

Atlas University Data Management System

Case Study

By: Abdullah Ahmed Salah Mohamed

Index:

1. Introduction
   * Background
   * Objective
   * Overview of Components
2. Database Design
   * Relational Database Schema Definition
   * Table Design for Students, Courses, Departments, and Grades
   * Normalization for Data Integrity
   * Database Design Documentation
3. SQL Implementation
   * SQL Scripts for Database Schema Creation
   * Database Population with Sample Data
   * Testing and Validation
4. PLSQL Implementation
   * PLSQL Procedure for Updating Student Information
   * PLSQL Function for Calculating GPA
   * PLSQL Procedure for Updating Department Information
   * PLSQL Procedure for Updating Course Information
   * PLSQL Procedure for Deleting Department Information
   * PLSQL Procedure for Deleting Course Information
   * Testing PLSQL Procedures with Sample Data
5. Automation Scripts
   * Bash Script for Database Backup
   * Bash Script for Disk Space Monitoring and Alerts
6. Java Application Development
   * CRUD Operations Implementation in Java
   * Integration of Java Application with SQL Database
   * Testing Java Application with Scenarios
7. Conclusion
   * Summary of Project Accomplishments
   * Challenges Faced and Overcome
   * Future Recommendations

Introduction

1.1 Background

The Atlas University Data Management System Case Study is conceived to address the evolving needs of modern educational institutions. With the growing complexity of university operations, efficient data management becomes paramount. This project aims to harness the power of SQL, PLSQL, Bash scripting, Java SE, and OOP principles to design and implement a comprehensive data management system for Atlas University.

1.2 Objective

The primary objective of this project is to create a robust data management system that encompasses various facets of university operations, including student information, course management, departmental data, and grading. The integration of SQL, PLSQL, Bash scripting, and Java applications will facilitate seamless data handling, automation of critical tasks, and insightful reporting for better decision-making.

1.3 Overview of Components

The project involves several key components, each contributing to the overall success of the data management system:

* Database Design: Establishing a well-structured relational database schema for students, courses, departments, and grades, with a focus on normalization to ensure data integrity.
* SQL Implementation: Creating SQL scripts for building the database schema, populating it with sample data, and rigorously testing its correctness.
* PLSQL Implementation: Developing PLSQL procedures for updating student, department, and course information. As well as a Function that calculates GPA. Testing these procedures and functions with sample data.
* Automation Scripts: Implementing Bash scripts for essential tasks such as database backup and disk space monitoring.
* Java Application Development: Implementing CRUD operations, and integrating the application with the SQL database.
* Integration and Reporting: Developing a feature within the Java application to generate comprehensive reports, showcasing a list of courses, enrolled students, and average GPA for each course.
* Project Presentation: Summarizing the project's key features and achievements for effective communication and understanding.

The individual assignment nature of this project provides an opportunity for each participant to showcase their understanding and mastery of the skills acquired during the courses. The subsequent sections of this documentation will delve into each milestone, detailing the steps taken to accomplish the objectives set for each day of the project.

Database Design

2.1 Relational Database Schema Definition

Students Table:

* student\_id (PK): Unique identifier for students.
* name: Student’s name.
* dob: Date of birth.
* city: City where the student resides.
* street: Street address.
* year: Academic year.
* join\_date: Date of joining.
* department\_id (FK): Foreign key referencing the department.

Departments Table:

* department\_id (PK): Unique identifier for departments.
* name: Department name.

Courses Table:

* course\_id (PK): Unique identifier for courses.
* name: Course name.
* hours: Duration of the course in hours.
* max\_grade: Maximum achievable grade.

Enrollments Table:

* student\_id (FK): Foreign key referencing the student.
* course\_id (FK): Foreign key referencing the course.
* grade: Student’s grade.
* semester: Academic semester.
* year: Course year.

Department\_Courses Table:

* department\_id (FK): Foreign key referencing the department.
* course\_id (FK): Foreign key referencing the course.

2.2 Entity-Relationship Diagram (ERD)

The ERD visually represents the relationships between different entities in the Atlas University Data Management System, offering a clear depiction of data flow and connectivity.

1. Students Table:

* Primary Key: student\_id (PK)

2. Departments Table:

* Primary Key: department\_id (PK)

3. Courses Table:

* Primary Key: course\_id (PK)

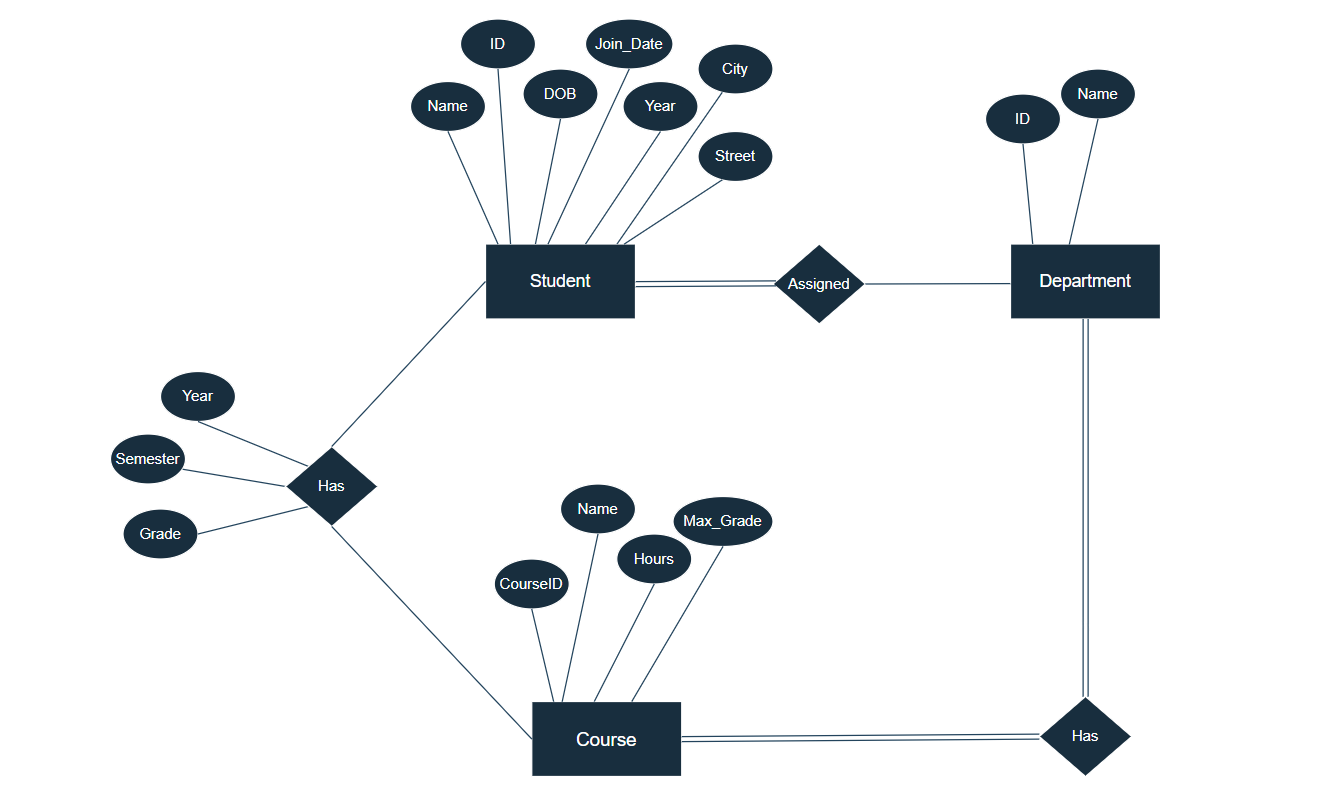
4. Enrollments Table:

