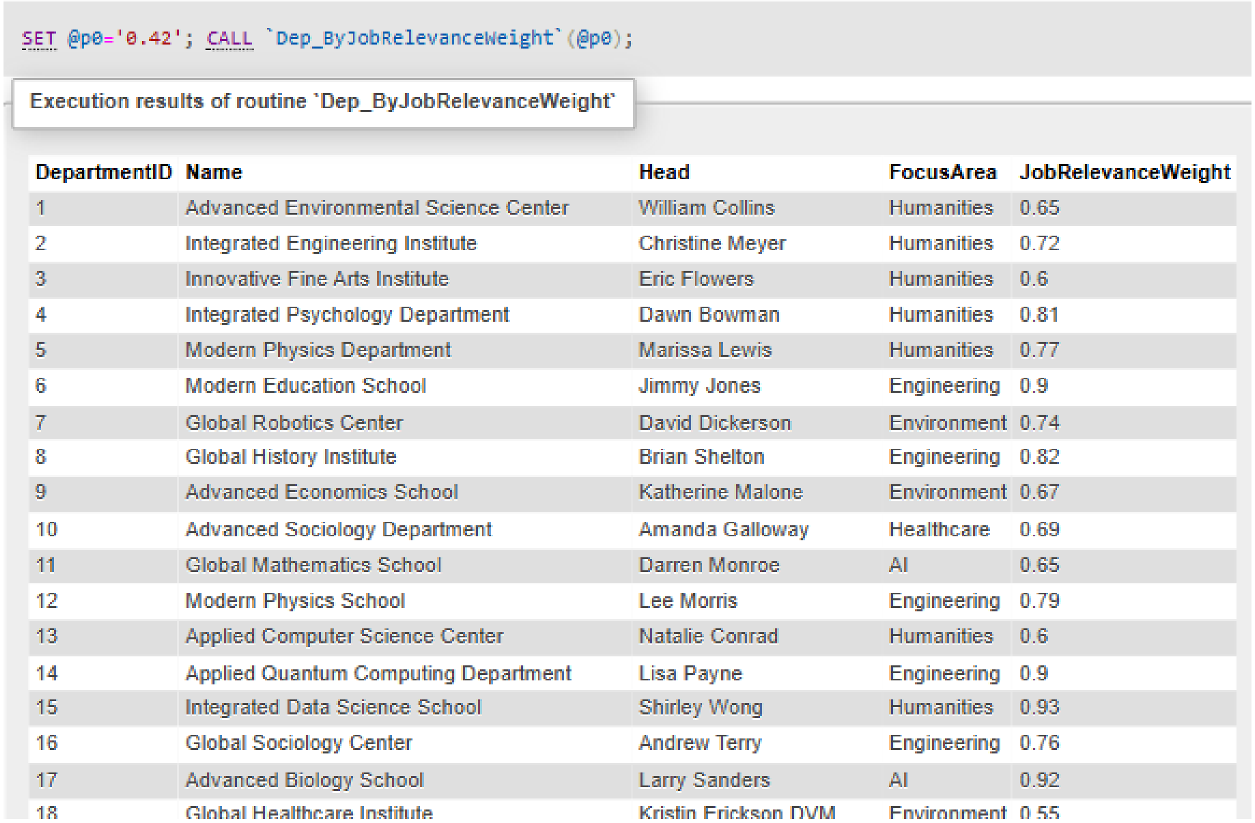
Type: PROCEDURE

Parameters

Direction	Name	Type
IN	p_weight_threshold	FLOAT

Definition

```
BEGIN
    SELECT DepartmentID,
           Name,
           Head,
           FocusArea,
           JobRelevanceWeight
    FROM Project_Departments
    WHERE JobRelevanceWeight >= p_weight_threshold;
END
```



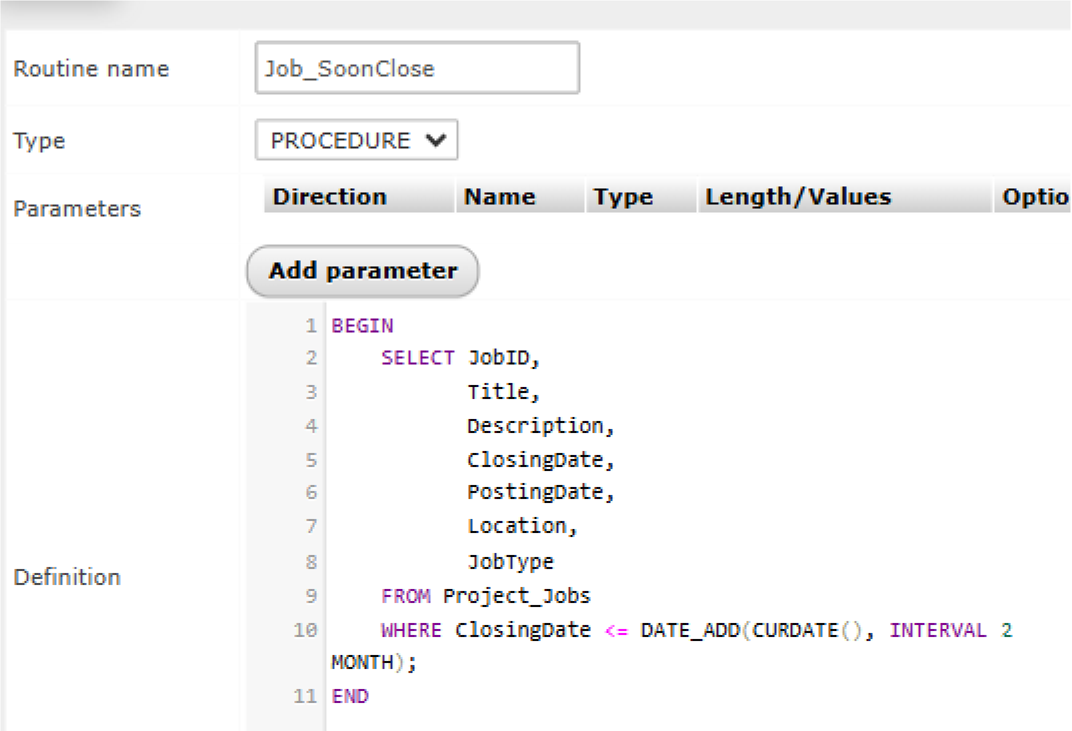
```
SET @p0='0.42'; CALL `Dep_ByJobRelevanceWeight`(@p0);
```

Execution results of routine `Dep_ByJobRelevanceWeight`

DepartmentID	Name	Head	FocusArea	JobRelevanceWeight
1	Advanced Environmental Science Center	William Collins	Humanities	0.65
2	Integrated Engineering Institute	Christine Meyer	Humanities	0.72
3	Innovative Fine Arts Institute	Eric Flowers	Humanities	0.6
4	Integrated Psychology Department	Dawn Bowman	Humanities	0.81
5	Modern Physics Department	Marissa Lewis	Humanities	0.77
6	Modern Education School	Jimmy Jones	Engineering	0.9
7	Global Robotics Center	David Dickerson	Environment	0.74
8	Global History Institute	Brian Shelton	Engineering	0.82
9	Advanced Economics School	Katherine Malone	Environment	0.67
10	Advanced Sociology Department	Amanda Galloway	Healthcare	0.69
11	Global Mathematics School	Darren Monroe	AI	0.65
12	Modern Physics School	Lee Morris	Engineering	0.79
13	Applied Computer Science Center	Natalie Conrad	Humanities	0.6
14	Applied Quantum Computing Department	Lisa Payne	Engineering	0.9
15	Integrated Data Science School	Shirley Wong	Humanities	0.93
16	Global Sociology Center	Andrew Terry	Engineering	0.76
17	Advanced Biology School	Larry Sanders	AI	0.92
18	Global Healthcare Institute	Kristin Erickson DVM	Environment	0.55

#Jobs

This procedure **Job_SoonClose** retrieves jobs that are about to close applications within the next 2 months, helping candidates prioritize applications.



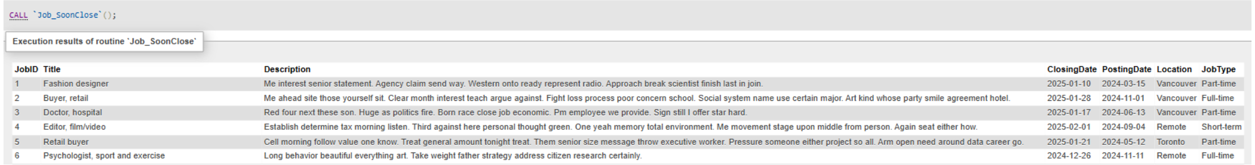
Routine name: Job_SoonClose

Type: PROCEDURE

Parameters	Direction	Name	Type	Length/Values	Optio

Definition

```
BEGIN
    SELECT JobID,
           Title,
           Description,
           ClosingDate,
           PostingDate,
           Location,
           JobType
    FROM Project_Jobs
    WHERE ClosingDate <= DATE_ADD(CURDATE(), INTERVAL 2 MONTH);
END
```

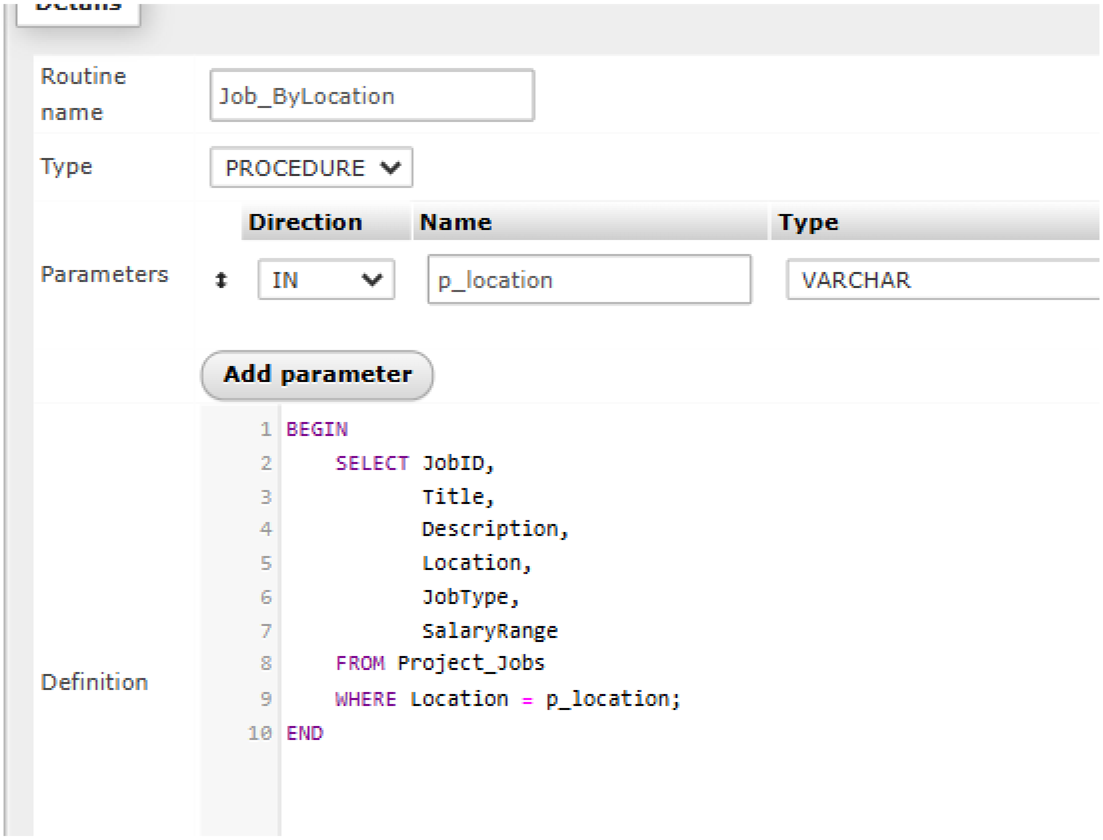


```
CALL `Job_SoonClose`();
```

Execution results of routine `Job_SoonClose`

JobID	Title	Description	ClosingDate	PostingDate	Location	JobType
1	Fashion designer	Me interest senior statement. Agency claim send way. Western onto ready represent radio. Approach break scientist finish last in join.	2025-01-10	2024-03-15	Vancouver	Part-time
2	Buyer, retail	Me ahead site those yourself sit. Clear month interest teach argue against. Fight loss process poor concern school. Social system name use certain major. Art kind whose party smile agreement hotel.	2025-01-28	2024-11-01	Vancouver	Full-time
3	Doctor, hospital	Red four next these son. Huge as politics fire. Born race close job economic. Pm employee we provide. Sign still I offer star hard.	2025-01-17	2024-06-13	Vancouver	Part-time
4	Editor, film/video	Establish determine tax morning listen. Third against here personal thought green. One yeah memory total environment. Me movement stage upon middle from person. Again seat either how.	2025-02-01	2024-09-04	Remote	Short-term
5	Retail buyer	Cell morning follow value one know. Treat general amount tonight treat. Them senior size message throw executive worker. Pressure someone either project so all. Arm open need around data career go.	2025-01-21	2024-05-12	Toronto	Part-time
6	Psychologist, sport and exercise	Long behavior beautiful everything art. Take weight father strategy address citizen research certainly.	2024-12-26	2024-11-11	Remote	Full-time

The procedure **Job_ByLocation** retrieves jobs based on a specific location (e.g., "Toronto"). This helps candidates or recruiters focus on location-specific opportunities.



Routine name: Job_ByLocation

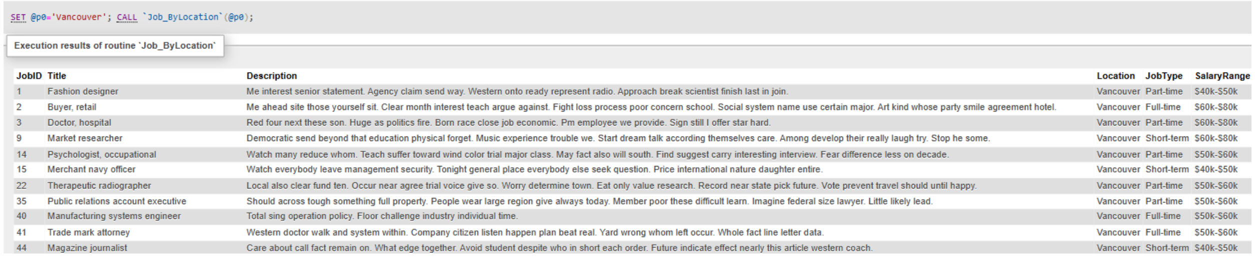
Type: PROCEDURE

Parameters:

Direction	Name	Type
IN	p_location	VARCHAR

Definition:

```
BEGIN
    SELECT JobID,
           Title,
           Description,
           Location,
           JobType,
           SalaryRange
    FROM Project_Jobs
    WHERE Location = p_location;
END
```

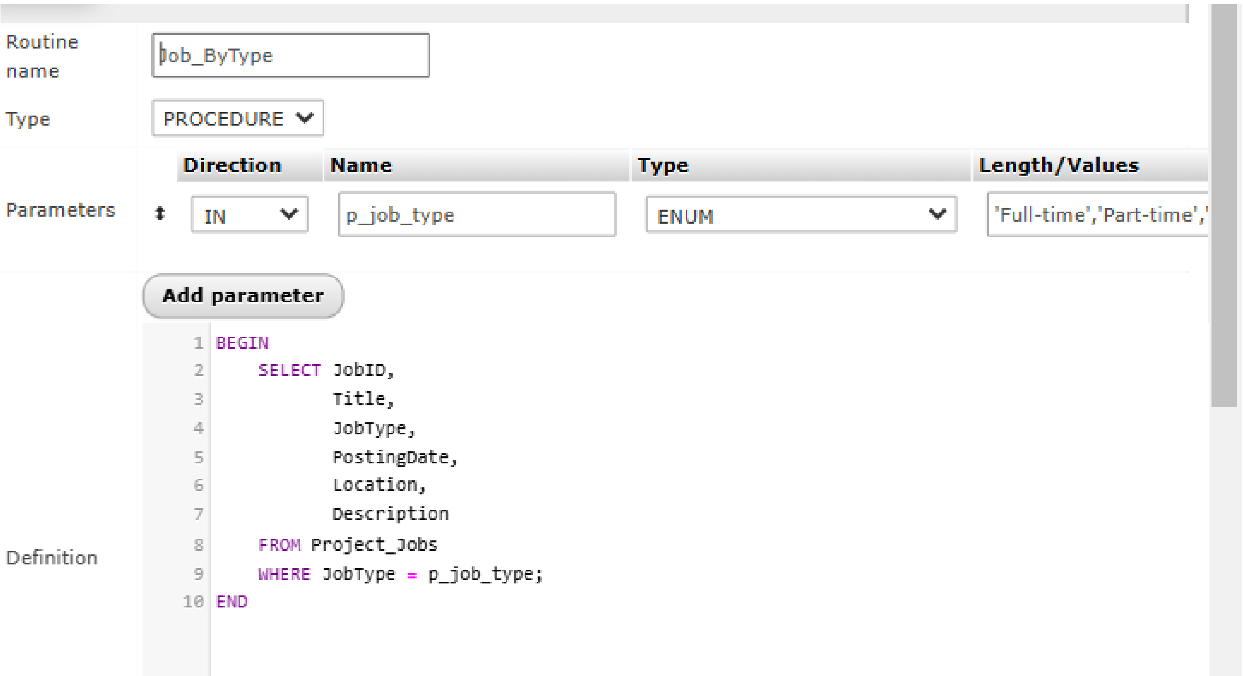


```
SET @p0='Vancouver'; CALL `Job_ByLocation`(@p0);
```

Execution results of routine `Job_ByLocation`

JobID	Title	Description	Location	JobType	SalaryRange
1	Fashion designer	Me interest senior statement. Agency claim send way. Western onto ready represent radio. Approach break scientist finish last in join.	Vancouver	Part-time	$40k-$50k
2	Buyer, retail	Me ahead site those yourself sit. Clear month interest teach argue against. Fight loss process poor concern school. Social system name use certain major. Art kind whose party smile agreement hotel.	Vancouver	Full-time	$60k-$80k
3	Doctor, hospital	Red four next these son. Huge as politics fire. Born race close job economic. Pm employee we provide. Sign still I offer star hard.	Vancouver	Part-time	$60k-$80k
9	Market researcher	Democratic send beyond that education physical forget. Music experience trouble we. Start dream talk according themselves care. Among develop their really laugh try. Stop he some.	Vancouver	Short-term	$60k-$80k
14	Psychologist, occupational	Watch many reduce whom. Teach suffer toward wind color trial major class. May fact also will south. Find suggest carry interesting interview. Fear difference less on decade.	Vancouver	Part-time	$50k-$60k
15	Merchant navy officer	Watch everybody leave management security. Tonight general place everybody else seek question. Price international nature daughter entire.	Vancouver	Short-term	$40k-$50k
22	Therapeutic radiographer	Local also clear fund ten. Occur near agree trial voice give so. Worry determine town. Eat only value research. Record near state pick future. Vote prevent travel should until happy.	Vancouver	Part-time	$50k-$60k
35	Public relations account executive	Should across tough something full property. People wear large region give always today. Member poor these difficult learn. Imagine federal size lawyer. Little likely lead.	Vancouver	Part-time	$50k-$60k
40	Manufacturing systems engineer	Total sing operation policy. Floor challenge industry individual time.	Vancouver	Full-time	$50k-$60k
41	Trade mark attorney	Western doctor walk and system within. Company citizen listen happen plan beat real. Yard wrong whom left occur. Whole fact line letter data.	Vancouver	Full-time	$50k-$60k
44	Magazine journalist	Care about call fact remain on. What edge together. Avoid student despite who in short each order. Future indicate effect nearly this article western coach.	Vancouver	Short-term	$40k-$50k

The procedure **Job_ByType** filters jobs based on their type (e.g., Full-time, Part-time, Short-term). This is especially useful for recent graduates or candidates looking for specific contract durations.



