

# **IT214 Database Management System**

## **2021**

# **E-Learning Management System**

## **Section 7, Team 5**

201901206 Sparsh Ghelani

201901225 Hastin Dave

201901264 Rishi Kataria

201901462 Dhaval Vaidya

**Mentor - Manoj Kumar Sir**



<u>Section</u>	<u>Page number</u>
Software Requirements Specifications (SRS)	3
Noun Analysis	23
Entity-Relationship Model (ER) Diagrams	30
Conversion of ER Diagram to Relational Model	33
Normalization and Schema Refinement	35
SQL	38
Front-End Implementation	71

**SECTION 1**

**Software Requirements Specifications**

**(SRS)**

## **INTRODUCTION**

### **Purpose**

The purpose of this project is to provide an efficient solution for maintaining and managing the database of students, faculty, administrators of an educational institution in a well-organized manner. The project will also consider the specific requirements of higher management authorities and provide an efficient method resulting in the smooth functioning of the educational institution.

### **Intended Audience**

For this project, the intended audience is

- Students - Students will have viewing access to only certain sections of the database.
- Professors, Teaching Assistants (TA), assistant faculty - They would have edit access to the academic section of the student database.
- Administrators and Other Higher Management Authorities - They would have edit access to the entire database.

### **Reading Suggestions**

The students and staff of the institute should go through the description and certain user classes and characteristics of the product. The administrators and higher authorities of the educational institute or organization should consider reading the full report, especially the fact-finding chart, requirements, product scope, product functions, and business constraints. The tester who will test the product should read the entire report properly, especially the functionalities offered by the product and their working. This will provide him/her with a clear idea of the functioning of the product and can provide some constructive feedback.

### **Product Scope**

This product has a huge potential of helping the management of the institution, schools, coaching institutes and also facilitating the people with various services within an institution. It will help eliminate the paperwork involved. Data would be secured and privacy would be maintained.

### **Description**

- This project will
  - ❖ Allow professors to grade students easily based on the marking criteria.
  - ❖ Keep track of the students' grades.
- However, the fields for a particular person would only depend on the position of the person serving within the institution. For, eg. a student can both be a TA and a student. In that case, the same person will be included in the database two times with 2 different roles and both roles would have a different set of fields.
- With proper documentation of all the roles, we would be able to overcome ambiguity in the data.
- This project is valid for all types of educational institutions.

## **FACT-FINDING PHASE**

### **Background Readings**

- **Learning Management Systems: An Overview**

[https://www.researchgate.net/publication/335463920 Learning Management Systems An Overview](https://www.researchgate.net/publication/335463920_Learning_Management_Systems_An_Overview)

Learning management systems (LMS) can be defined as web-based software platforms which provide an interactive online learning environment and automate the administration, organization as well as reporting of educational content and learner outcomes.

The LMS needs to be capable of executing a variety of functions that work together to provide a seamless experience for the user. The functions are as follows:

#### **Gradebook:**

Such functions include individual scores of assessments, instructor feedback, and student attendance. The ability to generate aggregate reporting information such as class grades, item score analysis, and some student academic information is included in this category.

#### **Social Connectivity:**

Social Connectivity is one of the great criticisms of LMSs is the lack of inherent community in online learning. Features that try to replicate a social environment online include discussion forums, live chats, and videoconference tools as discussed in the previous section.

#### **Security and Privacy:**

Important security features in LMSs include user authentication, access verification, password integrity controls, and intruder detection. Privacy controls are also important to ensure that sensitive information is made available to the intended recipient only.

#### **Ubiquitous Access:**

People are increasingly dependent on their mobile phones to connect to the Internet. It, therefore, stands to reason that online course participants need to interact with LMS course environments using their mobile devices.

#### **Cloud-Like Functionality:**

There is already a trend toward cloud-hosted proprietary LMSs as vendors target their offerings to client organizations that do not have the infrastructure or personnel to manage in-house hosting.

- **The use of distributed databases in E-Learning Systems**

<https://core.ac.uk/download/pdf/81996354.pdf>

E-learning courses include both content (that is, information) and instructional methods (that is, techniques) that help people learn the content. The concept of e-learning provides several advantages to educational organizations which use this technology, including short and effective training, flexibility, and modularization. The disadvantages of e-learning education are the high drop-out rate of students, working in small groups, high costs for design and maintenance. Further, the reading describes Distributed Database Systems in detail and their design, advantages, and disadvantages.

- **New E-Learning method using databases**

[https://www.dbjournal.ro/archive/9/9\\_4.pdf](https://www.dbjournal.ro/archive/9/9_4.pdf)

**Components:**

The integrating IT system contains one server, specialized software based on AJAX, PHP, and MySQL databases, and a broadband connection.

**CBCL:**

This method of direct transfer is the most accurate; it reflects Computer-Based Learning Systems (CBL). Blogs, wiki, Google Docs are usually used in the CBCL environment in the teaching community.

**Enriched Technology Learning (TEL):**

The enriched technology learning (TEL) has the objective to offer innovations, improvement of efficiency of costs) for practices of e-learning, about the physics persons and organizations, independently of time, place, and pace.