* Foreign Keys: student\_id (FK), course\_id (FK)
* Associated Attributes: grade, semester, year

5. Department\_Courses Table:

* Foreign Keys: department\_id (FK), course\_id (FK)

Entity-Relationship Diagram (ERD) Visualization:



Key Relationships:

The **Students** table is connected to the **Departments** table through the foreign key relationship (**department\_id**).

The Courses table is connected to the Enrollments table through the foreign key relationship (**course\_id**).

The **Students** and **Courses** tables are connected to the **Enrollments** table, representing the enrollment of students in specific courses.

The **Departments** and **Courses** tables are connected through the **Department\_Courses** table, illustrating the relationship between departments and the courses they offer.

2.3 Normalization Importance

Elimination of Data Redundancy:

Normalization plays a vital role in eliminating data redundancy by organizing data into related tables. In the Atlas University Data Management System, it ensures that information about students, courses, and departments is stored efficiently without unnecessary duplication. Redundancy reduction not only optimizes storage space but also minimizes the risk of inconsistencies that may arise from duplicate data.

Data Integrity:

Normalization contributes to maintaining data integrity by adhering to the principles of Atomicity, Consistency, Isolation, and Durability (ACID). The structured organization of tables and relationships helps prevent anomalies such as update, insertion, and deletion errors. In the Enrollments table, for instance, normalization ensures that each grade is associated with a specific student, course, semester, and year, maintaining consistency and accuracy.

Consistent Database Structure:

A normalized database structure leads to consistency in the way data is stored, making it easier to manage and maintain. In the Atlas University system, this consistency ensures that each department is uniquely identified, each course has a defined duration, and each student's information is linked to their respective department. Consistency in data representation facilitates easier updates, reduces errors, and enhances overall system stability.

SQL Implementation

3.1.1 Create Students Table

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

name VARCHAR(255),

dob DATE,

city VARCHAR(255), street VARCHAR(255),

year INT,

join\_date DATE, department\_id INT,

FOREIGN KEY (department\_id) REFERENCES Departments(department\_id) ON DELETE SET NULL

);

Purpose:

The **Students** table stores information about students, including their unique identifier (**student\_id**), personal details, academic information, and the department they belong to.

Key Features:

**Primary Key**: **student\_id**

**Foreign Key**: **department\_id** referencing **Departments**(**department\_id**) with ON DELETE SET NULL, allowing students to exist without a specified department.

3.1.2 Create Departments Table

CREATE TABLE Departments (

department\_id INT PRIMARY KEY,

name VARCHAR(255)

);

Purpose:

The **Departments** table holds information about university departments, such as their unique identifier (**department\_id**) and department name (**name**).

Key Features:

**Primary Key**: **department\_id**

3.1.3 Create Courses Table

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

name VARCHAR(255),

hours INT,

max\_grade INT

);

Purpose:

The **Courses** table stores details about university courses, including their unique identifier (**course\_id**), name, duration in hours (**hours**), and the maximum achievable grade (**max\_grade**).

Key Features:

**Primary Key**: **course\_id**

3.1.4 Create Enrollments Table

CREATE TABLE Enrollments (

student\_id INT,

course\_id INT,

grade INT ,

semester VARCHAR(255),

year INT,

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id) ON DELETE SET NULL,

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id) ON DELETE SET NULL

);

Purpose:

The **Enrollments** table represents the enrollment of students in courses, tracking details such as student **ID**, **course ID**, **grade**, **semester**, and **academic year**.

Key Features:

**Composite Primary Key**: (**student\_id, course\_id**)

Foreign Keys: **student\_id** referencing **Students**(**student\_id**) and **course\_id** referencing **Courses**(**course\_id**) with ON DELETE SET NULL, allowing flexibility in managing enrollments.

3.1.5 Create Department\_Courses Table

CREATE TABLE Department\_Courses (

department\_id INT,

course\_id INT,

PRIMARY KEY (department\_id, course\_id),

FOREIGN KEY (department\_id) REFERENCES Departments(department\_id) ON DELETE SET NULL,

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id) ON DELETE SET NULL

);

Purpose:

The **Department\_Courses** table establishes the relationship between departments and the courses they offer, linking department IDs with course IDs.