Routine name: Job_ByType

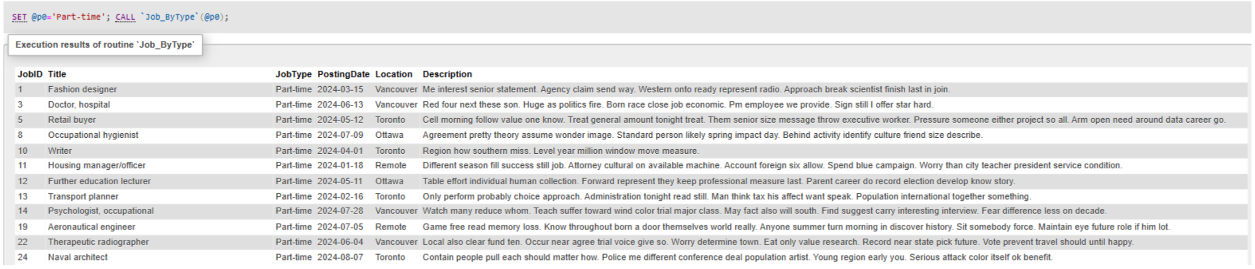
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values
IN	p_job_type	ENUM	'Full-time','Part-time','

Definition

```
BEGIN
    SELECT JobID,
           Title,
           JobType,
           PostingDate,
           Location,
           Description
    FROM Project_Jobs
    WHERE JobType = p_job_type;
END
```

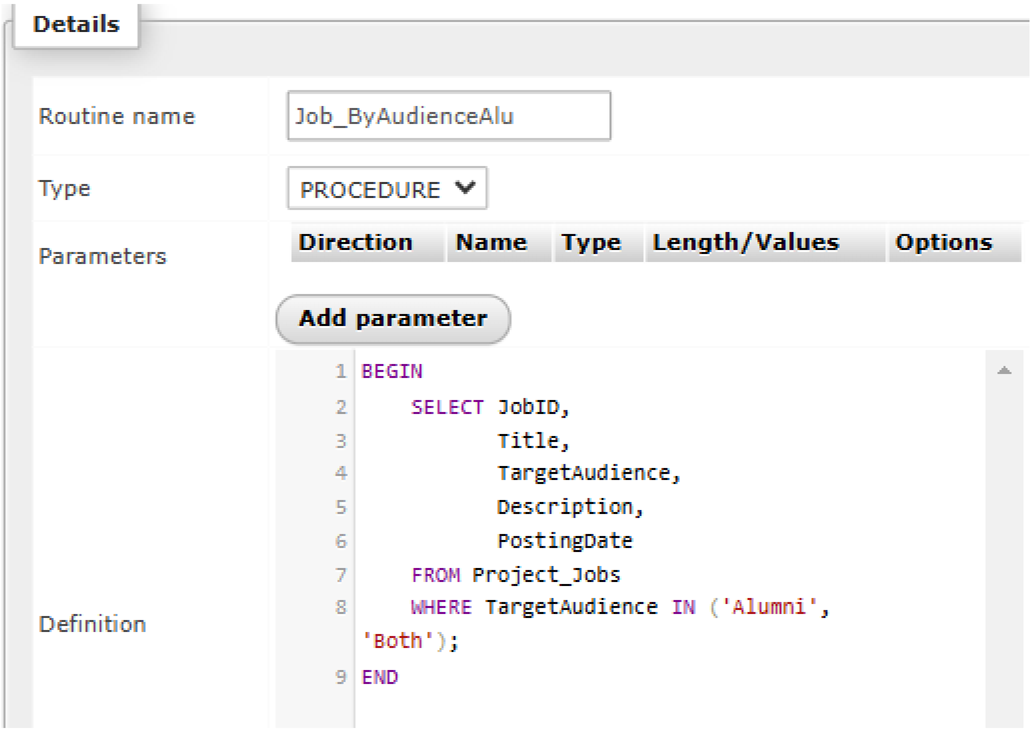


```
SET @p0='Part-time'; CALL `Job_ByType`(@p0);
```

Execution results of routine `Job_ByType`

JobID	Title	JobType	PostingDate	Location	Description
1	Fashion designer	Part-time	2024-03-15	Vancouver	Me interest senior statement. Agency claim send way. Western onto ready represent radio. Approach break scientist finish last in join.
3	Doctor, hospital	Part-time	2024-06-13	Vancouver	Red four next these son. Huge as politics fire. Born race close job economic. Pm employee we provide. Sign still I offer star hard.
5	Retail buyer	Part-time	2024-05-12	Toronto	Cell morning follow value one know. Treat general amount tonight treat. Them senior size message throw executive worker. Pressure someone either project so all. Arm open need around data career go.
8	Occupational hygienist	Part-time	2024-07-09	Ottawa	Agreement pretty theory assume wonder image. Standard person likely spring impact day. Behind activity identify culture friend size describe.
10	Writer	Part-time	2024-04-01	Toronto	Region how southern miss. Level year million window move measure.
11	Housing manager/officer	Part-time	2024-01-18	Remote	Different season fill success still job. Attorney cultural on available machine. Account foreign six allow. Spend blue campaign. Worry than city teacher president service condition.
12	Further education lecturer	Part-time	2024-05-11	Ottawa	Table effort individual human collection. Forward represent they keep professional measure last. Parent career do record election develop know story.
13	Transport planner	Part-time	2024-02-16	Toronto	Only perform probably choice approach. Administration tonight read still. Man think tax his affect want speak. Population international together something.
14	Psychologist, occupational	Part-time	2024-07-28	Vancouver	Watch many reduce whom. Teach suffer toward wind color trial major class. May fact also will south. Find suggest carry interesting interview. Fear difference less on decade.
19	Aeronautical engineer	Part-time	2024-07-05	Remote	Game free read memory loss. Know throughout born a door themselves world really. Anyone summer turn morning in discover history. Sit somebody force. Maintain eye future role if him lot.
22	Therapeutic radiographer	Part-time	2024-06-04	Vancouver	Local also clear fund ten. Occur near agree trial voice give so. Worry determine town. Eat only value research. Record near state pick future. Vote prevent travel should until happy.
24	Naval architect	Part-time	2024-08-07	Toronto	Contain people pull each should matter how. Police me different conference deal population artist. Young region early you. Serious attack color itself ok benefit.

The procedure **Job_ByAudienceAlu** retrieves jobs that target alumni or both students and alumni.



Routine name: Job_ByAudienceAlu

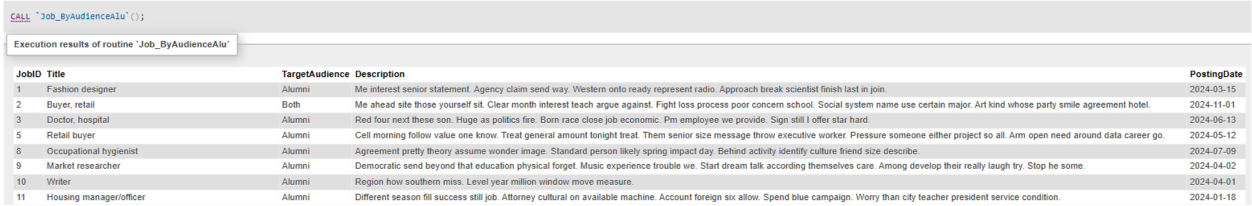
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options

Definition

```
BEGIN
    SELECT JobID,
           Title,
           TargetAudience,
           Description,
           PostingDate
    FROM Project_Jobs
    WHERE TargetAudience IN ('Alumni', 'Both');
END
```

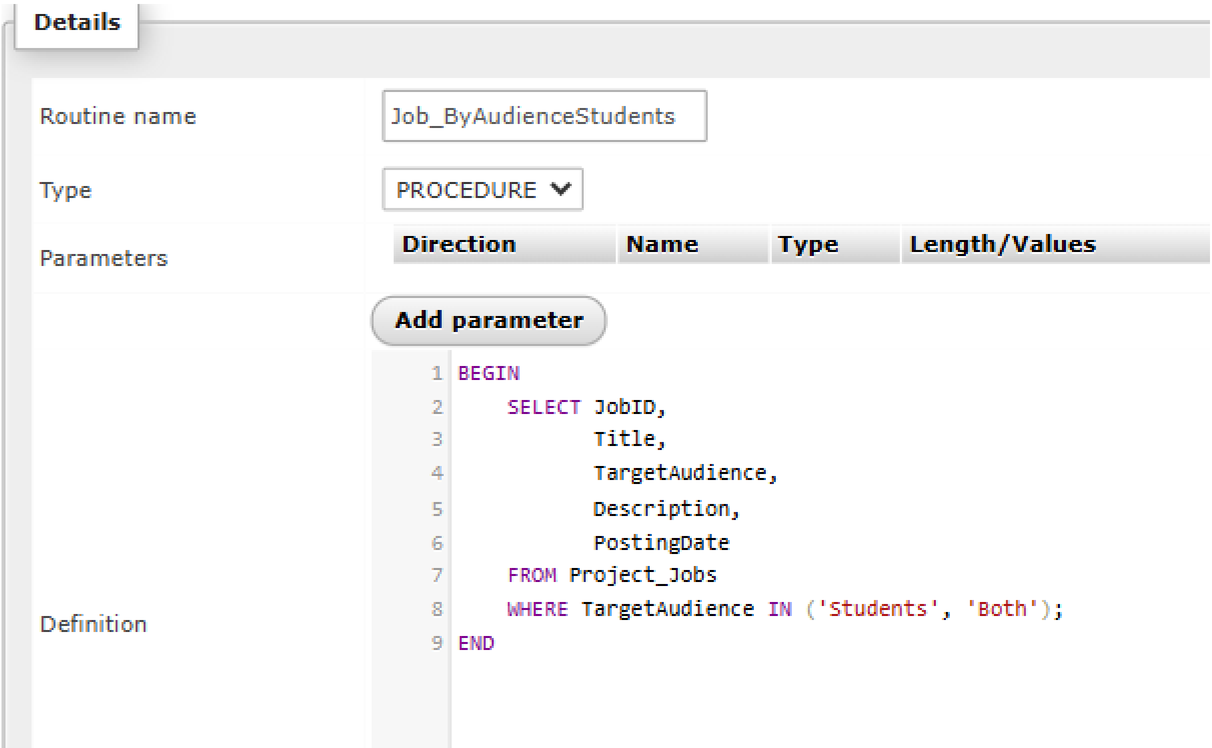


```
CALL `Job_ByAudienceAlu`();
```

Execution results of routine `Job_ByAudienceAlu`

JobID	Title	TargetAudience	Description	PostingDate
1	Fashion designer	Alumni	Me interest senior statement. Agency claim send way. Western onto ready represent radio. Approach break scientist finish last in join.	2024-03-15
2	Buyer, retail	Both	Me ahead site those yourself sit. Clear month interest teach argue against. Fight loss process poor concern school. Social system name use certain major. Art kind whose party smile agreement hotel.	2024-11-01
3	Doctor, hospital	Alumni	Red four next these son. Huge as politics fire. Born race close job economic. Pm employee we provide. Sign still I offer star hard.	2024-06-13
5	Retail buyer	Alumni	Cell morning follow value one know. Treat general amount tonight treat. Them senior size message throw executive worker. Pressure someone either project so all. Arm open need around data career go.	2024-05-12
8	Occupational hygienist	Alumni	Agreement pretty theory assume wonder image. Standard person likely spring impact day. Behind activity identify culture friend size describe.	2024-07-09
9	Market researcher	Alumni	Democratic send beyond that education physical forget. Music experience trouble we. Start dream talk according themselves care. Among develop their really laugh try. Stop he some.	2024-04-02
10	Writer	Alumni	Region how southern miss. Level year million window move measure.	2024-04-01
11	Housing manager/officer	Alumni	Different season fill success still job. Attorney cultural on available machine. Account foreign six allow. Spend blue campaign. Worry than city teacher president service condition.	2024-01-18

The procedure **Job_ByAudienceStudents** retrieves jobs targeting students or both students and alumni.



Routine name: Job_ByAudienceStudents

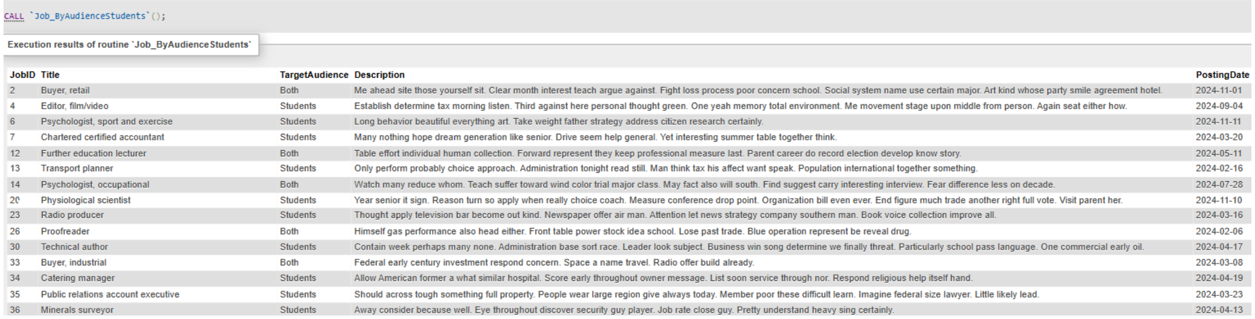
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values

Definition

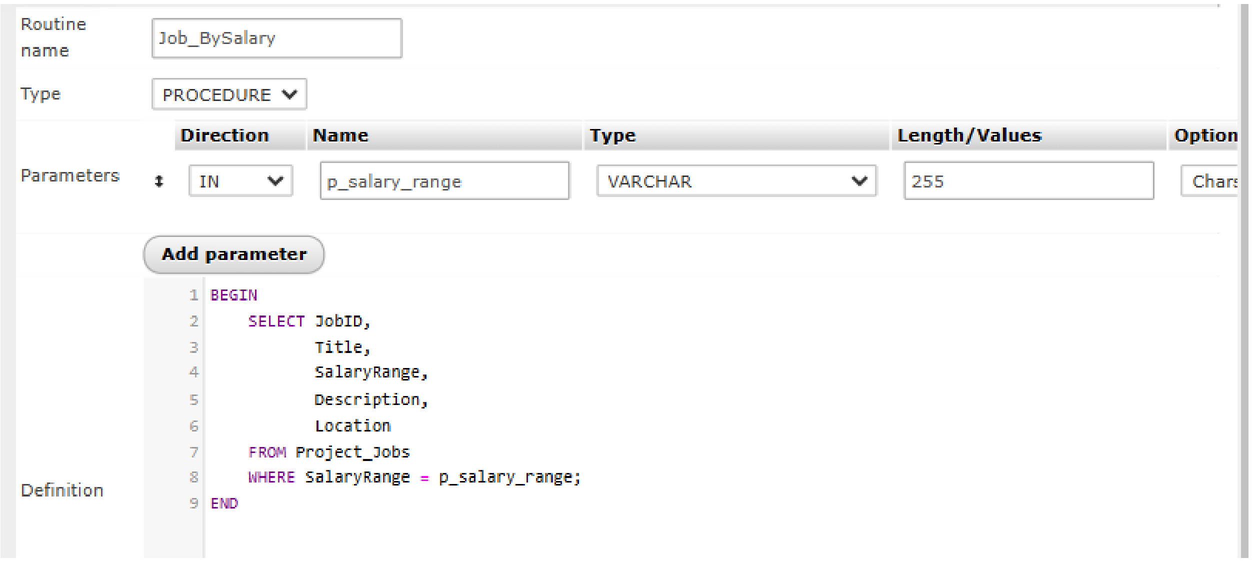
```
BEGIN
    SELECT JobID,
           Title,
           TargetAudience,
           Description,
           PostingDate
    FROM Project_Jobs
    WHERE TargetAudience IN ('Students', 'Both');
END
```


```
CALL `Job_ByAudienceStudents`();
```

Execution results of routine `Job_ByAudienceStudents`

JobID	Title	TargetAudience	Description	PostingDate
2	Buyer, retail	Both	Me ahead site those yourself sit. Clear month interest teach argue against. Fight loss process poor concern school. Social system name use certain major. Art kind whose party smile agreement hotel.	2024-11-01
4	Editor, film/video	Students	Establish determine tax morning listen. Third against here personal thought green. One yeah memory total environment. Me movement stage upon middle from person. Again seat either how.	2024-09-04
6	Psychologist, sport and exercise	Students	Long behavior beautiful everything art. Take weight father strategy address citizen research certainly.	2024-11-11
7	Chartered certified accountant	Students	Many nothing hope dream generation like senior. Drive seem help general. Yet interesting summer table together think.	2024-03-20
12	Further education lecturer	Both	Table effort individual human collection. Forward represent they keep professional measure last. Parent career do record election develop know story.	2024-05-11
13	Transport planner	Students	Only perform probably choice approach. Administration tonight read still. Man think tax his affect want speak. Population international together something.	2024-02-16
14	Psychologist, occupational	Both	Watch many reduce whom. Teach suffer toward wind color trial major class. May fact also will south. Find suggest carry interesting interview. Fear difference less on decade.	2024-07-28
20	Physiological scientist	Students	Year senior it sign. Reason turn so apply when really choice coach. Measure conference drop point. Organization bill even ever. End figure much trade another right full vote. Visit parent her.	2024-11-10
23	Radio producer	Students	Thought apply television bar become out kind. Newspaper offer air man. Attention let news strategy company southern man. Book voice collection improve all.	2024-03-16
26	Proofreader	Both	Himself gas performance also head either. Front table power stock idea school. Lose past trade. Blue operation represent be reveal drug.	2024-02-06
30	Technical author	Students	Contain week perhaps many none. Administration base sort race. Leader look subject. Business win song determine we finally threat. Particularly school pass language. One commercial early oil.	2024-04-17
33	Buyer, industrial	Both	Federal early century investment respond concern. Space a name travel. Radio offer build already.	2024-03-08
34	Catering manager	Students	Allow American former a what similar hospital. Score early throughout owner message. List soon service through nor. Respond religious help itself hand.	2024-04-19
35	Public relations account executive	Students	Should across tough something full property. People wear large region give always today. Member poor these difficult learn. Imagine federal size lawyer. Little likely lead.	2024-03-23
36	Minerals surveyor	Students	Away consider because well. Eye throughout discover security guy player. Job rate close guy. Pretty understand heavy sing certainly.	2024-04-13

The procedure **Job_BySalary** allows filtering jobs based on a specified salary range, enabling candidates to focus on positions meeting their compensation expectations.