- **Requirements gathered:**

- ❖ Our system would provide better functionalities at a more reasonable price.
- ❖ We would also take into consideration students' requirements.
- ❖ DDBs indicate that a database is functionally present everywhere but physically distributed. Our system would eliminate this physical limitation. This would not reduce the reliability of the database and also we would not require multiple servers storing different data.
- ❖ Our system would have an improved design architecture.
- ❖ Our system would not face allocation problems on cost, resources, and performance.
- ❖ It would solve the query-optimization problem.
- ❖ The aim is the creation of an educational network based on e-learning tools that allow greater flexibility in the training of persons in terms of efficiency under national standards.
- ❖ E-learning does not want to replace the traditional educational systems, but to reinforce the learning process.
- ❖ We have included the information of students' grades etc. in the database.
- ❖ To ensure connectivity between professors and students, the email IDs of professors and some information will be available to the students so that they can contact the professor whenever necessary and they don't miss out on something.
- ❖ Certain entities will be restricted to some users to ensure security and to make sure that private information is not provided to all users.
- ❖ It will depend on the developer to provide cloud-like functionality or another method to store the database.

## Interviews

3 interviews were held for developing an E-learning management system, and all were held with the Dean, Administrative staff, and professors respectively.

### **(Role Play) Interview Plan for Dean**

**System:** E-learning Management system

**Project Reference:** SF/SJ/2021/9

#### **Interviewee:**

Mr. Deepak Ahuja(Role Play)

**Designation:** Dean

**Contact Details:** [dean\\_EIT@eit.ac.in](mailto:dean_EIT@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

#### **Interviewers:**

Name	Designation
Rishi Kataria	CEO
Hastin Dave	Administrative Officer

**Date:** 7/10/2021

**Time:** 17:30

**Duration:** 1.5 hours

**Place:** Webex meeting

#### **Purpose of the Interview:**

Discussion regarding the hierarchy of the overall system and a general understanding of the complete functioning of the institute.

#### **Agendas for the Interview:**

- Gaining an idea about the overall budget of the system.
- Other implementation constraints and security measures.
- Features expected in the database.

#### **Documents to be brought in the Interview:**

- Rough plan regarding database design.
- Other relevant documents.

### **(Role Play) Interview plan for Dean**

**System:** E-learning Management system

**Project reference:** SF/SJ/2021/9

#### **Interviewee:**

Mr. Deepak Ahuja(Role Play)

**Designation:** Dean

**Contact Details:** [dean\\_EIT@eit.ac.in](mailto:dean_EIT@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

#### **Interviewers:**

Name	Designation
Rishi Kataria	CEO
Hastin Dave	Administrative Officer

**Date:** 7/10/2021

**Time:** 17:30

**Duration:** 1.5 hours

**Place:** Webex meeting

#### **Results from the Interview:**

- Data security is a concern to be solved.
- Staff shouldn't find it difficult to use.
- The institution is willing to allow a fair budget for the effective implementation of the E-learning database.
- A simple User Interface should be developed which is user-friendly and easy to use for students and faculty staff.

## **(Role Play) Interview plan for Administrative Staff**

**System:** E-learning Management system

**Project reference:** SF/SJ/2021/9

### **Interviewees:**

- |                      |   |
|----------------------|---|
| 1) Mr. Deepak Kabra  | <b>Designation:</b> Head, Research Department |
| 2) Mr. Rahul Mankad  | <b>Designation:</b> Head, UG committee        |
| 3) Ms. Shivani Joshi | <b>Designation:</b> Head, PG committee        |
| 4) Mr. Amit Joshi    | <b>Designation:</b> Accounting Head           |
| 5) Mr. Kanan Sharma  | <b>Designation:</b> Registrar                 |
| 6) Mr. Kiran Kataria | <b>Designation:</b> Warden(Men's Hostel)      |

**Contact Details:** [deepak\\_kabra@eit.ac.in](mailto:deepak_kabra@eit.ac.in)

[rahul\\_mankad@eit.ac.in](mailto:rahul_mankad@eit.ac.in)

[shivani\\_vaidya@eit.ac.in](mailto:shivani_vaidya@eit.ac.in)

[amit\\_joshi@eit.ac.in](mailto:amit_joshi@eit.ac.in)

[kanan\\_sharma@eit.ac.in](mailto:kanan_sharma@eit.ac.in)

[kiran\\_kataria@eit.ac.in](mailto:kiran_kataria@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

### **Interviewers:**

Name	Designation
Sparsh Ghelani	Technical Lead
Dhaval Vaidya	Sr. Software Developer

**Date:** 8/10/2021

**Time:** 17:30

**Duration:** 3 hours

**Place:** Webex meeting

### **Purpose of the Interview:**

To set up a hierarchy of data from an administrative point of view and inclusion of required features for better management.

### **Agendas for the Interview:**

- Gaining an idea about the user classes and characteristics to include as per requirements.
- Discuss adding a few additional features in the database like students' placement status, hostel allocation info, and registration through this system.

### **Documents to be brought in the Interview:**

- Rough plan regarding the organizational structure of requirements.
- Other relevant documents.

## **(Role Play) Interview plan for Administrative Staff**

**System:** E-learning Management system

**Project reference:** SF/SJ/2021/9

### **Interviewees:**

- |                       |   |
|-----------------------|---|
| 1. Mr. Deepak Kabra   | <b>Designation:</b> Head, Research Department |
| 2. Mr. Rahul Mankad   | <b>Designation:</b> Head, UG committee        |
| 3. Ms. Shivani Vaidya | <b>Designation:</b> Head, PG committee        |
| 4. Mr. Amit Joshi     | <b>Designation:</b> Accounting Head           |
| 5. Mr. Kanan Sharma   | <b>Designation:</b> Registrar                 |
| 6. Mr. Kiran Kataria  | <b>Designation:</b> Warden(Men's Hostel)      |

**Contact Details:** [deepak\\_kabra@eit.ac.in](mailto:deepak_kabra@eit.ac.in)

[rahul\\_mankad@eit.ac.in](mailto:rahul_mankad@eit.ac.in)

[shivani\\_vaidya@eit.ac.in](mailto:shivani_vaidya@eit.ac.in)

[amit\\_joshi@eit.ac.in](mailto:amit_joshi@eit.ac.in)

[kanan\\_sharma@eit.ac.in](mailto:kanan_sharma@eit.ac.in)

[kiran\\_kataria@eit.ac.in](mailto:kiran_kataria@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

### **Interviewers:**

Name	Designation
Sparsh Ghelani	Technical Lead
Dhaval Vaidya	Sr. Software Developer

**Date:** 8/10/2021

**Time:** 17:30

**Duration:** 3 hours

**Place:** Webex meeting

### **Results from the Interview:**

- Inclusion of entities representing details of Instructors, Teaching Assistants, and Administrative Staff.