Key Features:

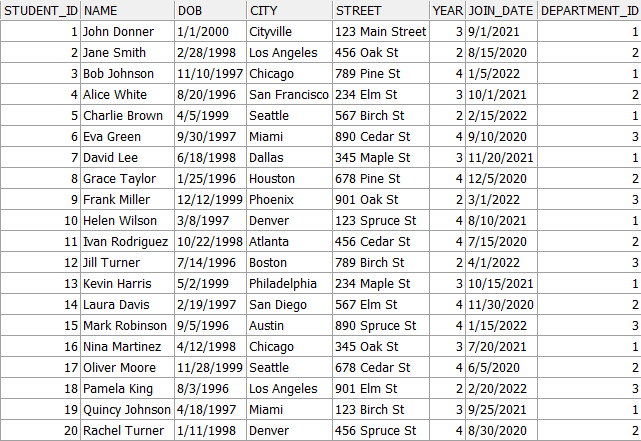
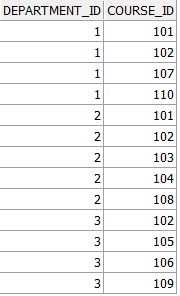
**Composite Primary Key**: (**department\_id, course\_id**)

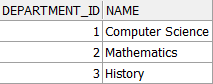
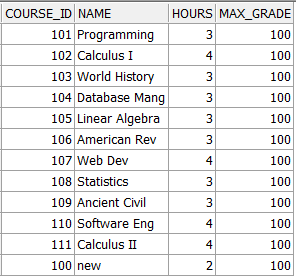
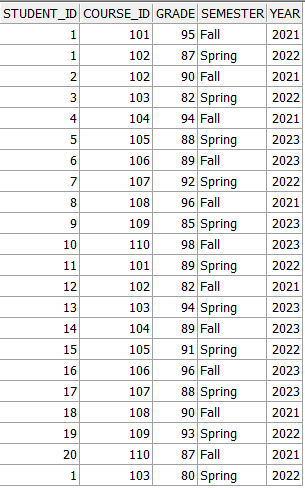
Foreign Keys: **department\_id** referencing **Departments**(**department\_id**) and **course\_id** referencing **Courses**(**course\_id**) with ON DELETE SET NULL, ensuring referential integrity.

These SQL scripts create the foundational tables for the Atlas University Data Management System, establishing relationships and constraints to maintain data integrity. In the next step, we can move on to populating the database with sample data. Let me know if you have any specific details to add or if you'd like to proceed.

3.2 Database Population with Sample Data

After creating the tables, the next step is to populate them with sample data to ensure that the database is functioning as intended.

These sample data entries represent a basic scenario with students enrolled in courses, each associated with a specific department.

3.3 Testing and Validation

Testing and validation are crucial to ensure the correctness and reliability of the database. Key aspects to consider include:

* Data Accuracy: Verify that the sample data accurately reflects the relationships between students, departments, courses, and enrollments.
* Referential Integrity: Confirm that foreign key constraints are properly enforced, ensuring that references between tables are valid.
* Primary Key Uniqueness: Ensure that primary key values are unique for each record in their respective tables.
* Data Consistency: Validate that data types and constraints are applied correctly, preventing inconsistencies in data representation.
* Query Execution: Execute sample queries to retrieve information from the database and confirm that the results align with expectations.
* Error Handling: Test error scenarios, such as attempting to insert duplicate primary key values, to validate that appropriate error messages or actions are triggered.

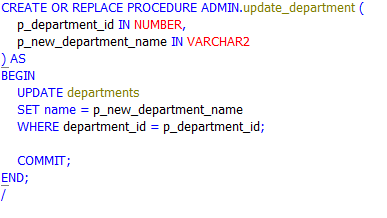
By rigorously testing and validating the database, you can identify and address any issues early in the development process, ensuring a robust and reliable data management system for Atlas University.

PLSQL Implementation

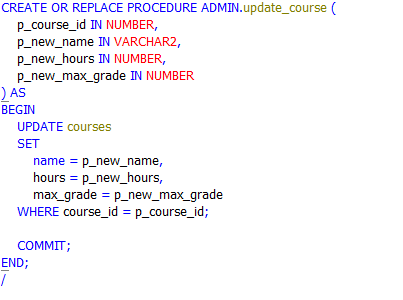
4.1.1 PLSQL Procedure for Updating Student Information



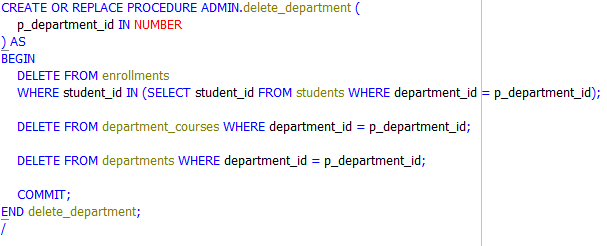
4.1.2 PLSQL Procedure for Updating Department Information



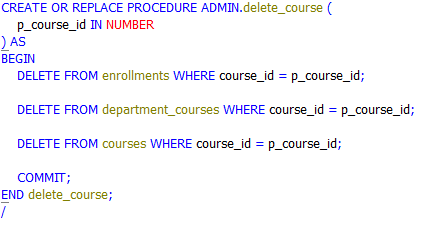
4.1.3 PLSQL Procedure for Updating Course Information



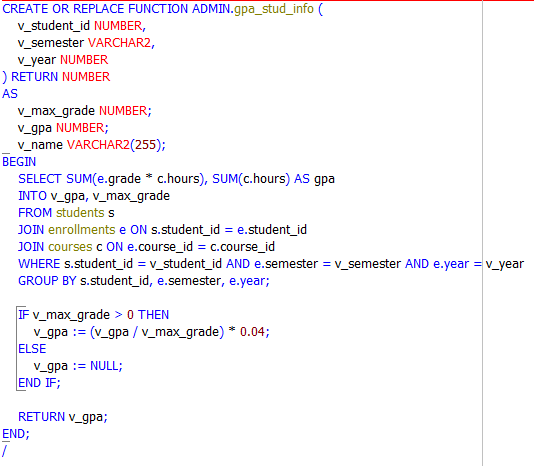
4.1.4 PLSQL Procedure for Deleting Department Information