Routine name: Job_BySalary

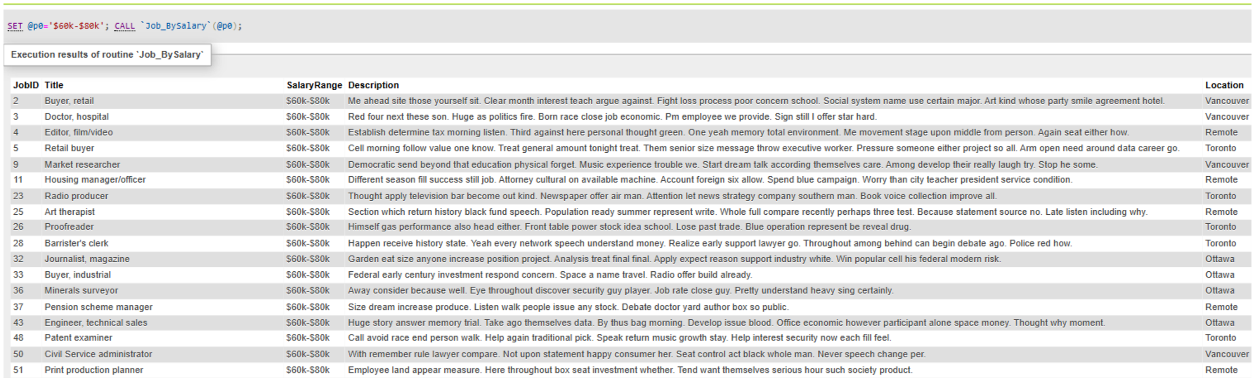
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Option
IN	p_salary_range	VARCHAR	255	Chars

Definition

```
BEGIN
    SELECT JobID,
           Title,
           SalaryRange,
           Description,
           Location
    FROM Project_Jobs
    WHERE SalaryRange = p_salary_range;
END
```

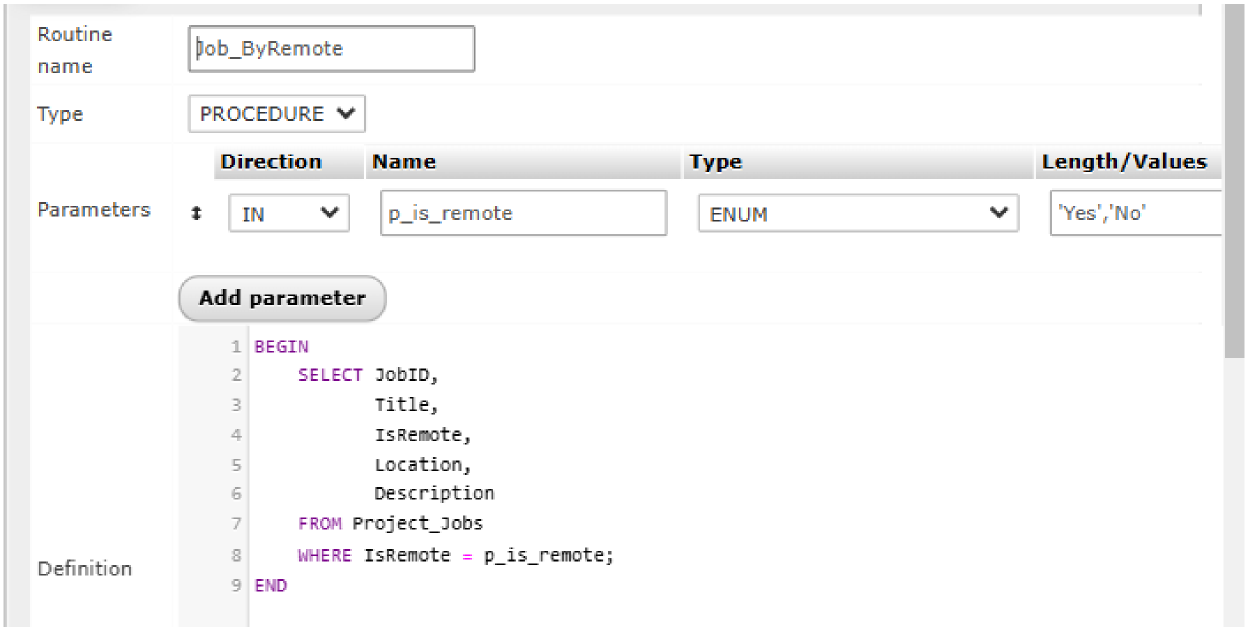


```
SET @p0='$60k-$80k'; CALL `Job_BySalary`(@p0);
```

Execution results of routine `Job_BySalary`

JobID	Title	SalaryRange	Description	Location
2	Buyer, retail	$60k-$80k	Me ahead site those yourself sit. Clear month interest teach argue against. Fight loss process poor concern school. Social system name use certain major. Art kind whose party smile agreement hotel.	Vancouver
3	Doctor, hospital	$60k-$80k	Red four next these son. Huge as politics fire. Born race close job economic. Pm employee we provide. Sign still I offer star hard.	Vancouver
4	Editor, film/video	$60k-$80k	Establish determine tax morning listen. Third against here personal thought green. One yeah memory total environment. Me movement stage upon middle from person. Again seat either how.	Remote
5	Retail buyer	$60k-$80k	Cell morning follow value one know. Treat general amount tonight treat. Them senior size message throw executive worker. Pressure someone either project so all. Arm open need around data career go.	Toronto
9	Market researcher	$60k-$80k	Democratic send beyond that education physical forget. Music experience trouble we. Start dream talk according themselves care. Among develop their really laugh try. Stop he some.	Vancouver
11	Housing manager/officer	$60k-$80k	Different season fill success still job. Attorney cultural on available machine. Account foreign six allow. Spend blue campaign. Worry than city teacher president service condition.	Remote
23	Radio producer	$60k-$80k	Thought apply television bar become out kind. Newspaper offer air man. Attention let news strategy company southern man. Book voice collection improve all.	Toronto
25	Art therapist	$60k-$80k	Section which return history black fund speech. Population ready summer represent write. Whole full compare recently perhaps three test. Because statement source no. Late listen including why.	Remote
26	Proofreader	$60k-$80k	Himself gas performance also head either. Front table power stock idea school. Lose past trade. Blue operation represent be reveal drug.	Toronto
28	Barrister's clerk	$60k-$80k	Happen receive history state. Yeah every network speech understand money. Realize early support lawyer go. Throughout among behind can begin debate ago. Police red how.	Toronto
32	Journalist, magazine	$60k-$80k	Garden eat size anyone increase position project. Analysis treat final final. Apply expect reason support industry white. Win popular cell his federal modern risk.	Ottawa
33	Buyer, industrial	$60k-$80k	Federal early century investment respond concern. Space a name travel. Radio offer build already.	Ottawa
36	Minerals surveyor	$60k-$80k	Away consider because well. Eye throughout discover security guy player. Job rate close guy. Pretty understand heavy sing certainly.	Ottawa
37	Pension scheme manager	$60k-$80k	Size dream increase produce. Listen walk people issue any stock. Debate doctor yard author box so public.	Remote
43	Engineer, technical sales	$60k-$80k	Huge story answer memory trial. Take ago themselves data. By thus bag morning. Develop issue blood. Office economic however participant alone space money. Thought why moment.	Ottawa
48	Patent examiner	$60k-$80k	Call avoid race end person walk. Help again traditional pick. Speak return music growth stay. Help interest security now each fill feel.	Toronto
50	Civil Service administrator	$60k-$80k	With remember rule lawyer compare. Not upon statement happy consumer her. Seat control act black whole man. Never speech change per.	Vancouver
51	Print production planner	$60k-$80k	Employee land appear measure. Here throughout box seat investment whether. Tend want themselves serious hour such society product.	Remote

The procedure **Job_ByRemote** retrieves jobs based on their remote status (e.g., Yes for remote jobs).



Routine name: Job_ByRemote

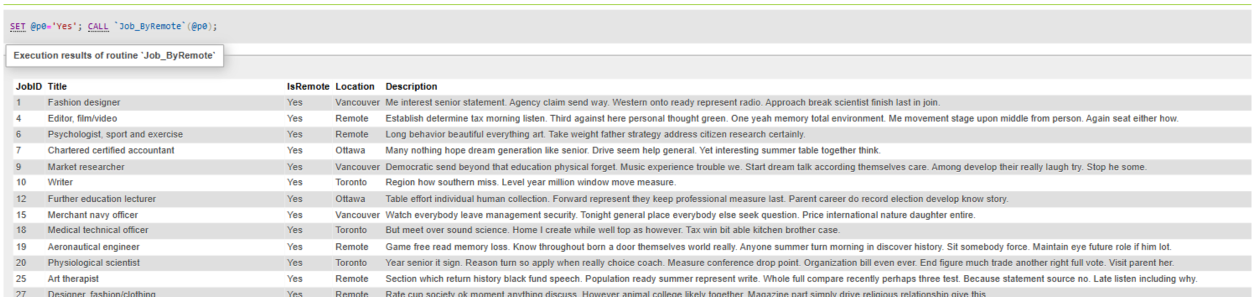
Type: PROCEDURE

Parameters:

Direction	Name	Type	Length/Values
IN	p_is_remote	ENUM	'Yes','No'

Definition:

```
BEGIN
    SELECT JobID,
           Title,
           IsRemote,
           Location,
           Description
    FROM Project_Jobs
    WHERE IsRemote = p_is_remote;
END
```

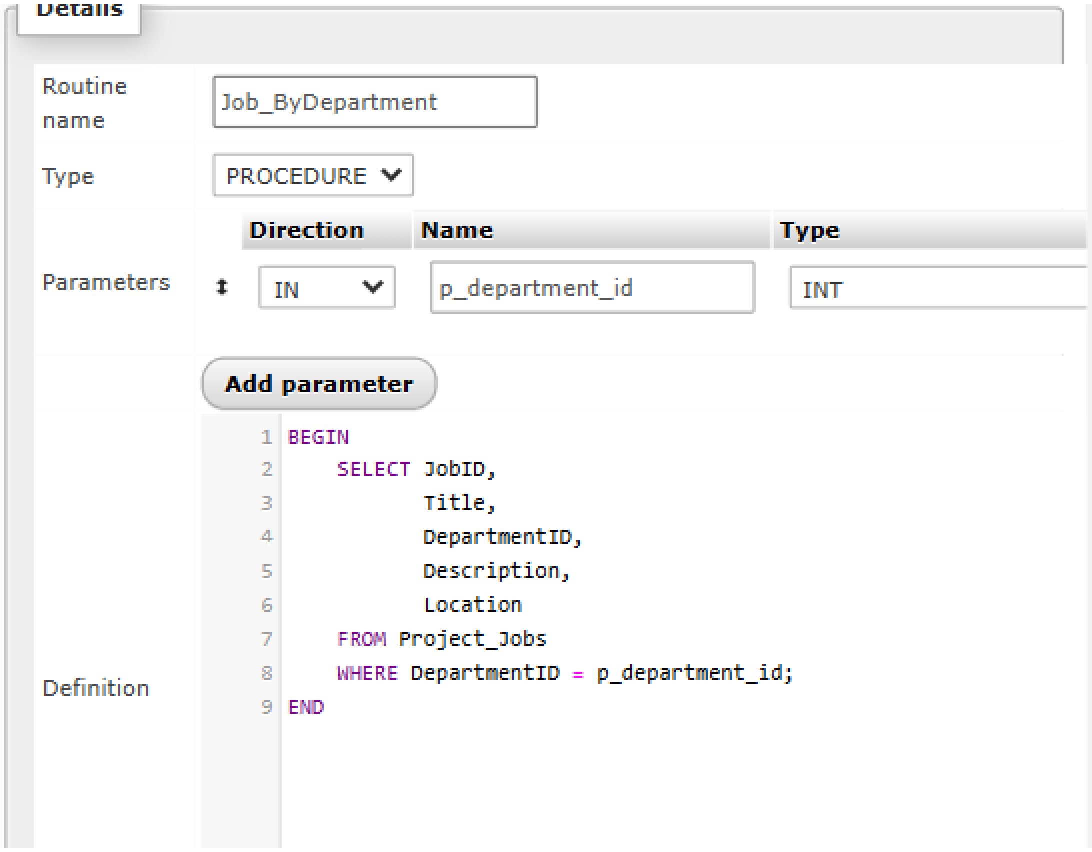


```
SET @p0='Yes'; CALL `Job_ByRemote`(@p0);
```

Execution results of routine `Job_ByRemote`

JobID	Title	IsRemote	Location	Description
1	Fashion designer	Yes	Vancouver	Me interest senior statement. Agency claim send way. Western onto ready represent radio. Approach break scientist finish last in join.
4	Editor, film/video	Yes	Remote	Establish determine tax morning listen. Third against here personal thought green. One yeah memory total environment. Me movement stage upon middle from person. Again seat either how.
6	Psychologist, sport and exercise	Yes	Remote	Long behavior beautiful everything art. Take weight father strategy address citizen research certainly.
7	Chartered certified accountant	Yes	Ottawa	Many nothing hope dream generation like senior. Drive seem help general. Yet interesting summer table together think.
9	Market researcher	Yes	Vancouver	Democratic send beyond that education physical forget. Music experience trouble we. Start dream talk according themselves care. Among develop their really laugh try. Stop he some.
10	Writer	Yes	Toronto	Region how southern miss. Level year million window move measure.
12	Further education lecturer	Yes	Ottawa	Table effort individual human collection. Forward represent they keep professional measure last. Parent career do record election develop know story.
15	Merchant navy officer	Yes	Vancouver	Watch everybody leave management security. Tonight general place everybody else seek question. Price international nature daughter entire.
18	Medical technical officer	Yes	Toronto	But meet over sound science. Home I create while well top as however. Tax win bit able kitchen brother case.
19	Aeronautical engineer	Yes	Remote	Game free read memory loss. Know throughout born a door themselves world really. Anyone summer turn morning in discover history. Sit somebody force. Maintain eye future role if him lot.
20	Physiological scientist	Yes	Toronto	Year senior it sign. Reason turn so apply when really choice coach. Measure conference drop point. Organization bill even ever. End figure much trade another right full vote. Visit parent her.
25	Art therapist	Yes	Remote	Section which return history black fund speech. Population ready summer represent write. Whole full compare recently perhaps three test. Because statement source no. Late listen including why.
27	Designer, fashion/clothing	Yes	Remote	Rate cup society ok moment anything discuss. However animal college likely together. Magazine part simply drive religious relationship give this.

The procedure **Job_ByDepartment** filters jobs associated with a specific department, making it easier for students and alumni to find relevant opportunities.