## **(Role Play) Interview plan for Professors**

**System:** E-learning Management system

**Project reference:** SF/SJ/2021/9

### **Interviewees:**

- |                    |   |
|--------------------|---|
| 1) Dr. Milan Kabra | <b>Designation:</b> Sr. professor(DIP)          |
| 2) Dr. Rahul Patel | <b>Designation:</b> Professor(Machine Learning) |
| 3) Dr. Ketul Joshi | <b>Designation:</b> Professor(Embedded Design)  |

**Contact Details:** [milan\\_kabra@eit.ac.in](mailto:milan_kabra@eit.ac.in)

[rahul\\_patel@eit.ac.in](mailto:rahul_patel@eit.ac.in)

[ketul\\_doshi@eit.ac.in](mailto:ketul_doshi@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

### **Interviewers:**

Name	Designation
Sparsh Ghelani	Technical Lead
Dhaval Vaidya	Sr. Software Developer

**Date:** 9/10/2021

**Time:** 17:30

**Duration:** 1 hour

**Place:** Webex meeting

### **Purpose of the Interview:**

To set up the hierarchy of data from professors' point of view.

### **Agendas of the Interview:**

- Gaining an idea about the important fields fulfilling professor requirements.
- The concern of students regarding meeting links.
- Inclusion of grade class and required features.
- Teaching Assistant database creation.

### **Documents to be brought in the Interview:**

- Rough plan regarding the organizational structure of requirements.
- Other relevant documents.

### **(Role Play) Interview plan for Professors**

**System:** E-learning Management system

**Project reference:** SF/SJ/2021/9

#### **Interviewees:**

- |                    |   |
|--------------------|---|
| 1. Dr. Milan Kabra | <b>Designation:</b> Sr. professor(DIP)          |
| 2. Dr. Rahul Patel | <b>Designation:</b> Professor(Machine Learning) |
| 3. Dr. Ketul Joshi | <b>Designation:</b> Professor(Embedded Design)  |

**Contact Details:** [milan\\_kabra@eit.ac.in](mailto:milan_kabra@eit.ac.in)

[rahul\\_patel@eit.ac.in](mailto:rahul_patel@eit.ac.in)

[ketul\\_doshi@eit.ac.in](mailto:ketul_doshi@eit.ac.in)

**Organization Details:** Engineering Institute of Technology

#### **Interviewers:**

Name	Designation
Sparsh Ghelani	Technical Lead
Dhaval Vaidya	Sr. Software Developer

**Date:** 9/10/2021

**Time:** 17:30

**Duration:** 1 hour

**Place:** Webex meeting

#### **Results from the interview:**

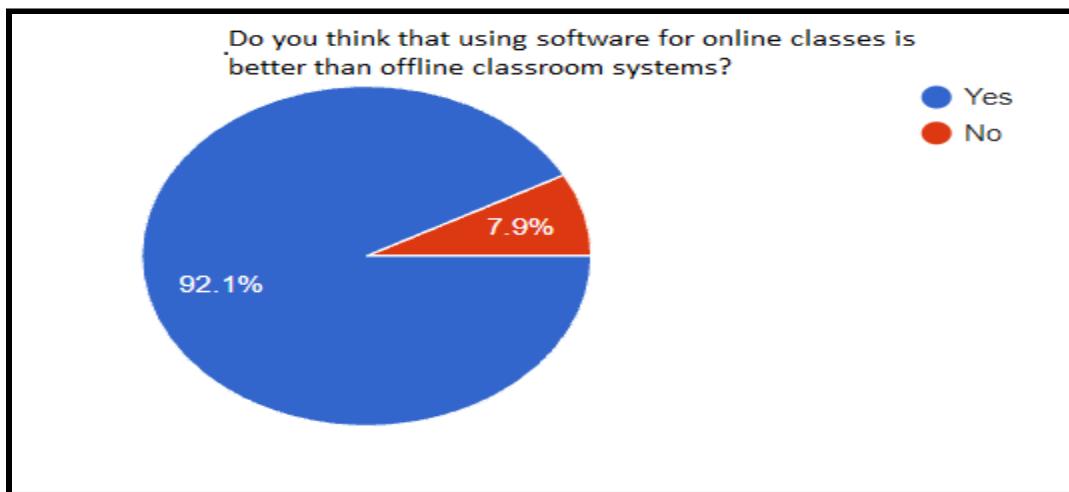
- Inclusion of an entity that would help maintain the meeting links for the online classes and eliminate the need for creating a different link every time. This would also eliminate the hassle of searching the link every time for students.
- Inclusion of grade cards for students and course details like credit structure.
- Inclusion of an entity, providing details about the Teaching Assistants.
- It was also mentioned that professors do not require all of the students' details and therefore, there should be a different section where only the academic details of students are visible.