4.1.5 PLSQL Procedure for Deleting Course Information

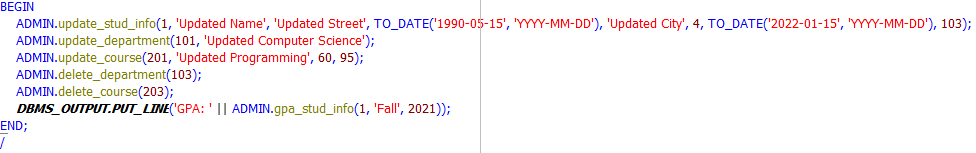


4.1.6 PLSQL Function for Calculating GPA



4.2 Testing PLSQL Procedures with Sample Data

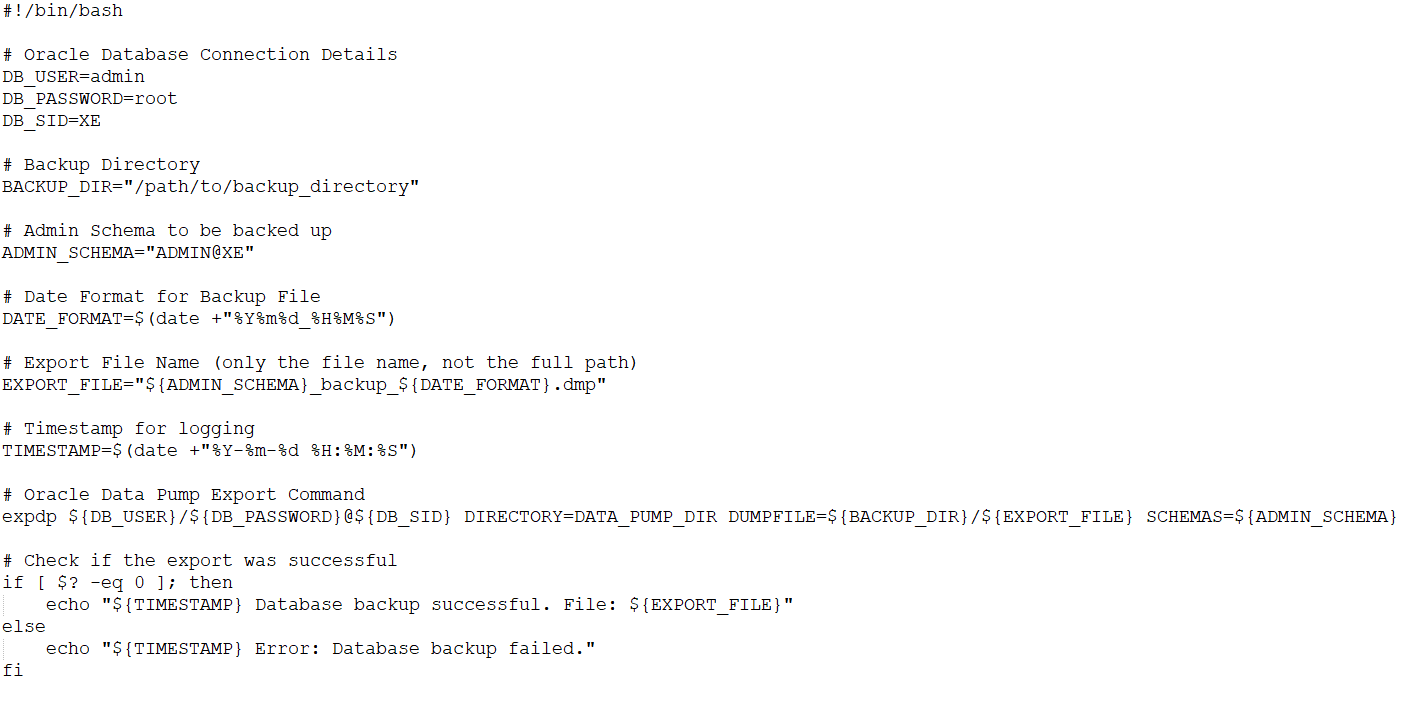
To test these procedures, you can execute them with sample data. For example:



These test examples demonstrate how to call and test each PLSQL procedure and function with sample data. The provided test cases cover updating student information, department information, course information, and deleting departments and courses, along with calculating GPA.

Automation Scripts

5.1 Bash Script for Database Backup



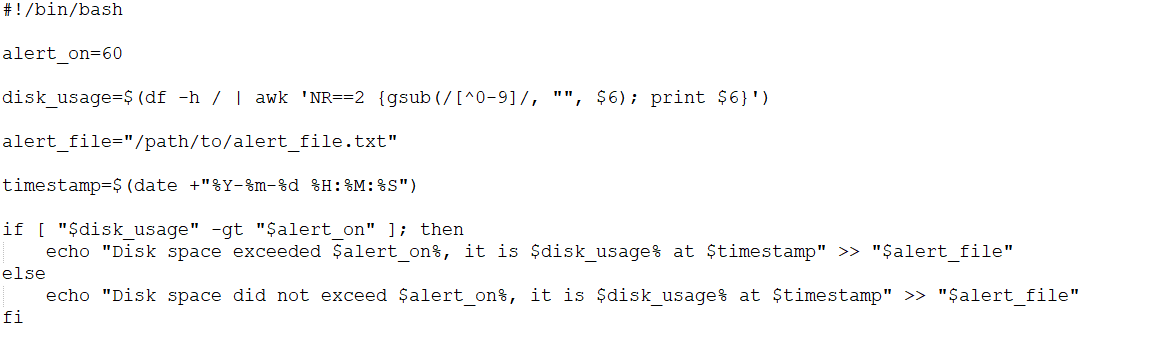
Explanation:

This script uses Oracle Data Pump (expdp) to export the specified schema from the database.

It includes parameters such as database connection details, backup directory, schema to be backed up, and a timestamped export file name.

After the export, it checks the exit status to determine if the backup was successful and logs the result.

5.2 Bash Script for Disk Space Monitoring and Alerts



Explanation:

This script monitors disk space usage using df and extracts the percentage usage for the specified directory (in this case, /).