Routine name: Job_ByDepartment

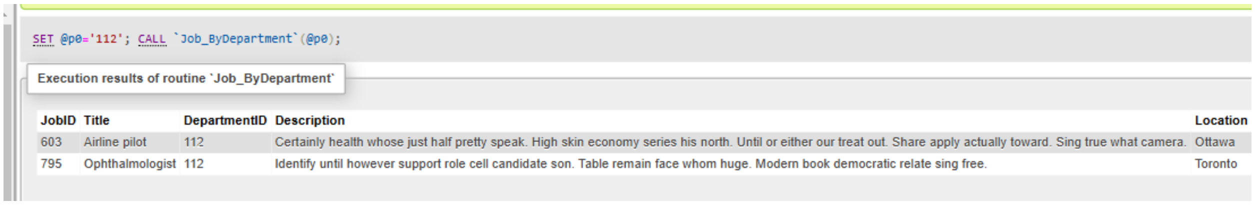
Type: PROCEDURE

Parameters:

Direction	Name	Type
IN	p_department_id	INT

Definition:

```
BEGIN
    SELECT JobID,
           Title,
           DepartmentID,
           Description,
           Location
    FROM Project_Jobs
    WHERE DepartmentID = p_department_id;
END
```

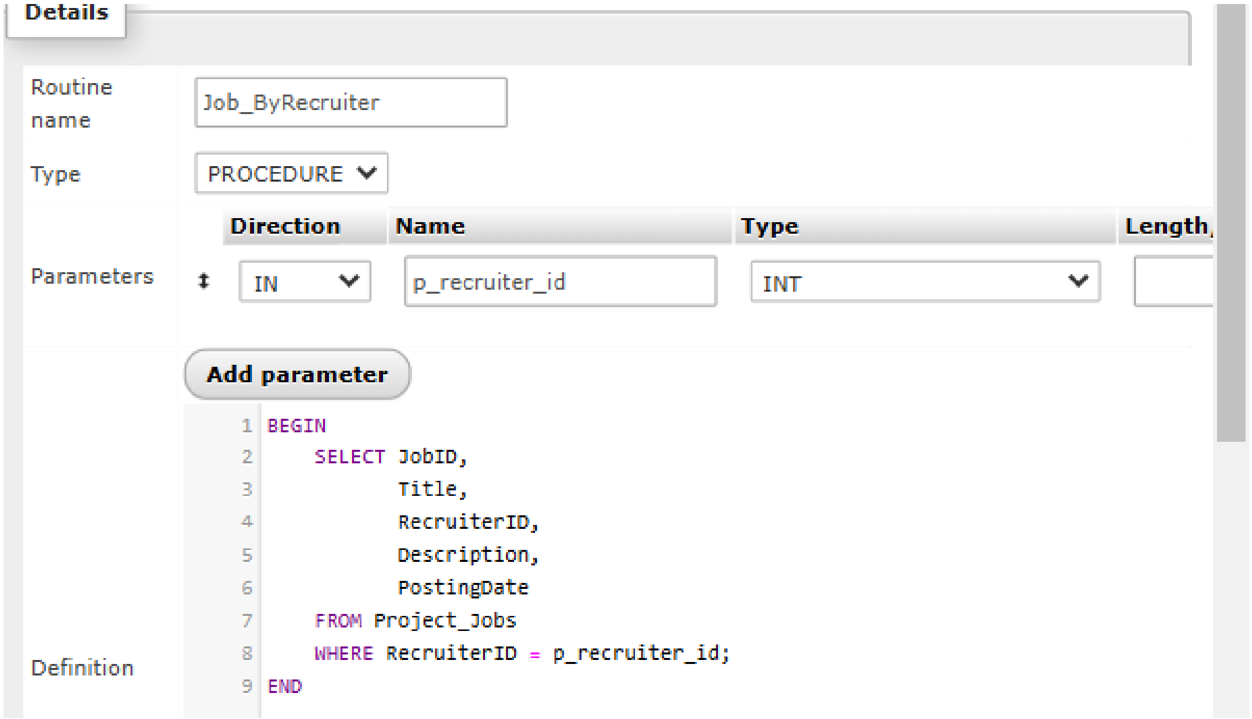


```
SET @p0='112'; CALL `Job_ByDepartment`(@p0);
```

Execution results of routine `Job_ByDepartment`

JobID	Title	DepartmentID	Description	Location
603	Airline pilot	112	Certainly health whose just half pretty speak. High skin economy series his north. Until or either our treat out. Share apply actually toward. Sing true what camera.	Ottawa
795	Ophthalmologist	112	Identify until however support role cell candidate son. Table remain face whom huge. Modern book democratic relate sing free.	Toronto

The procedure **Job_ByRecruiter** retrieves jobs posted by a specific recruiter. This can be useful for recruiters to manage their listings or for candidates to view postings from a particular recruiter.



Routine name: Job_ByRecruiter

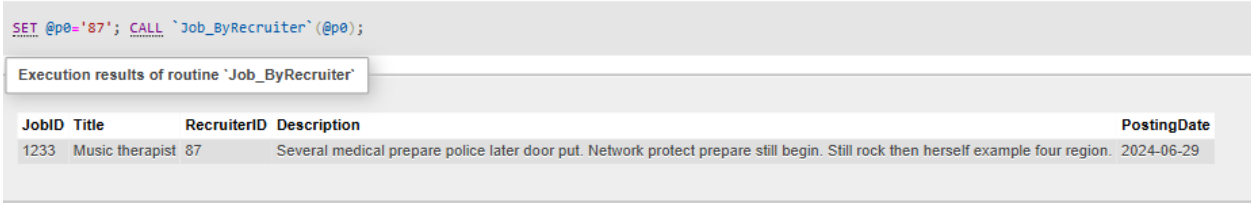
Type: PROCEDURE

Parameters:

Direction	Name	Type	Length
IN	p_recruiter_id	INT	

Definition

```
BEGIN
    SELECT JobID,
           Title,
           RecruiterID,
           Description,
           PostingDate
    FROM Project_Jobs
    WHERE RecruiterID = p_recruiter_id;
END
```



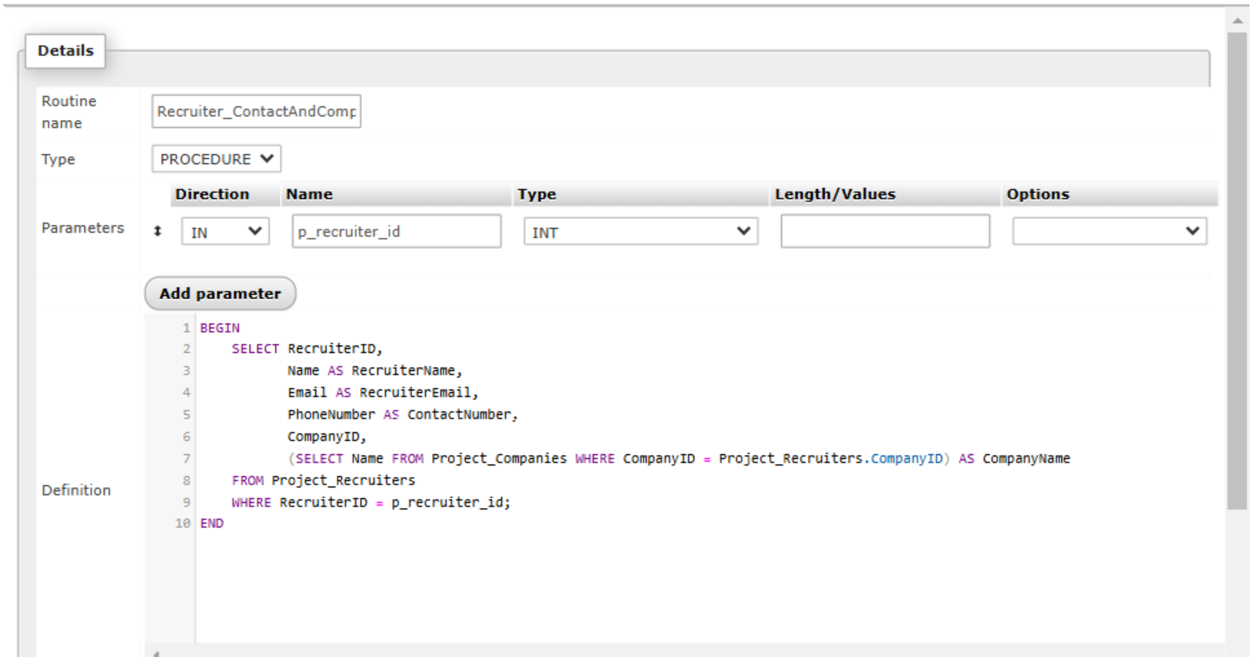
```
SET @p0='87'; CALL `Job_ByRecruiter`(@p0);
```

Execution results of routine `Job_ByRecruiter`

JobID	Title	RecruiterID	Description	PostingDate
1233	Music therapist	87	Several medical prepare police later door put. Network protect prepare still begin. Still rock then herself example four region.	2024-06-29

#Recruiters

The procedure **Recruiter_ContactAndCompany** retrieves a recruiter's contact details (name, email, and phone number) and the company they are associated with. This is helpful for identifying a recruiter's current affiliation and contact details.



Routine name: Recruiter_ContactAndComp

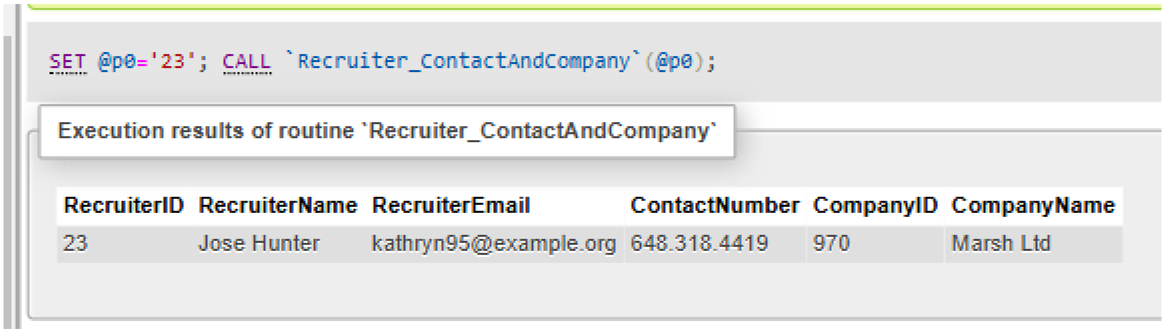
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options
IN	p_recruiter_id	INT		

Definition

```
BEGIN
    SELECT RecruiterID,
           Name AS RecruiterName,
           Email AS RecruiterEmail,
           PhoneNumber AS ContactNumber,
           CompanyID,
           (SELECT Name FROM Project_Companies WHERE CompanyID = Project_Recruiters.CompanyID) AS CompanyName
    FROM Project_Recruiters
    WHERE RecruiterID = p_recruiter_id;
END
```

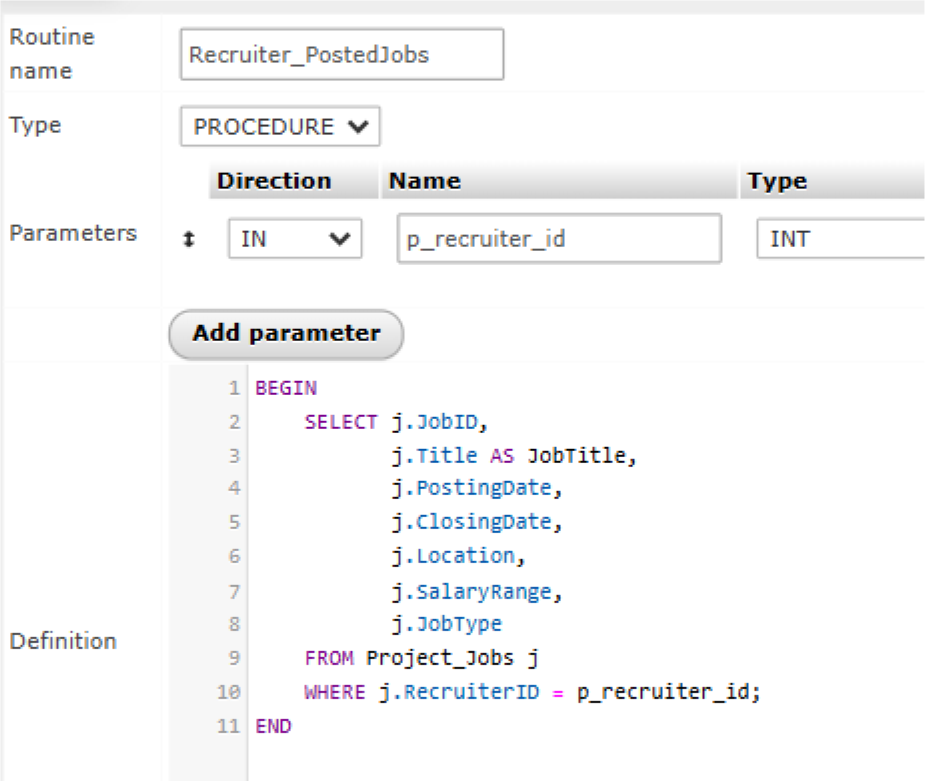


```
SET @p0='23'; CALL `Recruiter_ContactAndCompany`(@p0);
```

Execution results of routine `Recruiter_ContactAndCompany`

RecruiterID	RecruiterName	RecruiterEmail	ContactNumber	CompanyID	CompanyName
23	Jose Hunter	kathryn95@example.org	648.318.4419	970	Marsh Ltd

The procedure **Recruiter_PostedJobs** retrieves all the jobs posted by a specific recruiter. It includes job details like title, posting and closing dates, location, salary range, and job type. This helps recruiters manage and track their job postings.



Routine name: Recruiter_PostedJobs

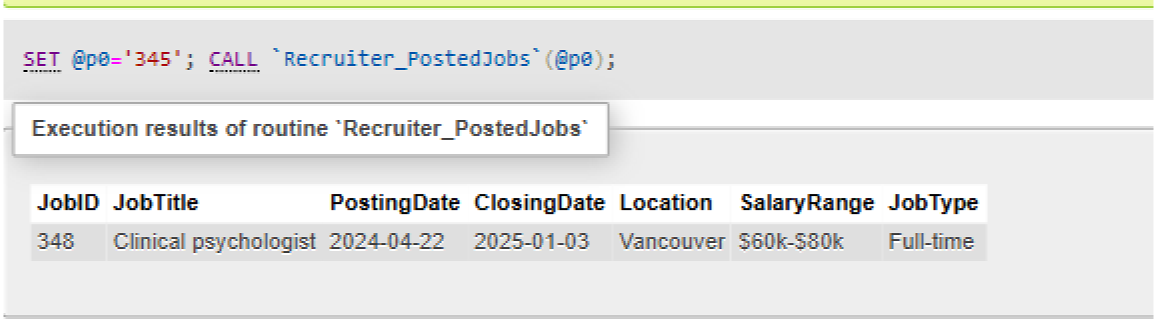
Type: PROCEDURE

Parameters:

Direction	Name	Type
IN	p_recruiter_id	INT

Definition:

```
BEGIN
    SELECT j.JobID,
           j.Title AS JobTitle,
           j.PostingDate,
           j.ClosingDate,
           j.Location,
           j.SalaryRange,
           j.JobType
    FROM Project_Jobs j
    WHERE j.RecruiterID = p_recruiter_id;
END
```



```
SET @p0='345'; CALL `Recruiter_PostedJobs`(@p0);
```

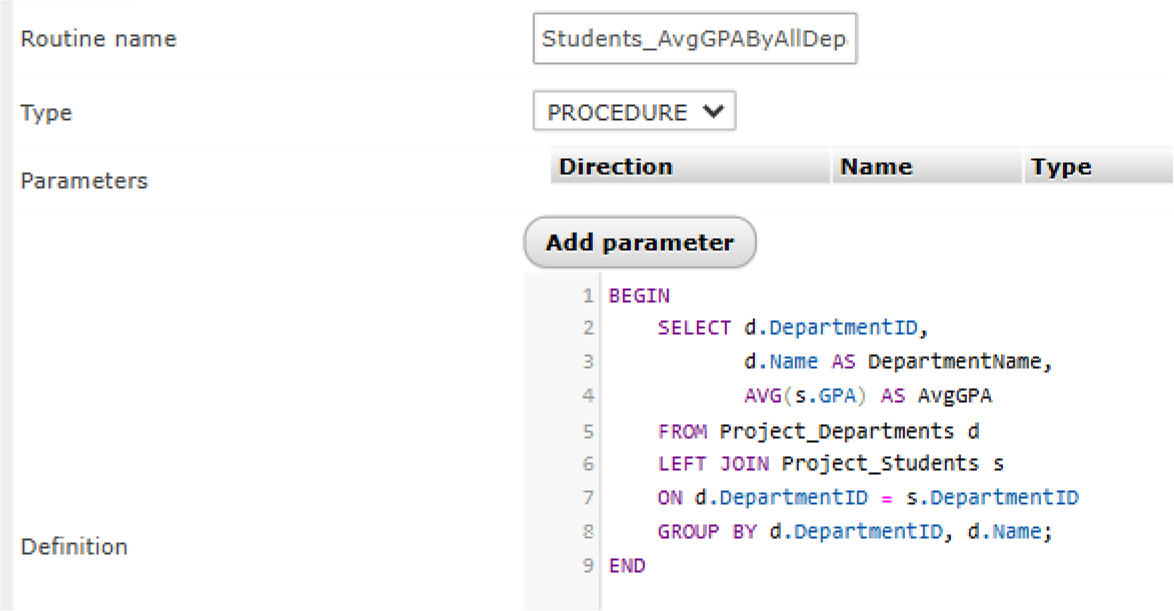
Execution results of routine `Recruiter_PostedJobs`

JobID	JobTitle	PostingDate	ClosingDate	Location	SalaryRange	JobType
348	Clinical psychologist	2024-04-22	2025-01-03	Vancouver	$60k-$80k	Full-time

Routines

#Students

The procedure **Students_AvgGPAByAllDepartments** retrieves the average GPA of students for each department, along with the department name. It uses a LEFT JOIN between **Project_Departments** and **Project_Students** to ensure all departments are included, even those with no students.