## Questionnaire

1. **Do you think that software-oriented online classes are better than offline classroom systems?**  
Yes/No
2. **Which software is your institute currently using for online classes?**  
Cisco Webex/Zoom Meetings/Microsoft Teams/Google Meet
3. **Do you think that the current system can be improved?**  
Yes/No
4. **Do you think it would be better if you have data of all the Professors/Teaching Assistants(TAs)/Students with you?**  
Yes/No
5. **If you are running a coaching institute, do you think it will be helpful to you if you have a list of students and their details?**  
Yes/No
6. **Will it be better if you can track individual students' academics?**  
Yes/No
7. **Do you have other suggestions that an E-learning management system can include?**
8. **Will you buy software that has all the above functionalities and is available at a reasonable price?**  
Yes/No/Maybe

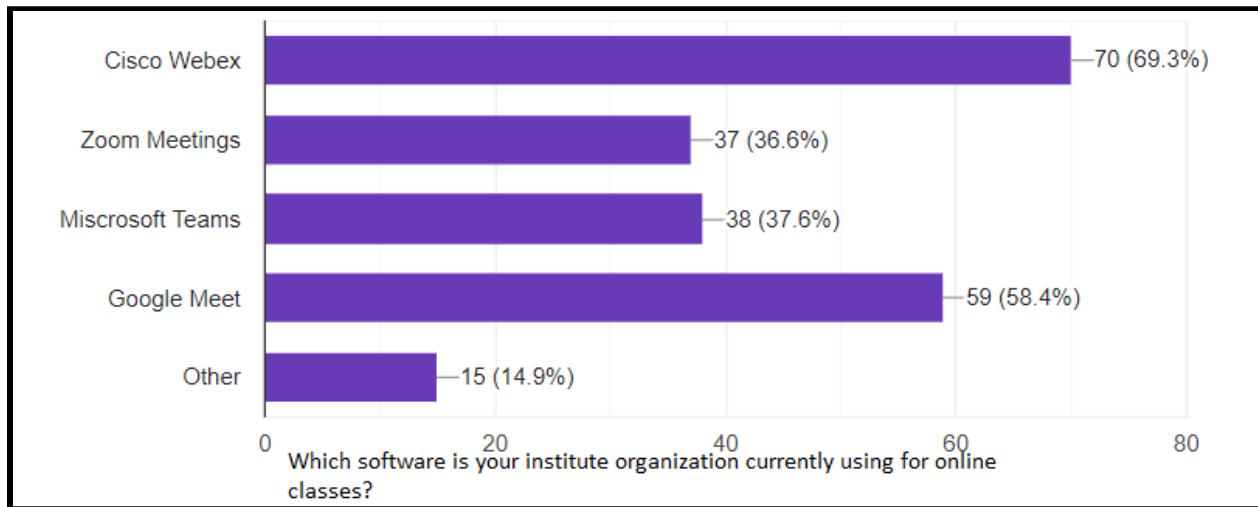
## Summary of Responses

1. **Do you think software-oriented online classes are better than offline classroom systems?**  
92.1% of the people responded with Yes since they believe online classes are better than offline classroom systems.



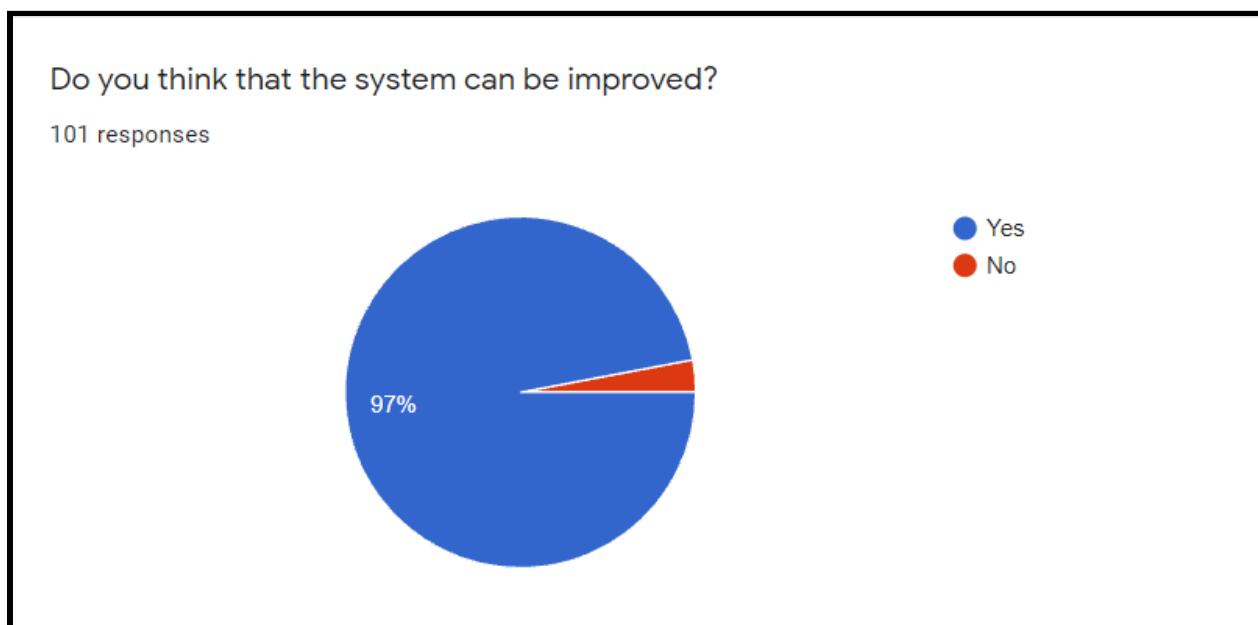
**2. Which software is your institute currently using for online classes?**

As a response to this question, we could conclude that Cisco Webex and google meet are widely used. As a third and fourth choice, zoom meet and Microsoft meet were pretty popular as well. Some organizations also used other Softwares. The detailed distribution is as follows:



**3. Do you think that the current system can be improved?**

Most of the people answered with Yes since it is a natural tendency for people to expect a better user experience with the advancement of new technologies.