It compares the current disk usage with the predefined threshold (alert\_on) and logs the result with a timestamp to an alert file.

If disk usage exceeds the threshold, an alert is written to the file.

5.3 Scheduling Scripts with Windows Task Scheduler

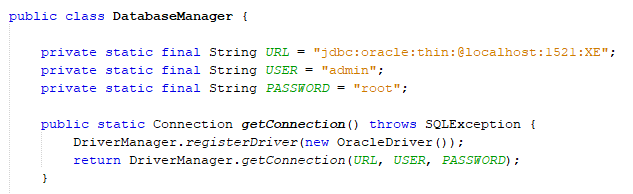
* Ensure that the Bash executable path is correctly specified in the Task Scheduler.
* Verify that the scripts have the necessary permissions to execute.
* If the script relies on environment variables, set them explicitly in the script or configure Task Scheduler to run the script in a specific directory.
* Regularly check the Task Scheduler logs for any issues with script execution.

By scheduling these tasks with Windows Task Scheduler, it is automating the execution of Bash scripts, simulating the scheduling functionality similar to cron on Linux.

Java Application Development

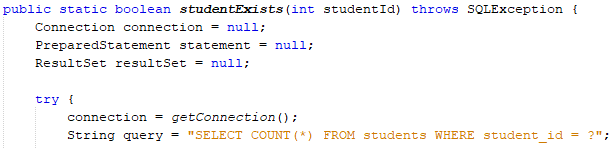
In this section, we'll delve into the Java application development for the University's data management system. The application will utilize Object-Oriented Programming (OOP) principles and interact with the SQL database.

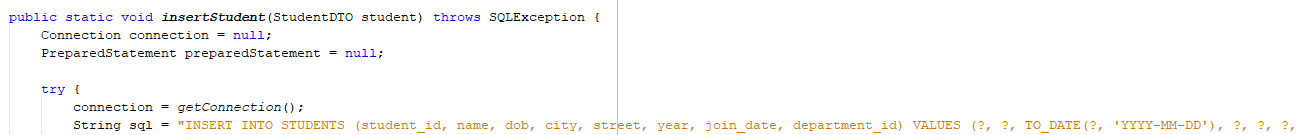
6.1 Database Connection Configuration



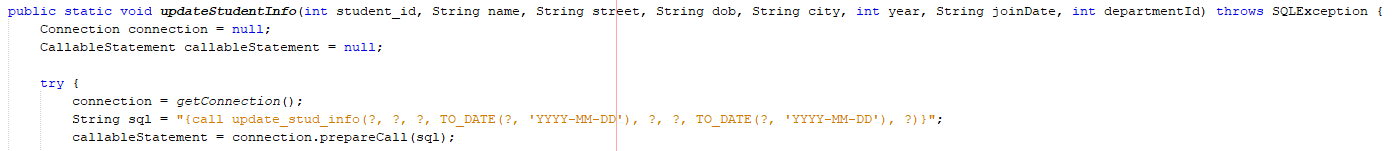
6.2 Student Information CRUD Operations

Inserting into the students table, and checking first if the student exists

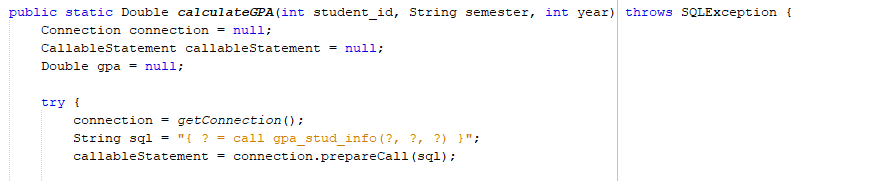




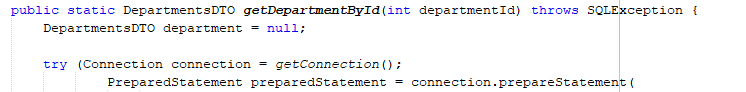
Calling the update student procedure



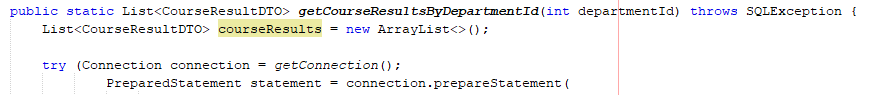
Calling the Calc GPA function



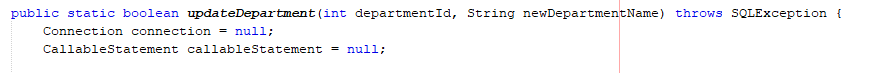
6.3 Department Information Retrieval

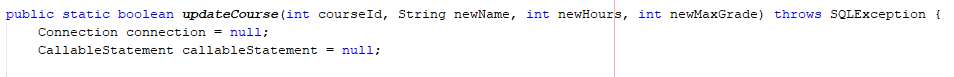


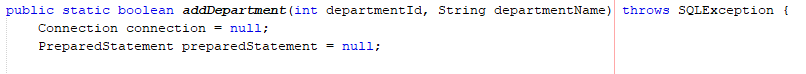
Course Results and Information

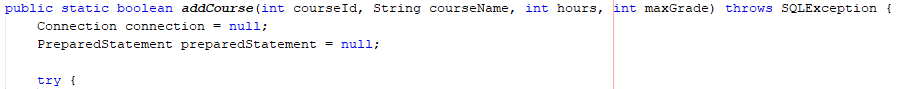


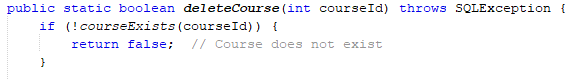
6.4 Department and Course CRUD Operations

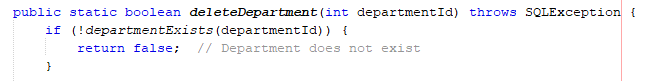












Calling out these methods ensures a smooth interaction between the database and the application. Making sure that every controller has the appropriate methods to call.