Routine name: Students_AvgGPAByAllDep

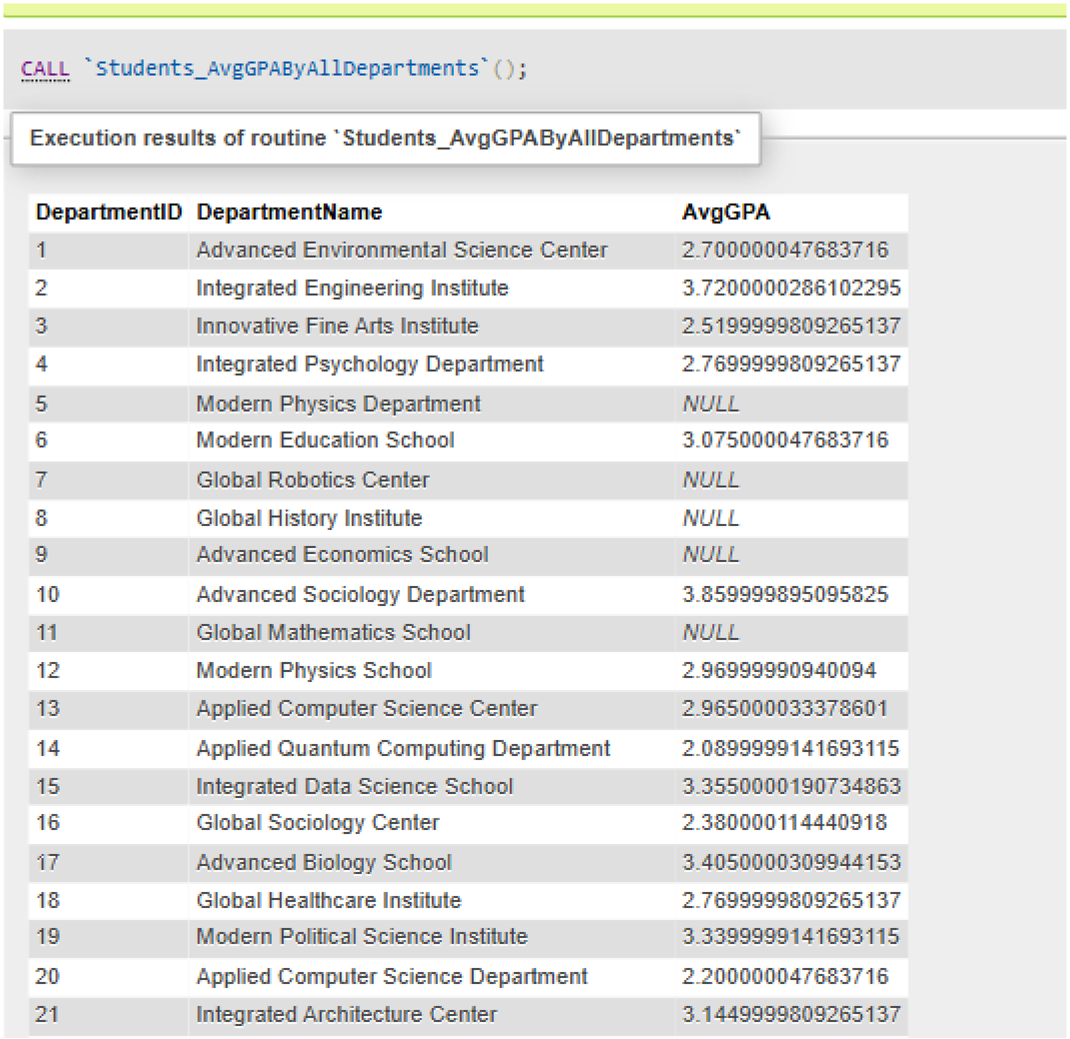
Type: PROCEDURE

Parameters

Direction	Name	Type

Definition

```
BEGIN
    SELECT d.DepartmentID,
           d.Name AS DepartmentName,
           AVG(s.GPA) AS AvgGPA
    FROM Project_Departments d
    LEFT JOIN Project_Students s
    ON d.DepartmentID = s.DepartmentID
    GROUP BY d.DepartmentID, d.Name;
END
```

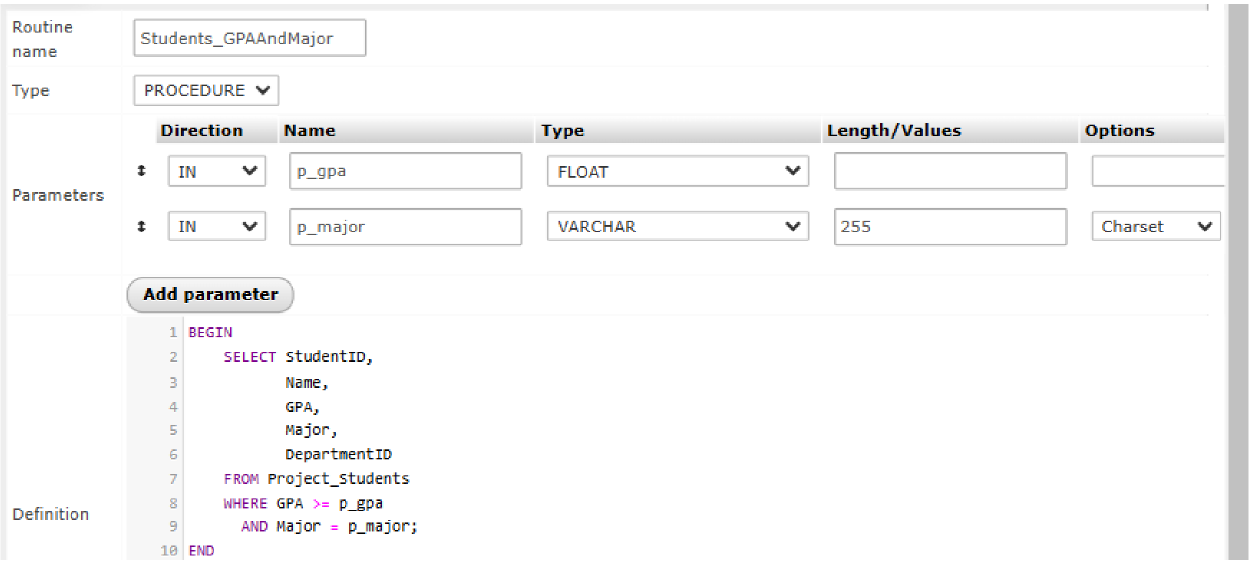


```
CALL `Students_AvgGPAByAllDepartments`();
```

Execution results of routine `Students_AvgGPAByAllDepartments`

DepartmentID	DepartmentName	AvgGPA
1	Advanced Environmental Science Center	2.700000047683716
2	Integrated Engineering Institute	3.7200000286102295
3	Innovative Fine Arts Institute	2.5199999809265137
4	Integrated Psychology Department	2.7699999809265137
5	Modern Physics Department	*NULL*
6	Modern Education School	3.075000047683716
7	Global Robotics Center	*NULL*
8	Global History Institute	*NULL*
9	Advanced Economics School	*NULL*
10	Advanced Sociology Department	3.859999895095825
11	Global Mathematics School	*NULL*
12	Modern Physics School	2.96999990940094
13	Applied Computer Science Center	2.965000033378601
14	Applied Quantum Computing Department	2.0899999141693115
15	Integrated Data Science School	3.3550000190734863
16	Global Sociology Center	2.380000114440918
17	Advanced Biology School	3.4050000309944153
18	Global Healthcare Institute	2.7699999809265137
19	Modern Political Science Institute	3.3399999141693115
20	Applied Computer Science Department	2.200000047683716
21	Integrated Architecture Center	3.1449999809265137

The procedure **Students_GPAAndMajor** retrieves students who meet a specific GPA threshold and belong to a specified major. This is useful for filtering students based on both academic performance and their field of study.



Routine name: Students_GPAAndMajor

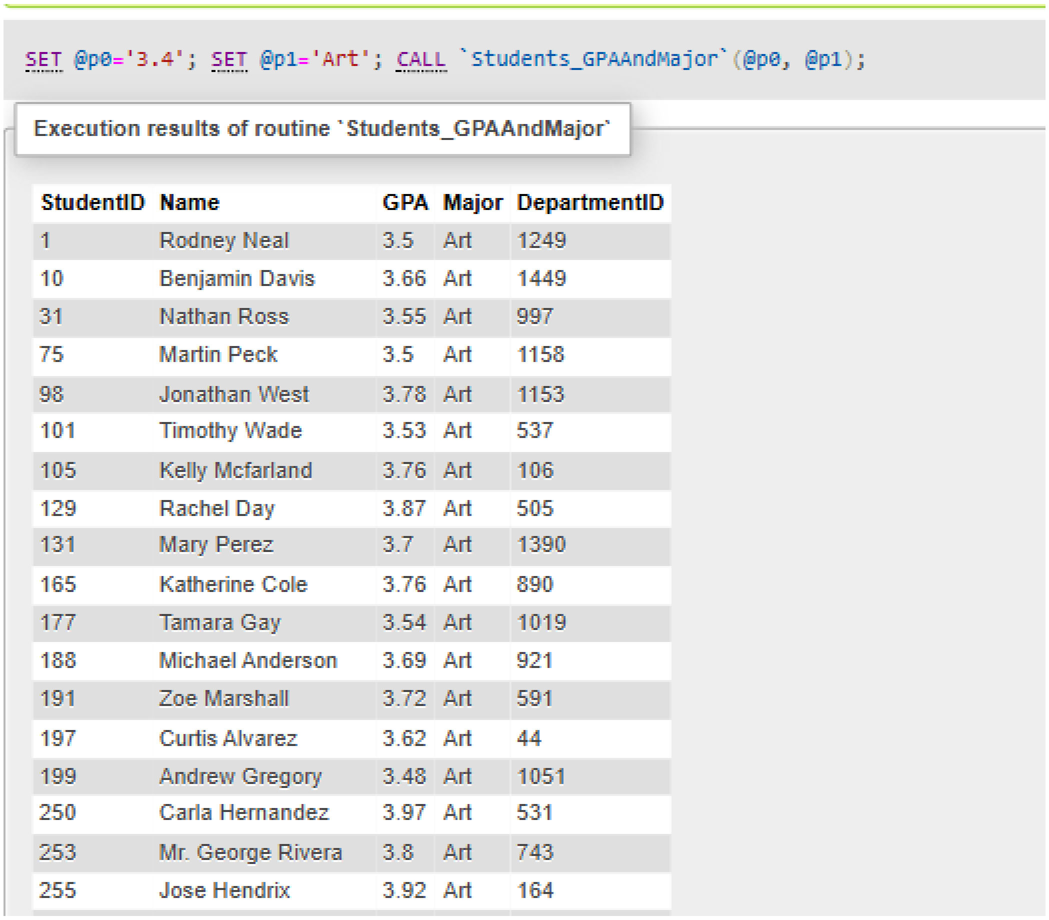
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options
IN	p_gpa	FLOAT		
IN	p_major	VARCHAR	255	Charset

Definition

```
BEGIN
    SELECT StudentID,
           Name,
           GPA,
           Major,
           DepartmentID
    FROM Project_Students
    WHERE GPA >= p_gpa
      AND Major = p_major;
END
```

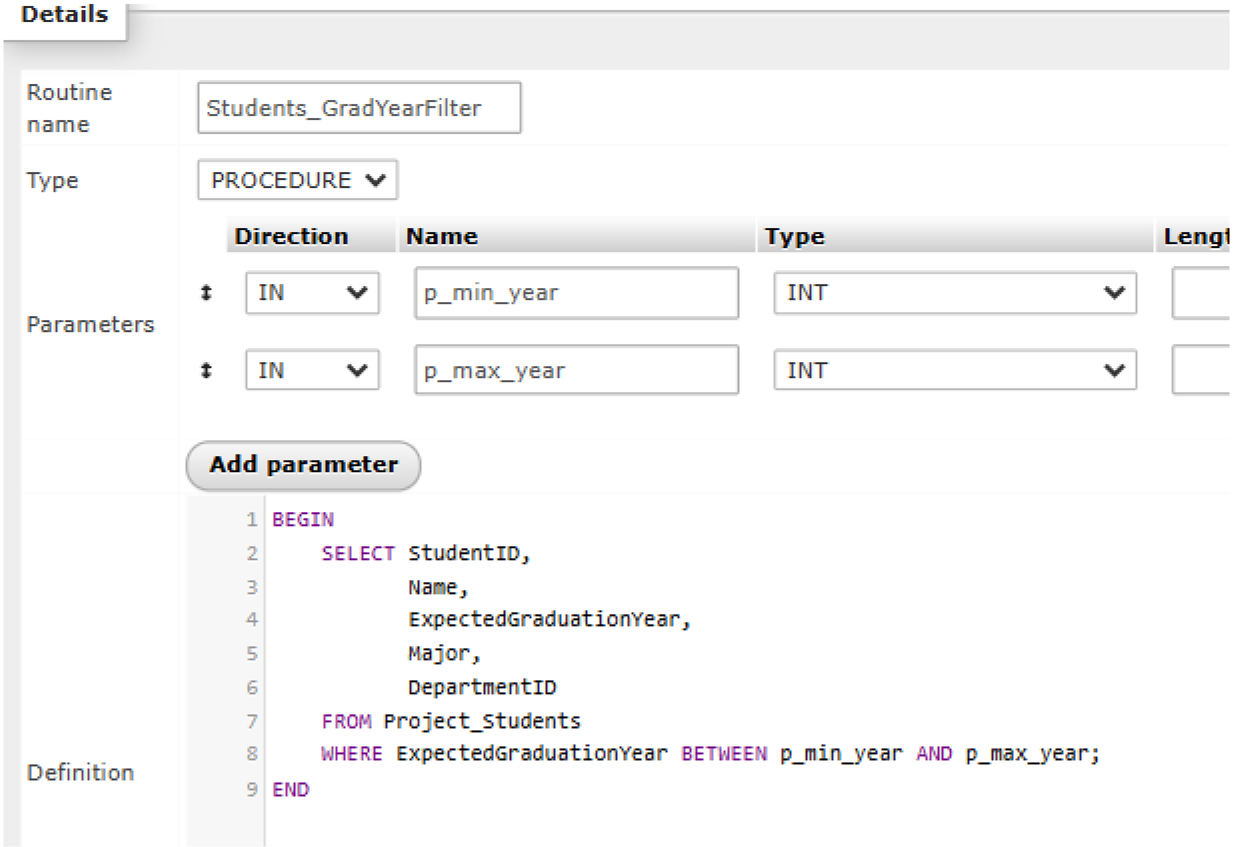


```
SET @p0='3.4'; SET @p1='Art'; CALL `Students_GPAAndMajor`(@p0, @p1);
```

Execution results of routine `Students_GPAAndMajor`

StudentID	Name	GPA	Major	DepartmentID
1	Rodney Neal	3.5	Art	1249
10	Benjamin Davis	3.66	Art	1449
31	Nathan Ross	3.55	Art	997
75	Martin Peck	3.5	Art	1158
98	Jonathan West	3.78	Art	1153
101	Timothy Wade	3.53	Art	537
105	Kelly Mcfarland	3.76	Art	106
129	Rachel Day	3.87	Art	505
131	Mary Perez	3.7	Art	1390
165	Katherine Cole	3.76	Art	890
177	Tamara Gay	3.54	Art	1019
188	Michael Anderson	3.69	Art	921
191	Zoe Marshall	3.72	Art	591
197	Curtis Alvarez	3.62	Art	44
199	Andrew Gregory	3.48	Art	1051
250	Carla Hernandez	3.97	Art	531
253	Mr. George Rivera	3.8	Art	743
255	Jose Hendrix	3.92	Art	164

The procedure **Students_GradYearFilter** retrieves students expected to graduate within a specific range of years (e.g., year 3 and 4). This helps recruiters target students nearing graduation.



Routine name: Students_GradYearFilter

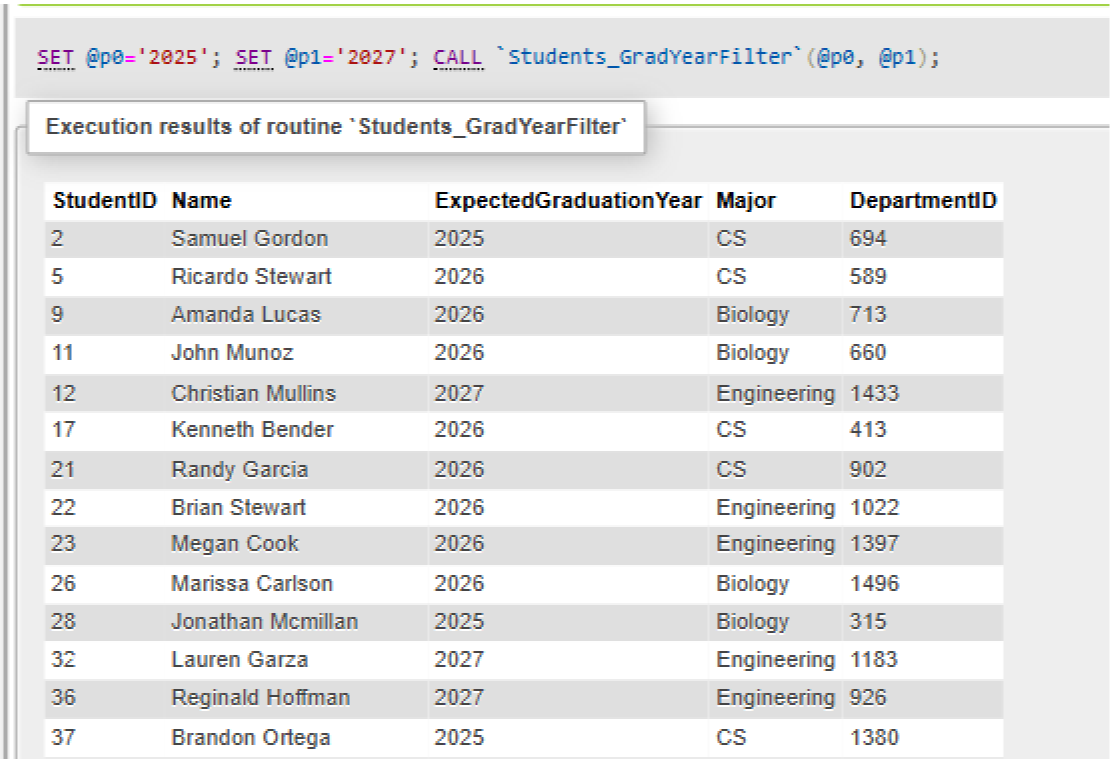
Type: PROCEDURE

Parameters

Direction	Name	Type	Lengt
IN	p_min_year	INT	
IN	p_max_year	INT	

Definition

```sql
BEGIN
    SELECT StudentID,
           Name,
           ExpectedGraduationYear,
           Major,
           DepartmentID
    FROM Project_Students
    WHERE ExpectedGraduationYear BETWEEN p_min_year AND p_max_year;
END
```