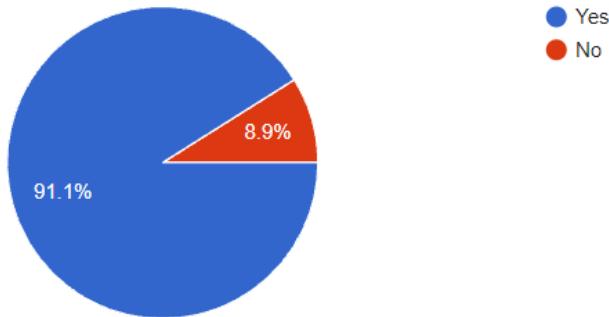


**4. Do you think it would be better if you have data of all the Professors/Teaching Assistants(TAs)/Students with you?**

Most of the institutions responded with Yes since it is always helpful to have as much data as possible.

Do you think it will be better if you have the database of all the Teaching Assistants, students and professors with you?

101 responses

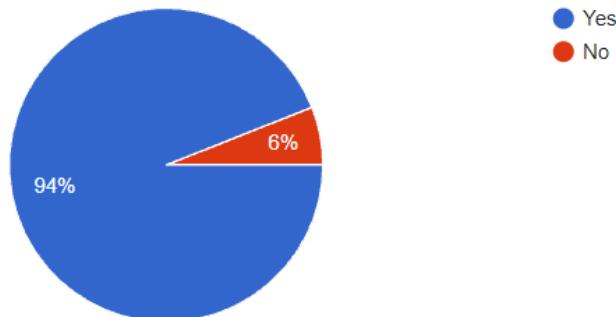


**5. If you are running a coaching institute, do you think it will be helpful to you if you have a list of students and their details?**

Most of the coaching institutions responded with Yes since again it is helpful to have as much data as possible.

If you are running a coaching institute, do you think it will be helpful to you if you have a list of students and their details (eg, Fees paid or not)

100 responses

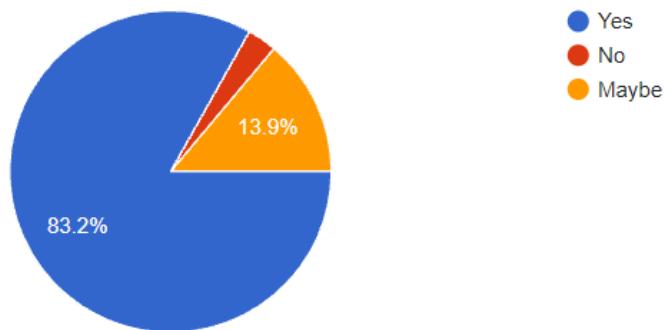


**6. Will it be better if you can track individual students' academics?**

It would be better if you can track individual performance as it would also be better for students as the professors would be able to concentrate on individuals.

Will it be better if you can track individual students' academic and extra-curricular performance?

101 responses



**7. Do you have other suggestions that an E-learning management system can include?**

Do you have any other suggestions that an E-learning Management System can include?

9 responses

Fair examinations

No

laszmlmsopm

Can you add functionality named Online registration ?

Can you implement plagiarism checker?

Students should be able to have a list of links

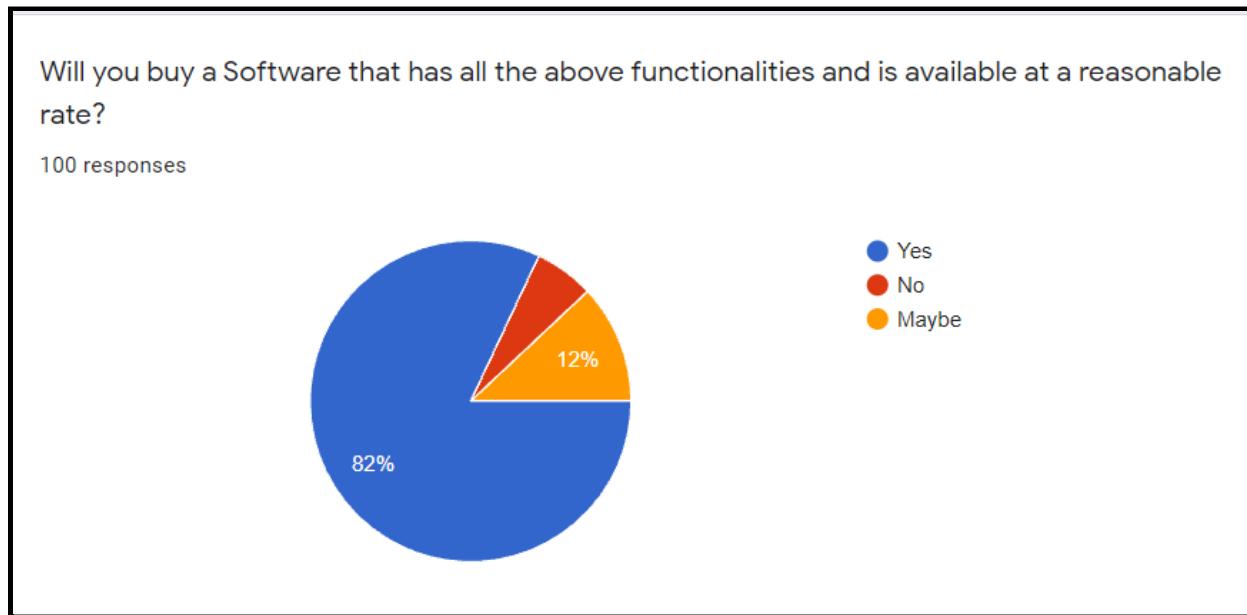
None

Can you add some more functionalities like grade calculator,etc?