**AND HERE IS THE MOMENT TO UNVEIL THE CURTAINS OF THE APP!**

1. Main Page



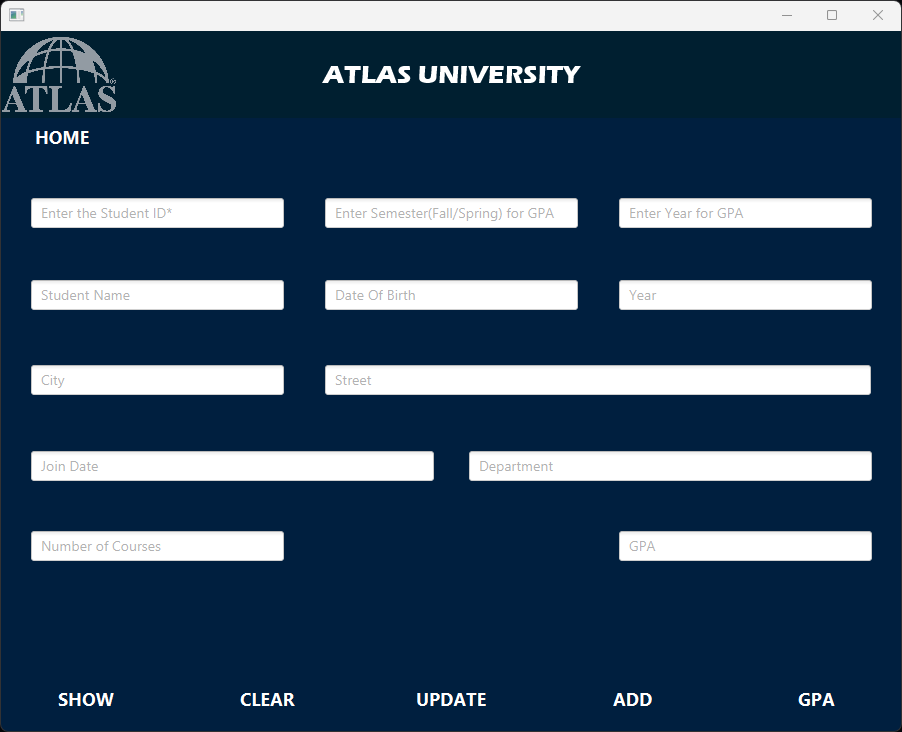
On this page you have the front of the App, and you get you choose your next destination.

You can choose between the Students page; in which you can check any student info or update it.

You can choose the Management page; in which you can check more reports and retrieve departments and courses information, as well applying CRUD there on the Departments or Courses.

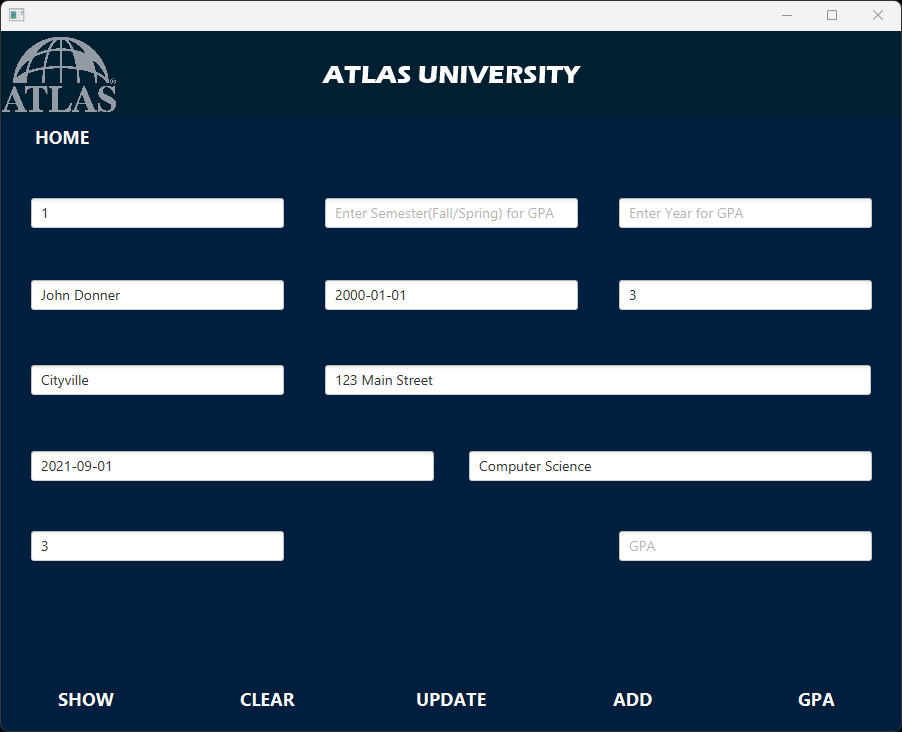
You can choose the Analysis page; in which you will see a summary of visuals on almost every important information about Atlas University.