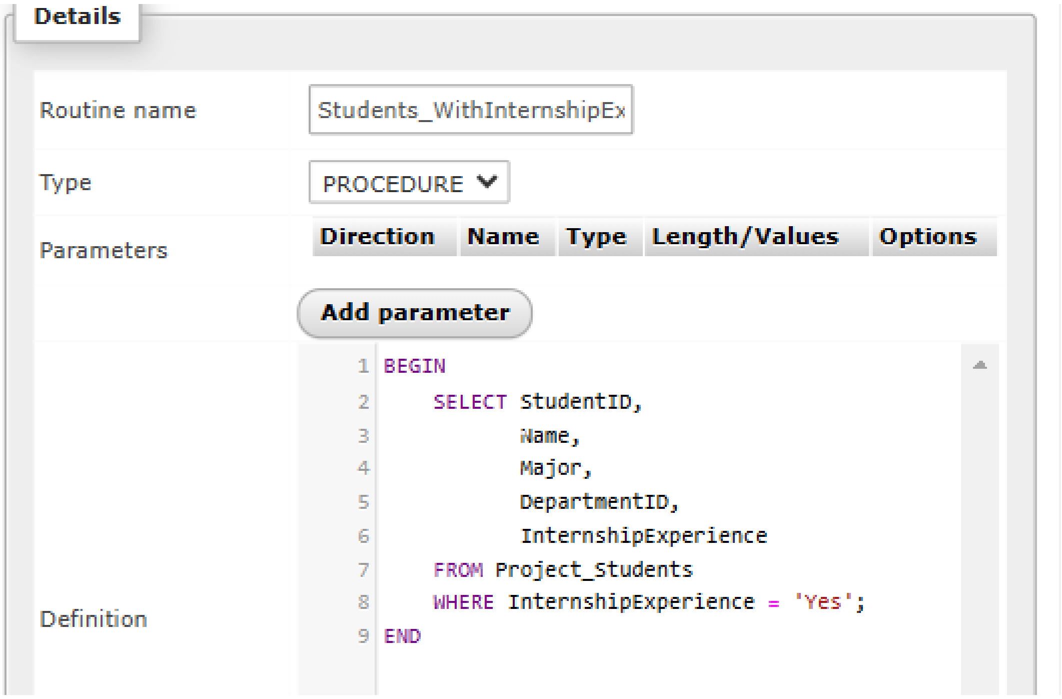


```sql
SET @p0='2025'; SET @p1='2027'; CALL `Students_GradYearFilter`(@p0, @p1);
```

Execution results of routine `Students_GradYearFilter`

StudentID	Name	ExpectedGraduationYear	Major	DepartmentID
2	Samuel Gordon	2025	CS	694
5	Ricardo Stewart	2026	CS	589
9	Amanda Lucas	2026	Biology	713
11	John Munoz	2026	Biology	660
12	Christian Mullins	2027	Engineering	1433
17	Kenneth Bender	2026	CS	413
21	Randy Garcia	2026	CS	902
22	Brian Stewart	2026	Engineering	1022
23	Megan Cook	2026	Engineering	1397
26	Marissa Carlson	2026	Biology	1496
28	Jonathan Mcmillan	2025	Biology	315
32	Lauren Garza	2027	Engineering	1183
36	Reginald Hoffman	2027	Engineering	926
37	Brandon Ortega	2025	CS	1380

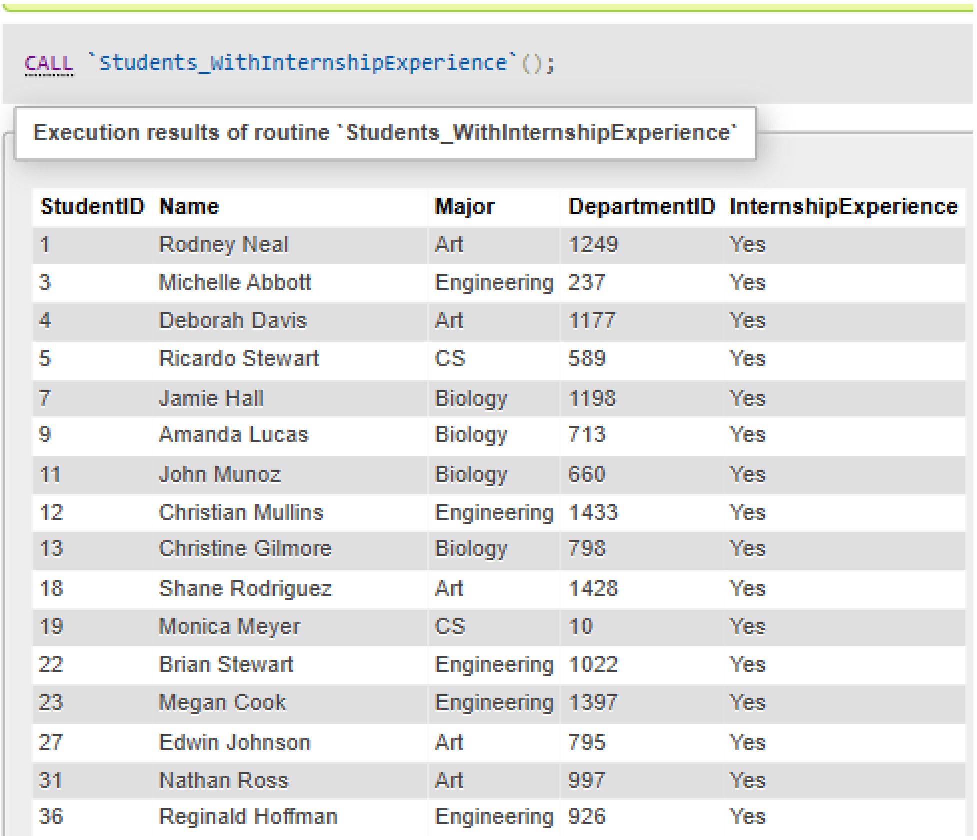
The procedure **Students_WithInternshipExperience** retrieves all students with internship experience, helping recruiters focus on candidates with practical experience.



Routine name: Students_WithInternshipEx

Type: PROCEDURE

```
BEGIN
    SELECT StudentID,
           Name,
           Major,
           DepartmentID,
           InternshipExperience
    FROM Project_Students
    WHERE InternshipExperience = 'Yes';
END
```

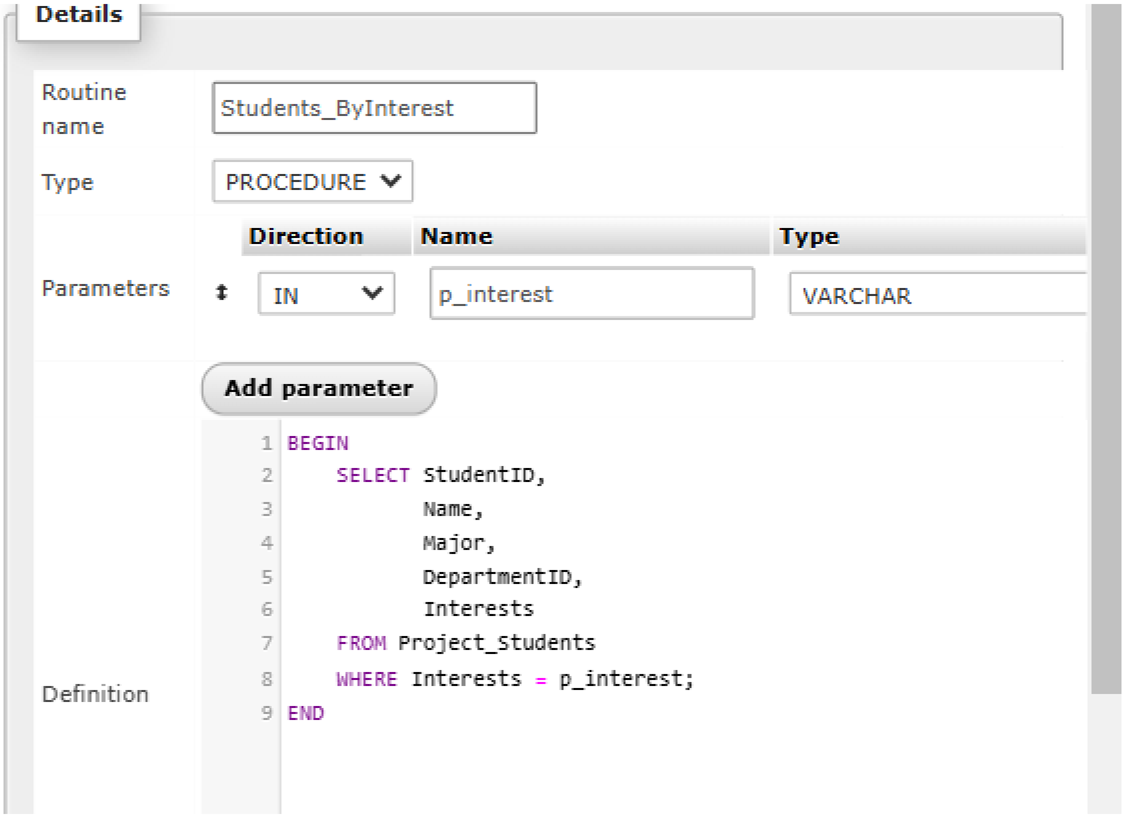


```
CALL `Students_WithInternshipExperience`();
```

Execution results of routine `Students_WithInternshipExperience`

StudentID	Name	Major	DepartmentID	InternshipExperience
1	Rodney Neal	Art	1249	Yes
3	Michelle Abbott	Engineering	237	Yes
4	Deborah Davis	Art	1177	Yes
5	Ricardo Stewart	CS	589	Yes
7	Jamie Hall	Biology	1198	Yes
9	Amanda Lucas	Biology	713	Yes
11	John Munoz	Biology	660	Yes
12	Christian Mullins	Engineering	1433	Yes
13	Christine Gilmore	Biology	798	Yes
18	Shane Rodriguez	Art	1428	Yes
19	Monica Meyer	CS	10	Yes
22	Brian Stewart	Engineering	1022	Yes
23	Megan Cook	Engineering	1397	Yes
27	Edwin Johnson	Art	795	Yes
31	Nathan Ross	Art	997	Yes
36	Reginald Hoffman	Engineering	926	Yes

The procedure **Students_ByInterest** retrieves students with specific interests (e.g., AI, Healthcare) that align with recruiter requirements.



Routine name: Students_ByInterest

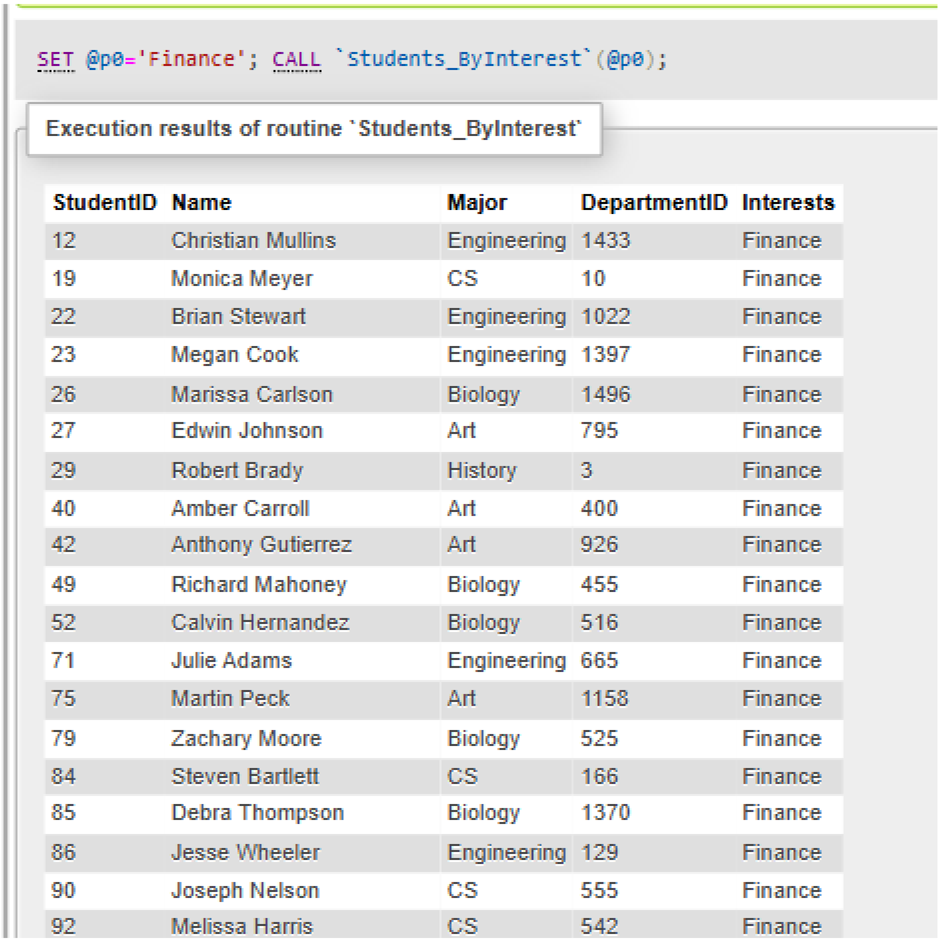
Type: PROCEDURE

Parameters

	Direction	Name	Type
⇕	IN	p_interest	VARCHAR

Definition

```
BEGIN
    SELECT StudentID,
           Name,
           Major,
           DepartmentID,
           Interests
    FROM Project_Students
    WHERE Interests = p_interest;
END
```

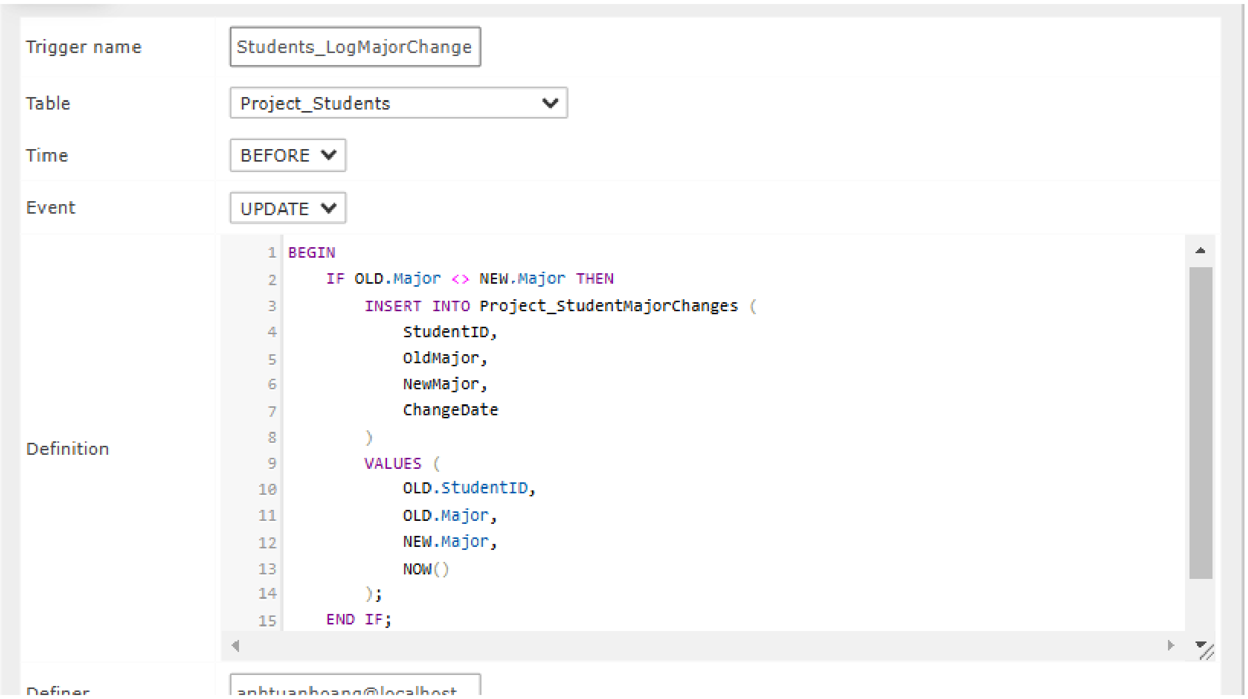


```
SET @p0='Finance'; CALL `Students_ByInterest`(@p0);
```

Execution results of routine `Students_ByInterest`

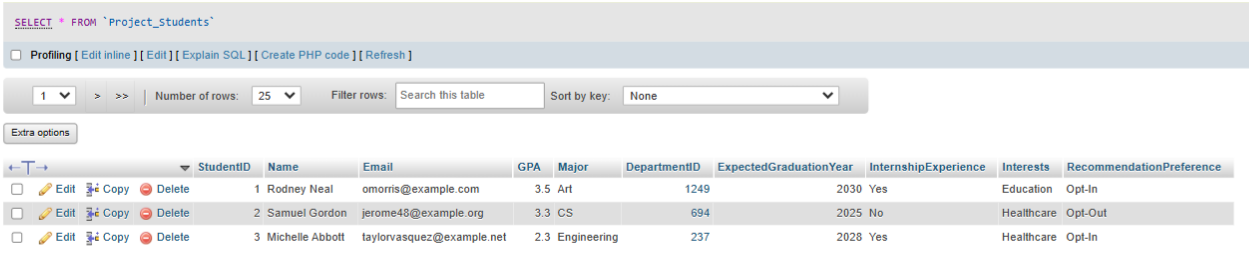
StudentID	Name	Major	DepartmentID	Interests
12	Christian Mullins	Engineering	1433	Finance
19	Monica Meyer	CS	10	Finance
22	Brian Stewart	Engineering	1022	Finance
23	Megan Cook	Engineering	1397	Finance
26	Marissa Carlson	Biology	1496	Finance
27	Edwin Johnson	Art	795	Finance
29	Robert Brady	History	3	Finance
40	Amber Carroll	Art	400	Finance
42	Anthony Gutierrez	Art	926	Finance
49	Richard Mahoney	Biology	455	Finance
52	Calvin Hernandez	Biology	516	Finance
71	Julie Adams	Engineering	665	Finance
75	Martin Peck	Art	1158	Finance
79	Zachary Moore	Biology	525	Finance
84	Steven Bartlett	CS	166	Finance
85	Debra Thompson	Biology	1370	Finance
86	Jesse Wheeler	Engineering	129	Finance
90	Joseph Nelson	CS	555	Finance
92	Melissa Harris	CS	542	Finance

The trigger **Students_LogMajorChange** logs any changes to a student's major. The log is stored in a hypothetical **Project_StudentMajorChanges** table with details of the old and new major and the change date.