Can you implement system to store all our link for lectures and materials?

**8. Will you buy software that has all the above functionalities and is available at a reasonable price?**

Most of the institutions responded with a Yes.



## **Observations**

**System:** E-learning Management system

**Project Reference:** SF/SJ/2021/9

**Observers:**

Name	Designation
Rishi Kataria	CEO
Hastin Dave	Administrative Officer

**Date:** 11/10/2021

**Time:** 17:30

**Duration:** 3 hours

**Place:** Webex meeting

**Observations:**

- Educational Institutions require a fully working efficient database at a reasonable price.
- Professors find it difficult to manually grade students on basis of their performance, hence they require a feature that can filter out students based on the instructor's score expectancy and get graded accordingly.
- Students require a platform where they can daily access the lecture links of respective courses.

## **FACT-FINDING CHART**

<b>Objective</b>	<b>Technique</b>	<b>Subject(s)</b>	<b>Time Commitment</b>
To get information about University/Coaching classes	Background reading	Wikipedia, Blogspot, Quora, etc	1-hour x 2 days
Discussion regarding the hierarchy of the overall system and a general understanding of the complete functioning of the institute	Interview	Dean	1.5 hours
To set up a hierarchy of data from an administrative point of view and inclusion of required features for better management	Interview	Administrative Staff	3 hours
To set up a hierarchy of data from instructors' point of view	Interview	Professors	1-hour
Find the problems in the current system that needed to be addressed	Observation	-----	3 hours
Getting help with the organization of data and its characteristics	Document Sampling	Administrative Staff	2 hours

## **REQUIREMENTS**

- A well-maintained database of students.
- A good, compact, and highly efficient user interface is expected at every level of the system.
- The database should be implemented with a required level of abstraction since all the people should not be able to view every part of the database.
- A separate entity of lecture links needs to be created which keeps changing its date and timings as and when the next class of a particular course is held.
- Division of Student specific entities required for better communication of details with management and academics.
- A separate entity of Teaching Assistant is required, with his/her academic contact details which remain easily accessible to course instructors of a particular course and students pursuing that course.

## **USER CLASSES AND CHARACTERISTICS**

The prominent user classes here are as follows:

- Students
- Teaching Assistants
- Instructors
- Administrative Officers

The users can be classified according to their frequency of use. A subset of product functions used technical expertise, security or privilege levels, educational level, or experience they have to the E-learning Management System.

### **1. Student Details**

**Attributes** - Student\_ID, Student\_Name, Email\_Address, Current\_Semester, Current\_CPI

### **2. Instructor Details**

**Attributes** - Instructor\_ID, Instructor\_Name, Email\_Address

### **3. Admin Details**

**Attributes** - Admin\_Name, Email\_Address, Department

### **4. Course Details**

**Attributes** - Course\_ID, Course\_Name, Credits

### **5. Teaching Assistant Details**

**Attributes** - Student\_ID, Course\_ID

### **6. Course-Instructor Relationship**

**Attributes** - Instructor\_ID, Course\_ID

### **7. Lecture/Lab Delivery Details**

**Attributes** - Course\_ID, Lecture\_Day, Lecture\_Link

### **8. Grading Details**

**Attributes** - Student\_ID, Course\_ID, Grade

## **OPERATING ENVIRONMENT**

### **Hardware Requirements**

- The software should run on any sort of desktop or laptop environment with utmost ease.
- A simplified version of the software would also be available on mobile and tablet devices.
- The basic hardware interface devices required are a display, keyboard and mouse (For PC version), and a smartphone (For mobile version).

### **Software Requirements**

- To store the entire data, a server would require at least 500 Gigabytes of dedicated memory.
- A RAM of at least 32 Gigabytes for fast database operations is required.
- Backup Plan - In case of shutdown or a system failure, data backup would help resurrect the database. Therefore, time-to-time data backup would help with the cause.

### **Connectivity Requirements**

- For data synchronization, we have two options. Either integrate our database into the cloud database system or connect all database servers of each branch through physical links.
- A centralized database system would be required at both inter and intra levels for synchronizing data.
- A WiFi link with a connection speed of at least 40 Mbps would be required for smooth transactions.

### **External Interface Requirements**

- **Reliability**
  - ❖ The system should be available 24x7.
  - ❖ The system failure should not last more than 12 hours.
  - ❖ Other software failures in the system should be reported and repaired in under 3 hours.
  - ❖ The system should be able to handle all types of data types.
- **Performance**
  - ❖ System response to an access or edit request should be within 5 seconds.
  - ❖ The system can handle up to 1000 concurrent requests at a given point in time.
- **Supportability**
  - ❖ The system shall accommodate changes and enhancements that the development team is going to make within 5 days.
  - ❖ Easy management of authorization access is required.
  - ❖ A user-friendly, compact and simple user interface is required at every level of data access in the database.
- **Design Constraints**
  - ❖ The system is designed according to the IEEE standards.
  - ❖ The system should be capable enough to replace the current existing system.