1. Students Page

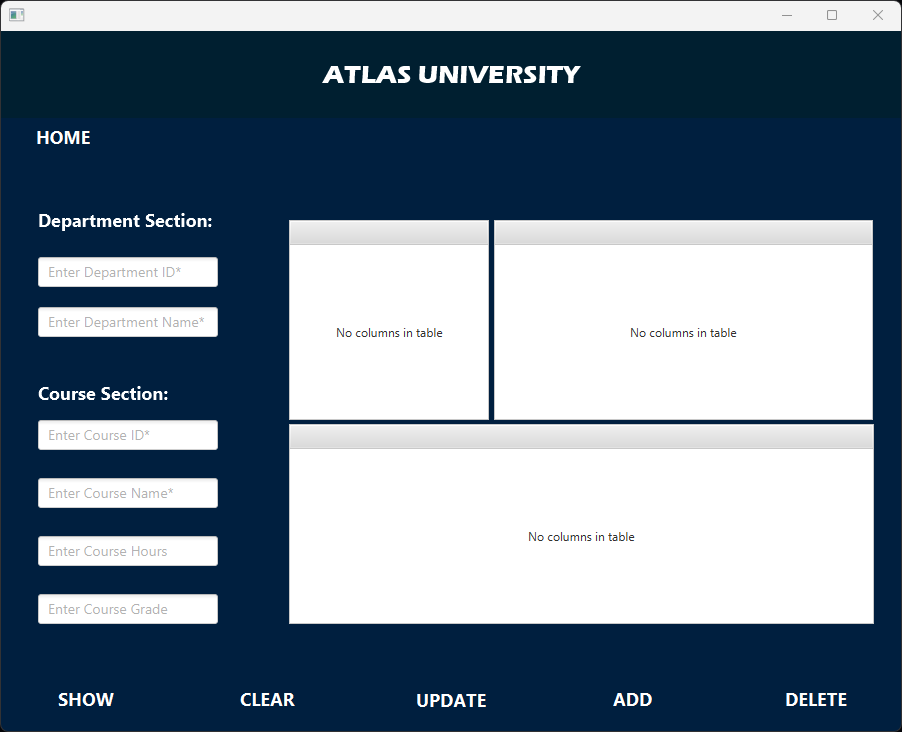


On here you can show a student by entering the student ID, which is unique.

For the GPA calculation as it shows, you will need to enter the semester of the course and the year as well. And you can always go to main page with Home button.