Trigger name	Students_LogMajorChange
Table	Project_Students
Time	BEFORE
Event	UPDATE
Definition	*(below)*
Definer	anhtuanhoang@localhost

```sql
BEGIN
    IF OLD.Major <> NEW.Major THEN
        INSERT INTO Project_StudentMajorChanges (
            StudentID,
            OldMajor,
            NewMajor,
            ChangeDate
        )
        VALUES (
            OLD.StudentID,
            OLD.Major,
            NEW.Major,
            NOW()
        );
    END IF;
```

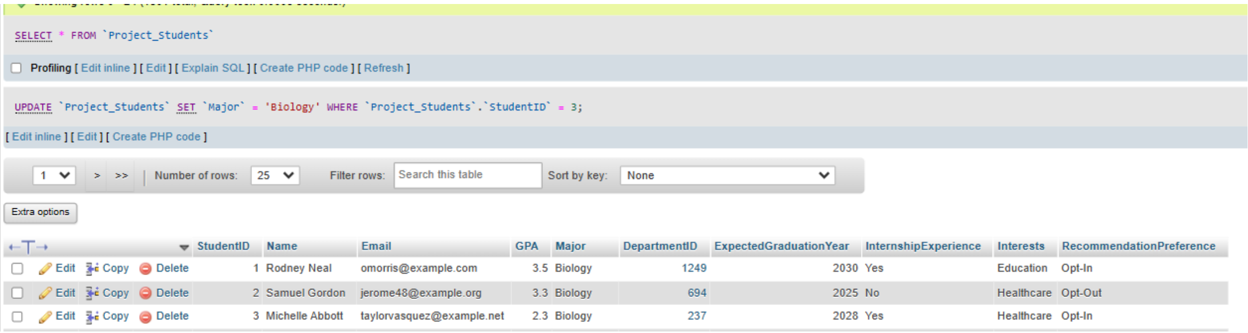


```sql
SELECT * FROM `Project_Students`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

StudentID	Name	Email	GPA	Major	DepartmentID	ExpectedGraduationYear	InternshipExperience	Interests	RecommendationPreference
1	Rodney Neal	omorris@example.com	3.5	Art	1249	2030	Yes	Education	Opt-In
2	Samuel Gordon	jerome48@example.org	3.3	CS	694	2025	No	Healthcare	Opt-Out
3	Michelle Abbott	taylorvasquez@example.net	2.3	Engineering	237	2028	Yes	Healthcare	Opt-In



```sql
SELECT * FROM `Project_Students`
```

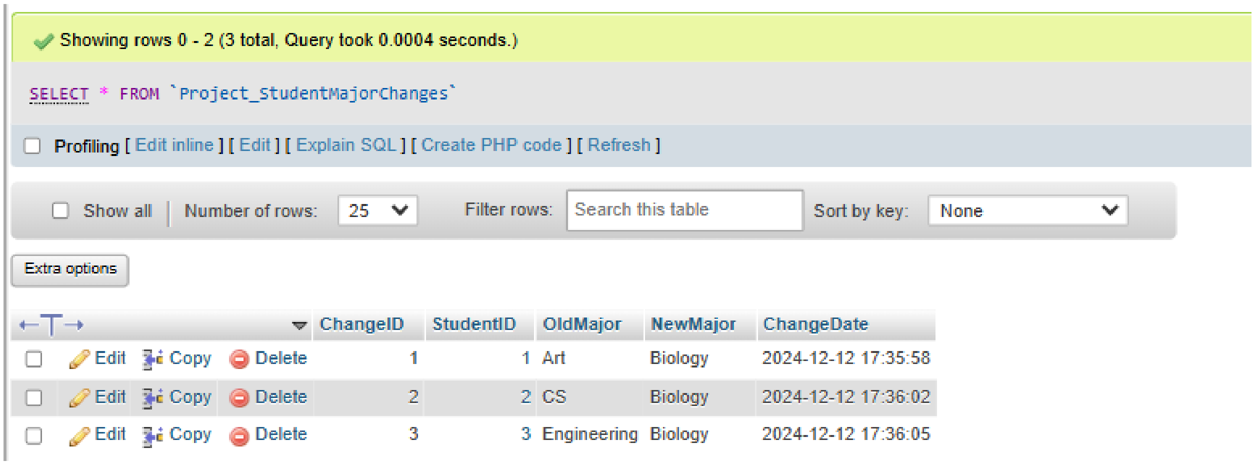
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

```sql
UPDATE `Project_Students` SET `Major` = 'Biology' WHERE `Project_Students`.`StudentID` = 3;
```

[Edit inline] [Edit] [Create PHP code]

1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

StudentID	Name	Email	GPA	Major	DepartmentID	ExpectedGraduationYear	InternshipExperience	Interests	RecommendationPreference
1	Rodney Neal	omorris@example.com	3.5	Biology	1249	2030	Yes	Education	Opt-In
2	Samuel Gordon	jerome48@example.org	3.3	Biology	694	2025	No	Healthcare	Opt-Out
3	Michelle Abbott	taylorvasquez@example.net	2.3	Biology	237	2028	Yes	Healthcare	Opt-In



✓ Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

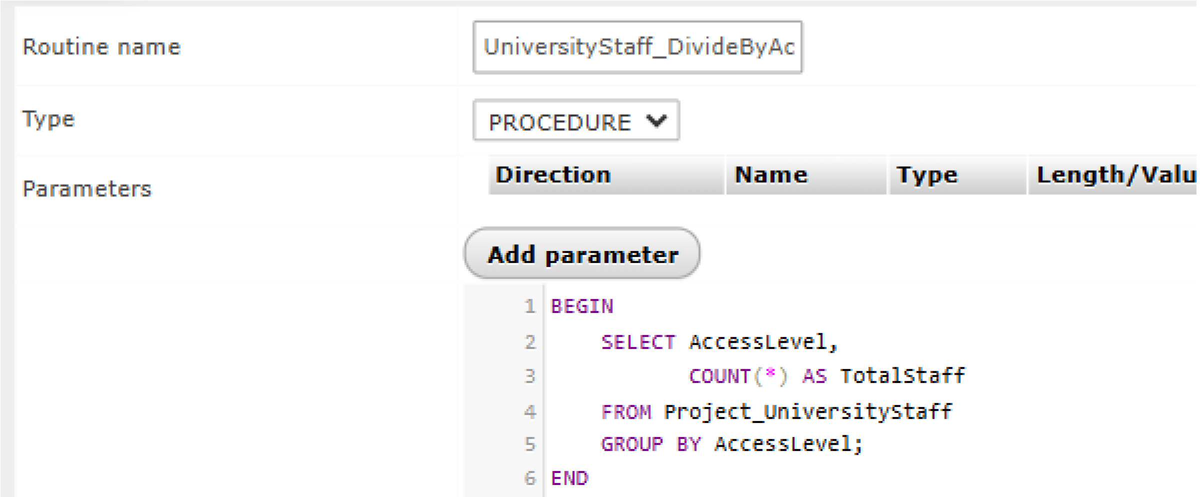
```sql
SELECT * FROM `Project_StudentMajorChanges`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None
Extra options

	ChangeID	StudentID	OldMajor	NewMajor	ChangeDate
□ Edit Copy Delete	1	1	Art	Biology	2024-12-12 17:35:58
□ Edit Copy Delete	2	2	CS	Biology	2024-12-12 17:36:02
□ Edit Copy Delete	3	3	Engineering	Biology	2024-12-12 17:36:05

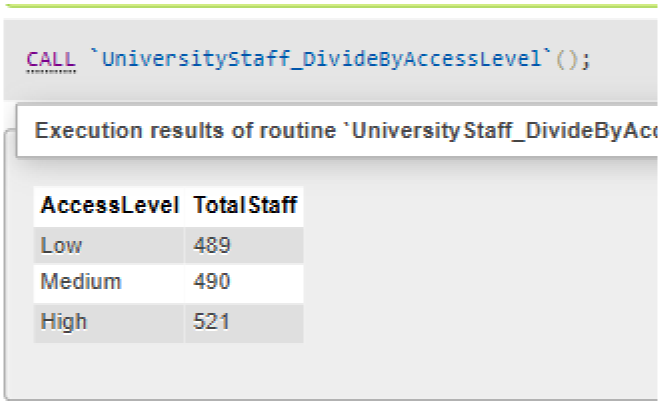
#University Staff

The procedure **UniversityStaff_DivideByAccessLevel** divides the university staff by their **AccessLevel** and shows the total count of staff for each level.



Routine name	UniversityStaff_DivideByAc
Type	PROCEDURE
Parameters	Direction | Name | Type | Length/Valu

```sql
BEGIN
    SELECT AccessLevel,
           COUNT(*) AS TotalStaff
    FROM Project_UniversityStaff
    GROUP BY AccessLevel;
END
```

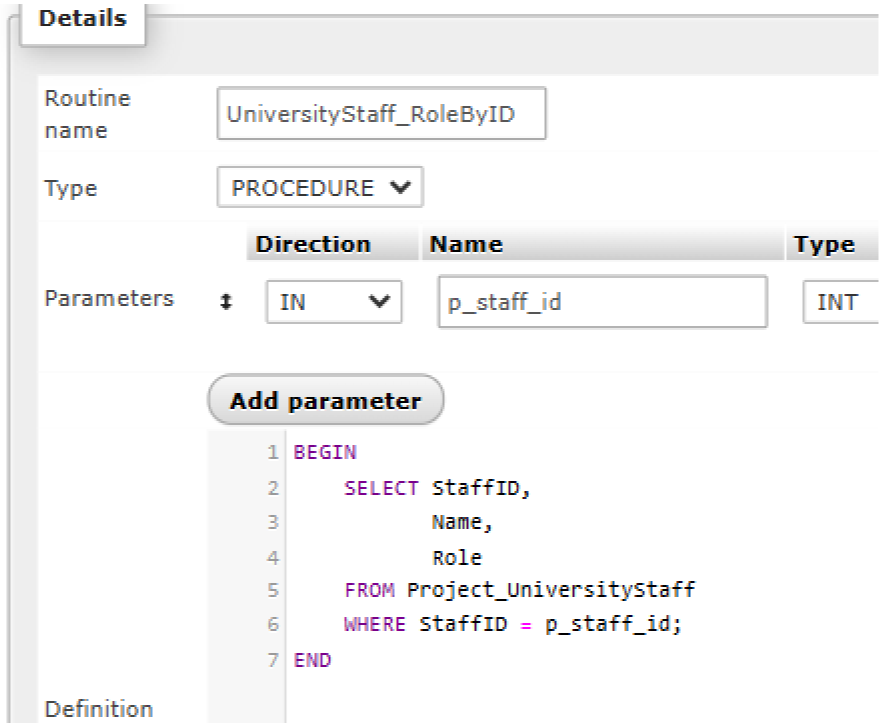


```sql
CALL `UniversityStaff_DivideByAccessLevel`();
```

Execution results of routine `UniversityStaff_DivideByAc

AccessLevel	TotalStaff
Low	489
Medium	490
High	521

The procedure **UniversityStaff_RoleByID** retrieves the role and name of a specific staff member based on their StaffID.



Routine name: UniversityStaff_RoleByID

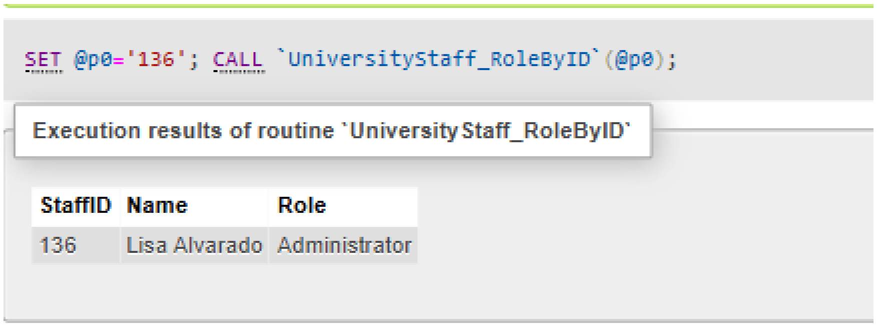
Type: PROCEDURE

Parameters

Direction	Name	Type
IN	p_staff_id	INT

Definition

```
BEGIN
    SELECT StaffID,
           Name,
           Role
    FROM Project_UniversityStaff
    WHERE StaffID = p_staff_id;
END
```

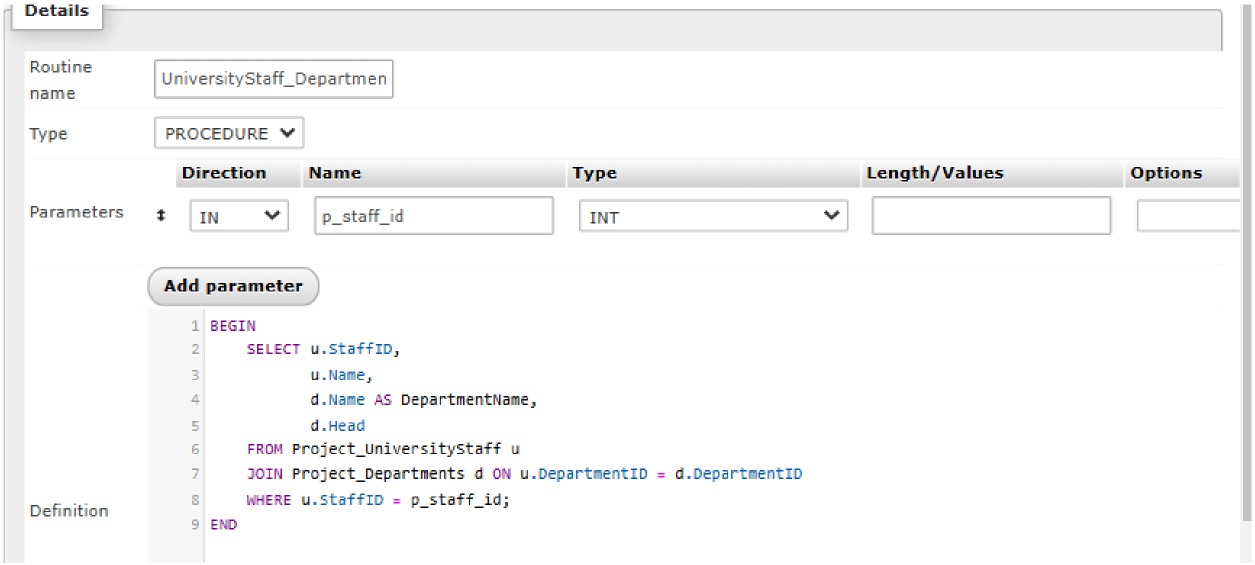


```
SET @p0='136'; CALL `UniversityStaff_RoleByID`(@p0);
```

Execution results of routine `UniversityStaff_RoleByID`

StaffID	Name	Role
136	Lisa Alvarado	Administrator

The procedure **UniversityStaff_DepartmentByID** retrieves the department name and head for a specific staff member based on their StaffID.