## **PRODUCT FUNCTIONS AND PRIVILEGES**

### **1. Login**

**Accessible by** - All User Classes

### **2. Logout**

**Accessible by** - All User Classes

### **3. Student Personal and Registration details**

**View Access** - Students and Administrative Officers

**Edit Access** - Students and Administrative Officers (Registrar)

### **4. Student Academic details**

**View Access** - Students, Instructors, and Administrative Officers

**Edit Access** - Administrative Officers (Academic Officer)

### **5. Teaching Assistants', Instructors' and Administrative Officers' details**

**View Access** - Instructors and Administrative Officers

**Edit Access** - Instructors and Administrative Officers

### **6. Course Details**

**View Access** - All User Classes

**Edit Access** - Administrative Officers (Academic Officer)

### **7. Course-Instructor Details**

**View Access** - All User Classes

**Edit Access** - Instructors and Administrative Officers (Academic Officer)

### **8. Lecture/Lab Delivery Details**

**View Access** - All User Classes

**Edit Access** - Teaching Assistants, Instructors, and Administrative Officers (Academic Officer)

### **9. Students' Grades' Details**

**View Access** - Students, and Teaching Assistants and Instructors

**Edit Access** - Instructors and Administrative Officers (Academic Officer)

## **ASSUMPTIONS AND DEPENDENCIES**

- A person cannot serve more than one post in an administrative department. For eg: A person cannot be an accounting officer and an Academic officer at the same time.
- Every student gets graded on the same basis. For eg- Every student will be graded on a scale of 10.
- A professor cannot teach more than 1 course in a single semester.

## **BUSINESS CONSTRAINTS**

- The Institution hasn't specified any money-related or space-related constraint. Although, the only constraint mentioned was a Proper User Accessible database.
- For centralizing the database, the physical connection was checked out and cloud integration was recommended with minimal cost to the Institution.

**SECTION 2**  
**Noun Analysis**

## **DESCRIPTION**

- This project deals with the storage and retrieval of student details and management of other information included in the E-learning management systems.
- This platform will contain a database of Students, Teaching Assistants, Instructors, Admins, and Courses related to each other by different relationships.
- A student should have a **Student ID**, Name, and an Email Address.
- Also, it will store every student's current CPI.
- The **Teaching Assistant database** will have fields like **Student ID** and the Course ID, which they are currently assisting.
- However, the fields for a particular person would only depend on his role.  
For example, a student can be both a TA and a student.  
In that case, the same person will be included in the database two times with two different roles, and both positions would have a different set of fields.
- The **Instructor database** will contain an **Instructor ID** that will uniquely identify an instructor.
- It would also have other details like name and Email address.
- The relationship between instructor and course entities will contain uniquely identifiable **Instructor ID** and **Course ID** as its fields.
- There will be a separate entity named **Admin Details** that would contain the names, **Email addresses**, and the department of the administrators.
- The **Grading Details database** contains the Student ID, Course ID and their respective grades.
- This database will allow professors to keep academic records of students.
- The instructor can access the grading details and assign grades to the students enrolled in the course.
- The grade has to be between 3 (minimum grade to pass a course) to 10 (maximum grade a student can get).
- If a student fails in any course, 0 will appear before the corresponding course in the grading table.
- On behalf of special concerns raised by students, a separate entity named **Lecture Delivery Details** is included in the database. It will include the lecture links for the online delivery of the courses.
- With proper documentation of all the roles, we would be able to overcome ambiguity in the data.
- This project is valid for all types of educational institutions.
- The users can be classified according to their frequency of use.

- A subset of product functions used technical expertise, security or privilege levels, educational level, or experience to the E-learning Management System.

Below are the attributes of entity sets of the database and the access specification details of each entity set in the database.

**1. Instructor Details**

**Attributes** - Instructor\_ID, Instructor\_Name, Email\_Address

**2. Admin Details**

**Attributes** - Admin\_Name, Email\_Address, Department

**3. Course Details**

**Attributes** - Course\_ID, Course\_Name, Credits

**4. Teaching Assistant Details**

**Attributes** - Student\_ID, Course\_ID

**5. Course-Instructor Relationship**

**Attributes** - Instructor\_ID, Course\_ID

**6. Lecture/Lab Delivery Details**

**Attributes** - Course\_ID, Lecture\_Day, Lecture\_Link

**7. Grading Details**

**Attributes** - Student\_ID, Course\_ID, Grade

• **Privileges**

**1. Login**

**Accessible by** - All User Classes

**2. Logout**

**Accessible by** - All User Classes

**3. Student Personal and Registration details**

**View Access** - Students and Administrative Officers

**Edit Access** - Students and Administrative Officers (Registrar)

**4. Student Academic details**

**View Access** - Students, Instructors, and Administrative Officers

**Edit Access** - Administrative Officers (Academic Officer)

**5. Teaching Assistants', Instructors' and Administrative Officers' details**

**View Access** - Instructors and Administrative Officers

**Edit Access** - Instructors and Administrative Officers

**6. Course Details**

**View Access** - All User Classes

**Edit Access** - Administrative Officers (Academic Officer)

**7. Course-Instructor Details**

**View Access** - All User Classes

**Edit Access** - Instructors and Administrative Officers (Academic Officer)

**8. Lecture/Lab Delivery Details**

**View Access** - All User Classes

**Edit Access** - Teaching Assistants, Instructors, and Administrative Officers (Academic Officer)

**9. Students' Grades' Details**

**View Access** - Students, and Teaching Assistants and Instructors

**Edit Access** - Instructors and Administrative Officers (Academic Officer)

## **NOUNS (& VERBS) ANALYSIS**

**Table 1**

All extracted nouns and groups

<b><u>Nouns</u></b>	<b><u>Verbs</u></b>
Student	Studies under an Instructor
Student	Studies a Course
Student	Attends Lecture
Student	Registers
Student	Pays
Student	Earns Grade
Student	Interns
Student	Gets Placed
Student	Resides
Instructor	Instructs Students
Instructor	Teaches Course
Instructor	Assigns TA
Instructor	Gives Lecture
Instructor	Grades
Instructor	Gets Paid
Teaching Assistant	Teaches Course
Teaching Assistant	Assists Instructor
Teaching Assistant	Evaluates
Teaching Assistant	Groups Students
Admin	Manages Details
Admin	Designs Course
Admin	Updates Details
Admin	Collects Fees