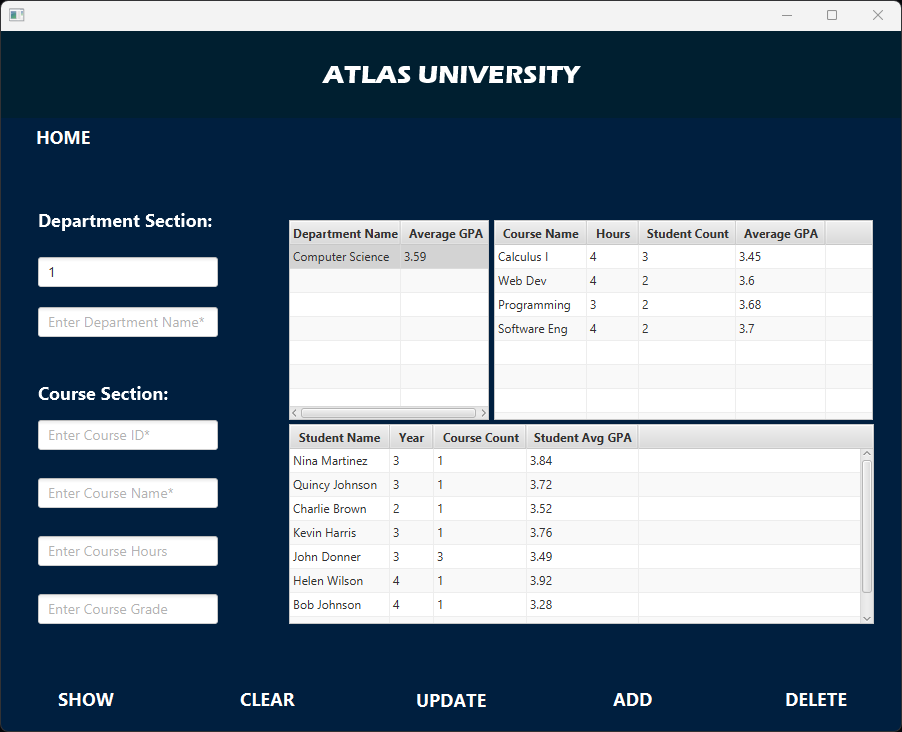


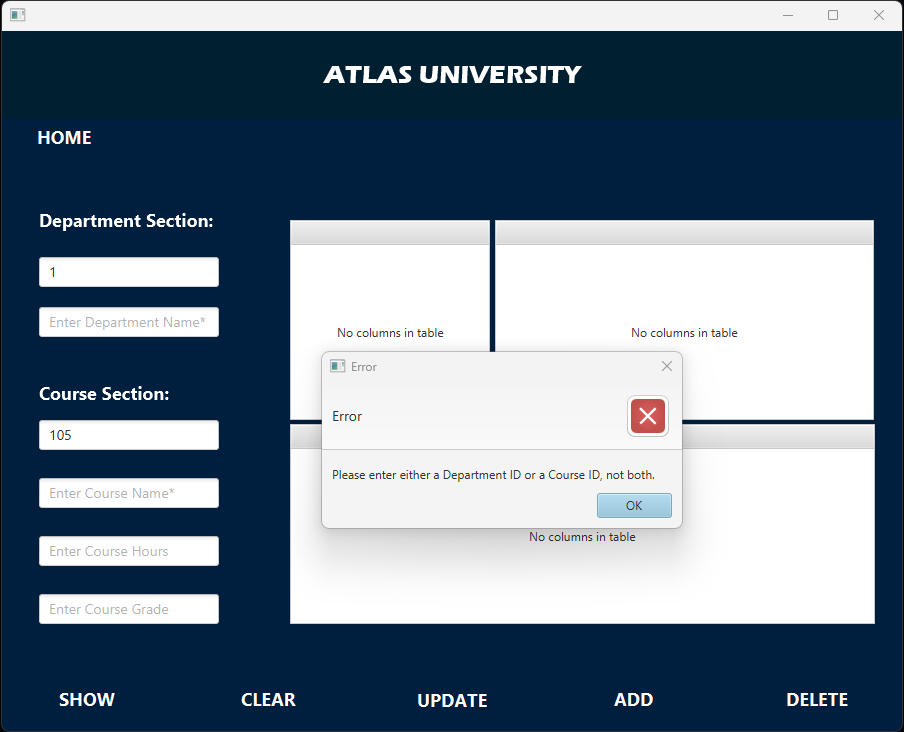
1. Management Page

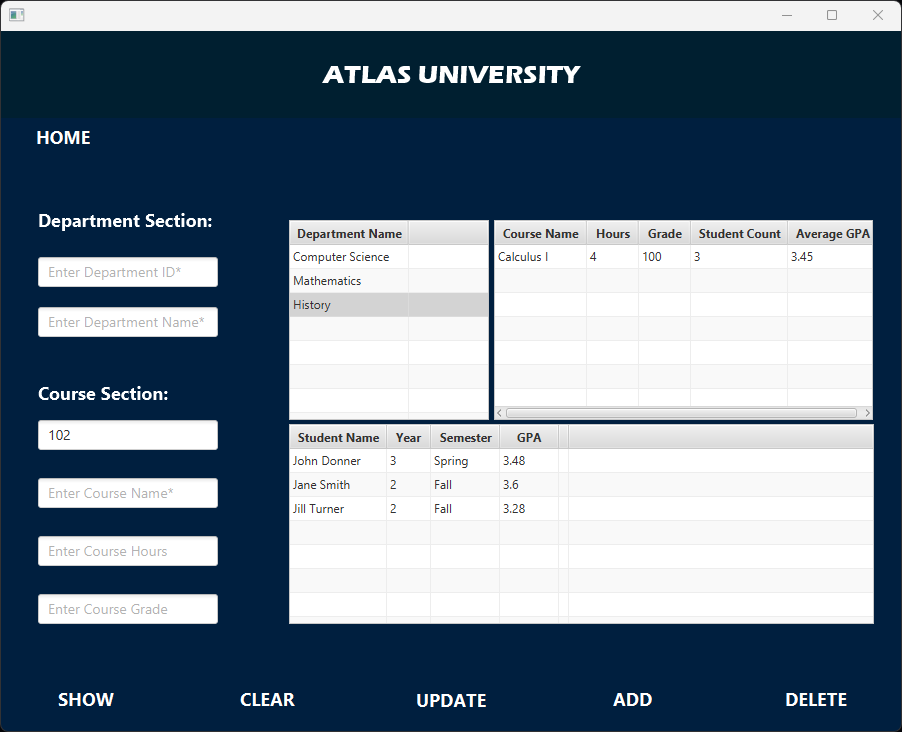


On the management page, you will need to enter either the Department ID or Course ID to retrieve the info of either of them. It will show an error message if both are filler or both are empty.

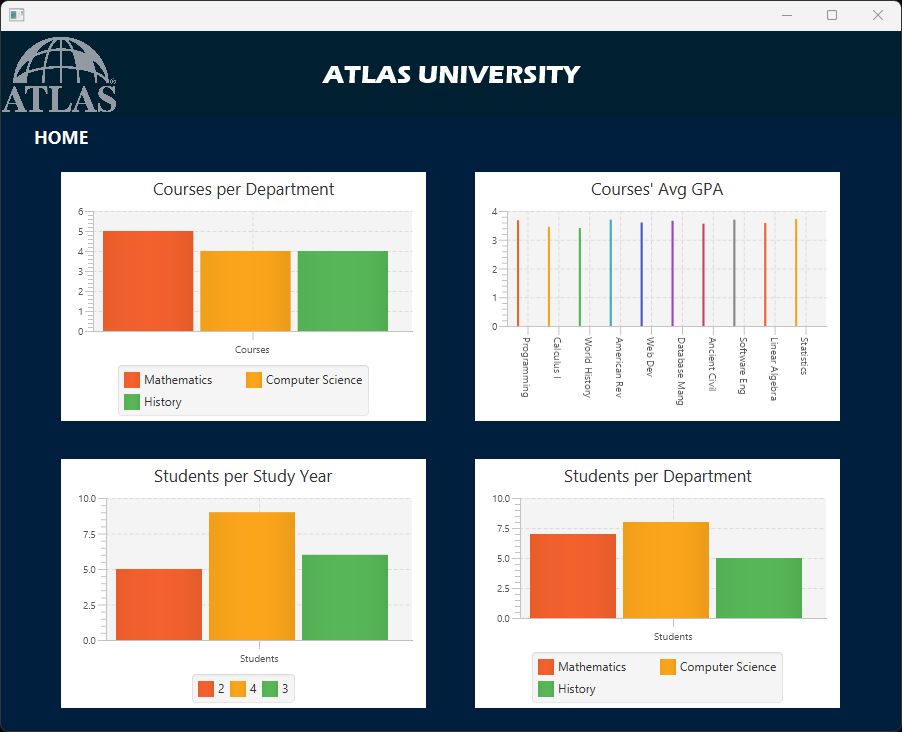
Clicking on the table view when a department is chosen, you will be able to retrieve the course and student info (On mouse click feature).







1. Analysis Page



On the Analysis page you will be able to see 4 different relationships:

* Courses per Department: shows how many student is assigned in each department.
* Courses’ average GPA: shows the average GPA of the students for a specific course.
* Students per Study Year: shows the number of students in each study year group.
* Students per Department: shows the number of students on each department.

Conclusion

In conclusion, the comprehensive case study on university database design and management has provided a thorough exploration of various aspects, ranging from conceptualization and normalization to SQL implementation, PL/SQL procedures, and Java application development. This journey has equipped you with the skills and understanding needed to design, implement, and build a robust university database system.

Embarking on this case study has allowed you to grasp fundamental concepts such as relational database schema design, entity-relationship diagrams, normalization, and the importance of database design documentation. The SQL implementation, including the creation of tables and population with sample data, demonstrated the practical application of theoretical concepts.

The discussion on PL/SQL procedures and functions further enriched your knowledge by covering essential operations like updating student information, calculating GPA, updating department and course information, and handling deletions. This segment showcased the power and versatility of PL/SQL in managing and manipulating data within an Oracle database.

The automation scripts, implemented in Bash, added a layer of operational efficiency by introducing backup procedures and disk space monitoring. Scheduling these scripts using Windows Scheduler ensures the regular execution of critical tasks, enhancing the system's reliability and data integrity.

The Java application development section presented a robust Database Manager class, showcasing CRUD operations, information retrieval, and various functionalities related to students, departments, courses, and GPA calculations. This section encapsulated the integration of database operations within a Java application, emphasizing modularity and adherence to OOP principles.

Lastly, the inclusion of automation scripts and the Java application acknowledges the importance of real-world applicability and operational efficiency in database systems.

This accomplishment is a testament to dedication and collaborative effort in crafting a holistic case study. Appreciation is extended to those who contributed, guided, and supported me throughout this learning journey. As I continue to explore and apply database management concepts, may this case study serve as a valuable reference and foundation for my future endeavors!