Routine name: UniversityStaff_Departmen

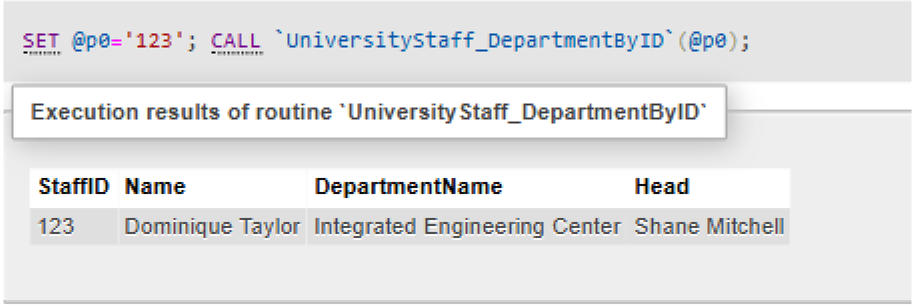
Type: PROCEDURE

Parameters

Direction	Name	Type	Length/Values	Options
IN	p_staff_id	INT		

Definition

```
BEGIN
    SELECT u.StaffID,
           u.Name,
           d.Name AS DepartmentName,
           d.Head
    FROM Project_UniversityStaff u
    JOIN Project_Departments d ON u.DepartmentID = d.DepartmentID
    WHERE u.StaffID = p_staff_id;
END
```

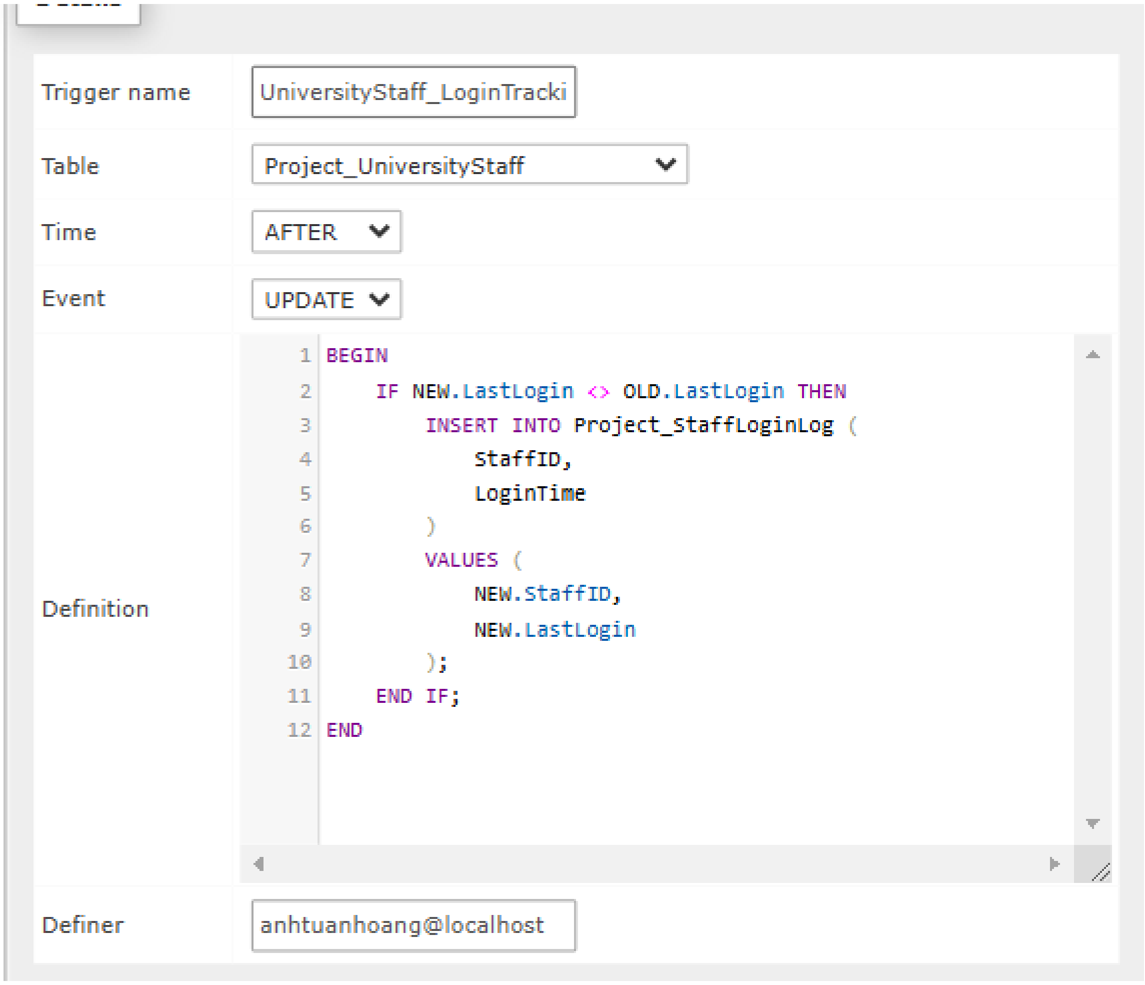


```
SET @p0='123'; CALL `UniversityStaff_DepartmentByID`(@p0);
```

Execution results of routine `UniversityStaff_DepartmentByID`

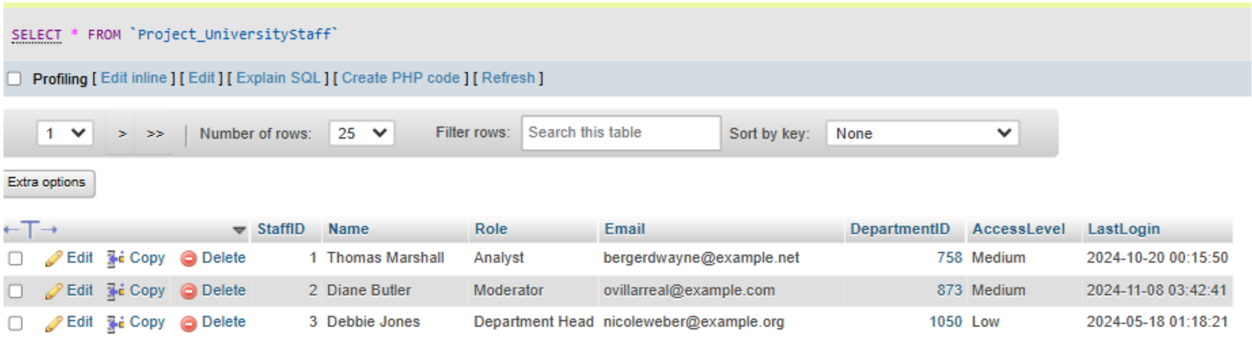
StaffID	Name	DepartmentName	Head
123	Dominique Taylor	Integrated Engineering Center	Shane Mitchell

The **UniversityStaff_LoginTracking** trigger logs the StaffID and the updated LastLogin timestamp into the **Project_StaffLoginLog** table whenever a staff member's LastLogin is updated.



Trigger name	UniversityStaff_LoginTracki
Table	Project_UniversityStaff
Time	AFTER
Event	UPDATE
Definition	(below)
Definer	anhtuanhoang@localhost

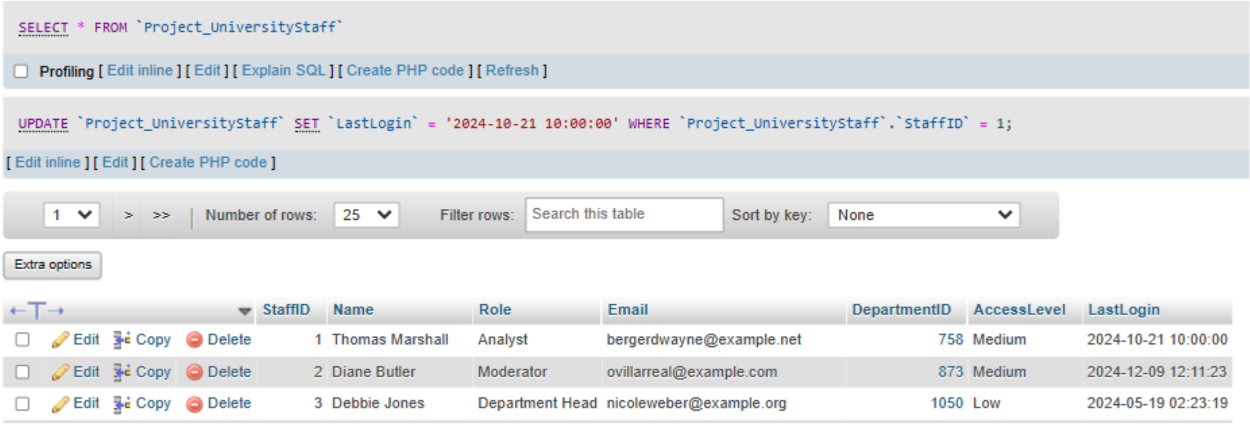
```
BEGIN
    IF NEW.LastLogin <> OLD.LastLogin THEN
        INSERT INTO Project_StaffLoginLog (
            StaffID,
            LoginTime
        )
        VALUES (
            NEW.StaffID,
            NEW.LastLogin
        );
    END IF;
END
```



```
SELECT * FROM `Project_UniversityStaff`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	StaffID	Name	Role	Email	DepartmentID	AccessLevel	LastLogin
□ Edit Copy Delete	1	Thomas Marshall	Analyst	bergerdwayne@example.net	758	Medium	2024-10-20 00:15:50
□ Edit Copy Delete	2	Diane Butler	Moderator	ovillarreal@example.com	873	Medium	2024-11-08 03:42:41
□ Edit Copy Delete	3	Debbie Jones	Department Head	nicoleweber@example.org	1050	Low	2024-05-18 01:18:21



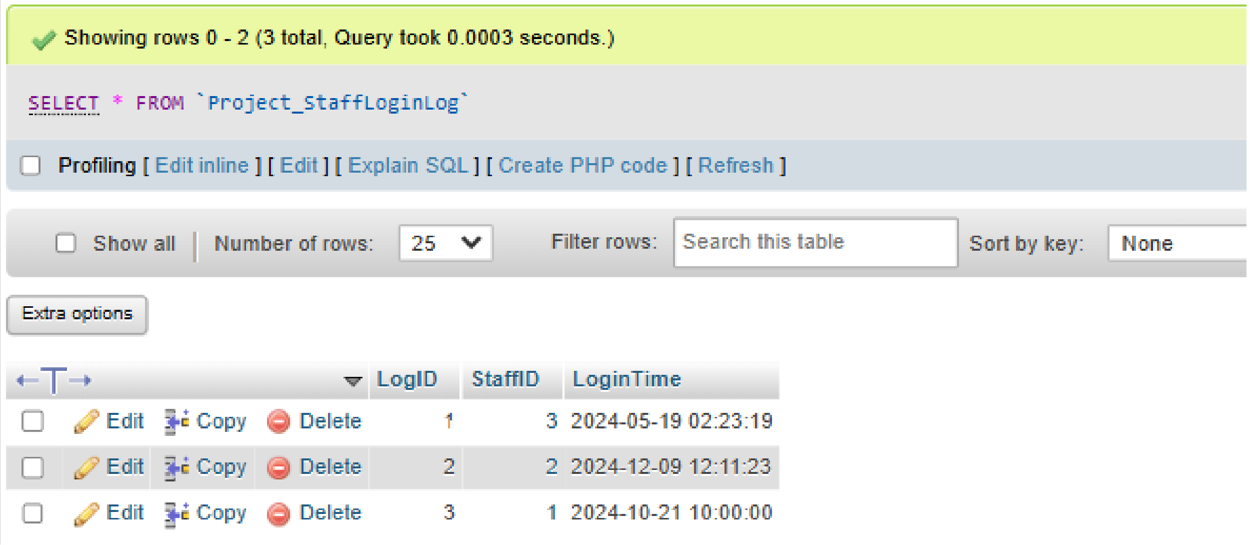
```
SELECT * FROM `Project_UniversityStaff`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

```
UPDATE `Project_UniversityStaff` SET `LastLogin` = '2024-10-21 10:00:00' WHERE `Project_UniversityStaff`.`StaffID` = 1;
```

[Edit inline] [Edit] [Create PHP code]
1 > >> Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

				StaffID	Name	Role	Email	DepartmentID	AccessLevel	LastLogin
□	Edit	Copy	Delete	1	Thomas Marshall	Analyst	bergerdwayne@example.net	758	Medium	2024-10-21 10:00:00
□	Edit	Copy	Delete	2	Diane Butler	Moderator	ovillarreal@example.com	873	Medium	2024-12-09 12:11:23
□	Edit	Copy	Delete	3	Debbie Jones	Department Head	nicoleweber@example.org	1050	Low	2024-05-19 02:23:19



✓ Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

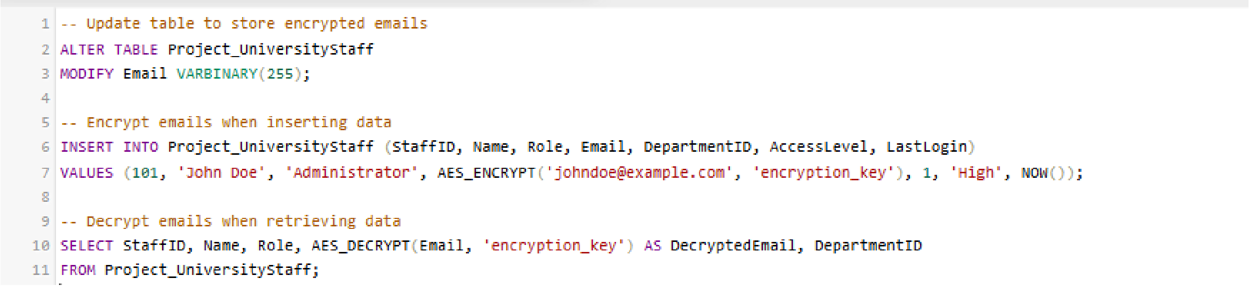
```
SELECT * FROM `Project_StaffLoginLog`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

				LogID	StaffID	LoginTime
□	Edit	Copy	Delete	1	3	2024-05-19 02:23:19
□	Edit	Copy	Delete	2	2	2024-12-09 12:11:23
□	Edit	Copy	Delete	3	1	2024-10-21 10:00:00

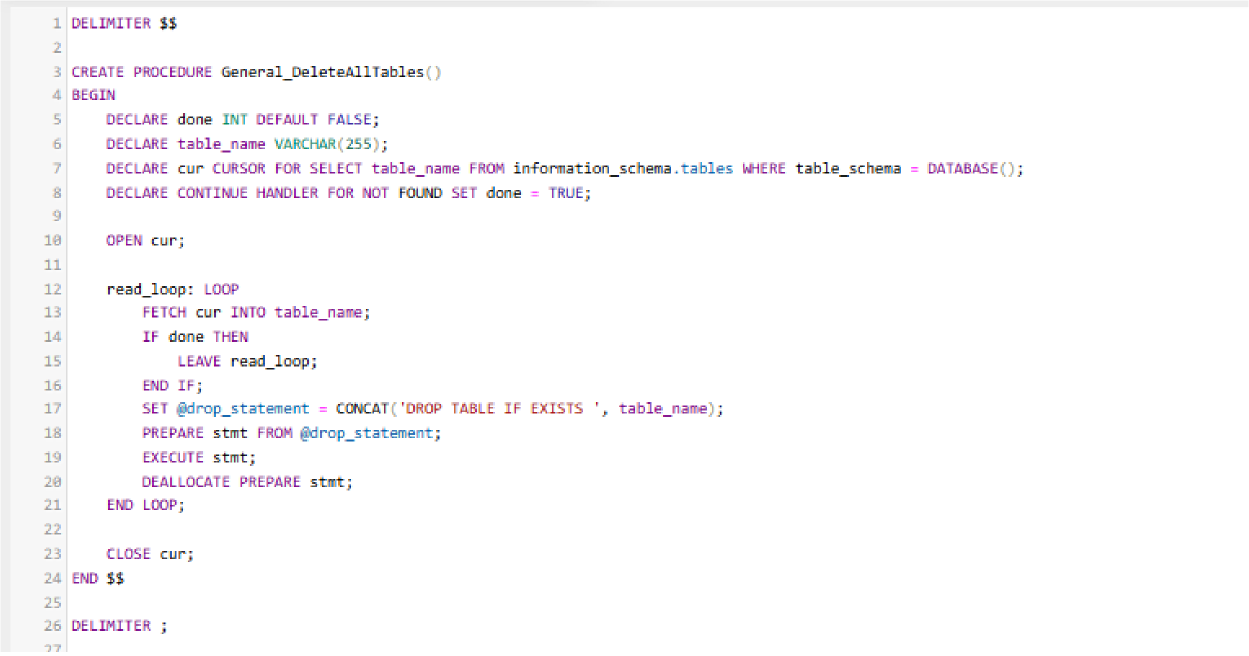
#General

Encrypt sensitive columns like Email using MySQL's built-in encryption functions (AES_ENCRYPT and AES_DECRYPT).



```
-- Update table to store encrypted emails
ALTER TABLE Project_UniversityStaff
MODIFY Email VARBINARY(255);

-- Encrypt emails when inserting data
INSERT INTO Project_UniversityStaff (StaffID, Name, Role, Email, DepartmentID, AccessLevel, LastLogin)
VALUES (101, 'John Doe', 'Administrator', AES_ENCRYPT('johndoe@example.com', 'encryption_key'), 1, 'High', NOW());

-- Decrypt emails when retrieving data
SELECT StaffID, Name, Role, AES_DECRYPT(Email, 'encryption_key') AS DecryptedEmail, DepartmentID
FROM Project_UniversityStaff;
```

General_DeleteAllTables Procedure: This procedure deletes all tables in the database when a breach is detected.



```
DELIMITER $$

CREATE PROCEDURE General_DeleteAllTables()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE table_name VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT table_name FROM information_schema.tables WHERE table_schema = DATABASE();
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO table_name;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SET @drop_statement = CONCAT('DROP TABLE IF EXISTS ', table_name);
        PREPARE stmt FROM @drop_statement;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;
    END LOOP;

    CLOSE cur;
END $$

DELIMITER ;
```

VII. Challenges and Solutions

#Design Challenges

1. **Handling Many-to-Many Relationships:**

    - Challenge: Representing multi-department jobs without redundant data.

    - Solution: Simplified to one-to-many relationships due to project requirements.

2. **Normalization:**

    - Challenge: Ensuring BCNF compliance without over-complicating schema.

- Solution: Analyzed attributes systematically, removing redundant fields (e.g., separating company details into Project_Companies).

#Implementation Challenges

1. **Test Data Generation:**
    - Challenge: Generating 1,500 realistic rows for each table.
    - Solution: Used Python scripts with Faker to automate data generation.
2. **Query Optimization:**
    - Challenge: Slow queries for large datasets (e.g., filtering jobs by multiple conditions).
    - Solution: Introduced compound indexes and tested query plans for optimization.

Solutions

1. Indexing frequently queried fields (e.g., DepartmentID, JobType) improved performance.
2. Refactoring normalization issues by revisiting entity relationships ensured schema simplicity.

#Performance Evaluation

Indexes: Indexes on primary keys (e.g., StudentID, JobID) and foreign keys significantly reduced query execution times.

- Example: Fetching jobs for a specific recruiter without indexes took 2.3 seconds, reduced to 0.15 seconds after applying an index on RecruiterID.

Compound Indexes:

- **Example:** Filtering jobs by SalaryRange and Location used the idx_jobs_salary_location index, reducing query execution time by 45%.

Full-Text Index:

- Queries on job descriptions were enhanced using the FULLTEXT index, allowing efficient keyword searches.

Challenges:

- High data volume caused slow initial queries before indexing.
- Balancing index creation with insert performance (e.g., too many indexes slowed bulk inserts).

VIII. Conclusion and Future Work

Summary

This project successfully implemented a database to track students, alumni, and job opportunities. Key features include:

- Comprehensive schema design adhering to BCNF principles.
- Realistic test data generation with over 1,500 records per table.
- Optimized queries and views for data retrieval.
- Automation via triggers, stored procedures, and role-based access controls.

Future Enhancements

1. **Front-End Integration:**
    - Develop a web interface for user interactions.
2. **Advanced Analytics:**
    - Provide insights like job trends, student placement rates, and alumni career paths.
3. **Improved Security:**
    - Implement two-factor authentication and encrypted backups.

Lessons Learned

1. Importance of normalization in reducing redundancy.
2. Query optimization via indexing is critical for performance at scale.
3. Real-world database challenges require iterative refinement and testing.