Course	Is Taught
Hostel	Accommodates

**Table 2**

Accepted Nouns and Verbs list

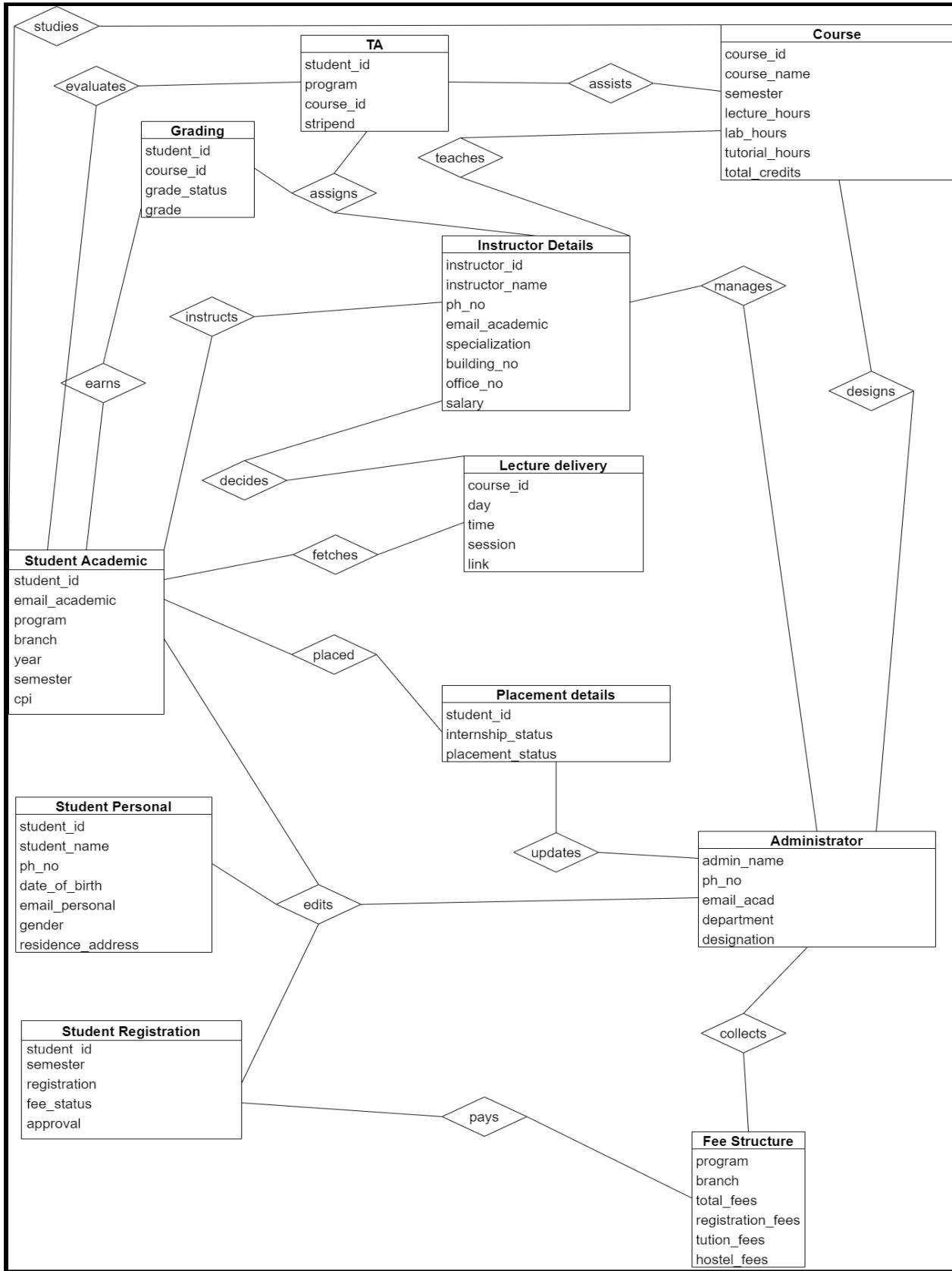
<b>Candidate Entity Set</b>	<b>Candidate Attribute Set</b>	<b>Candidate Relationship Set</b>
Student Details	Student_ID Student_Name Email_Address Current_Semester Current_CPI	Attends Lecture Earns Grade Studies a Course
Teaching Assistant Details	Student_ID Course_ID	Assists an Instructor
Instructor Details	Instructor_ID Instructor_Name Email_Address	Teaches a Course Assign Grade Gives Lecture
Admin Details	Admin_Name Email_Address Department	Manages Instructor Edits Student details Designs Course
Course details	Course_ID Course_Name Credits	
Course-Instructor Relationship	Instructor_ID Course_ID	
Lecture Delivery Details	Course_ID Lecture_Day Lecture_Link	
Grading Details	Student_ID Course_ID Grade	

**Table 3****Rejected Nouns and Verbs List**

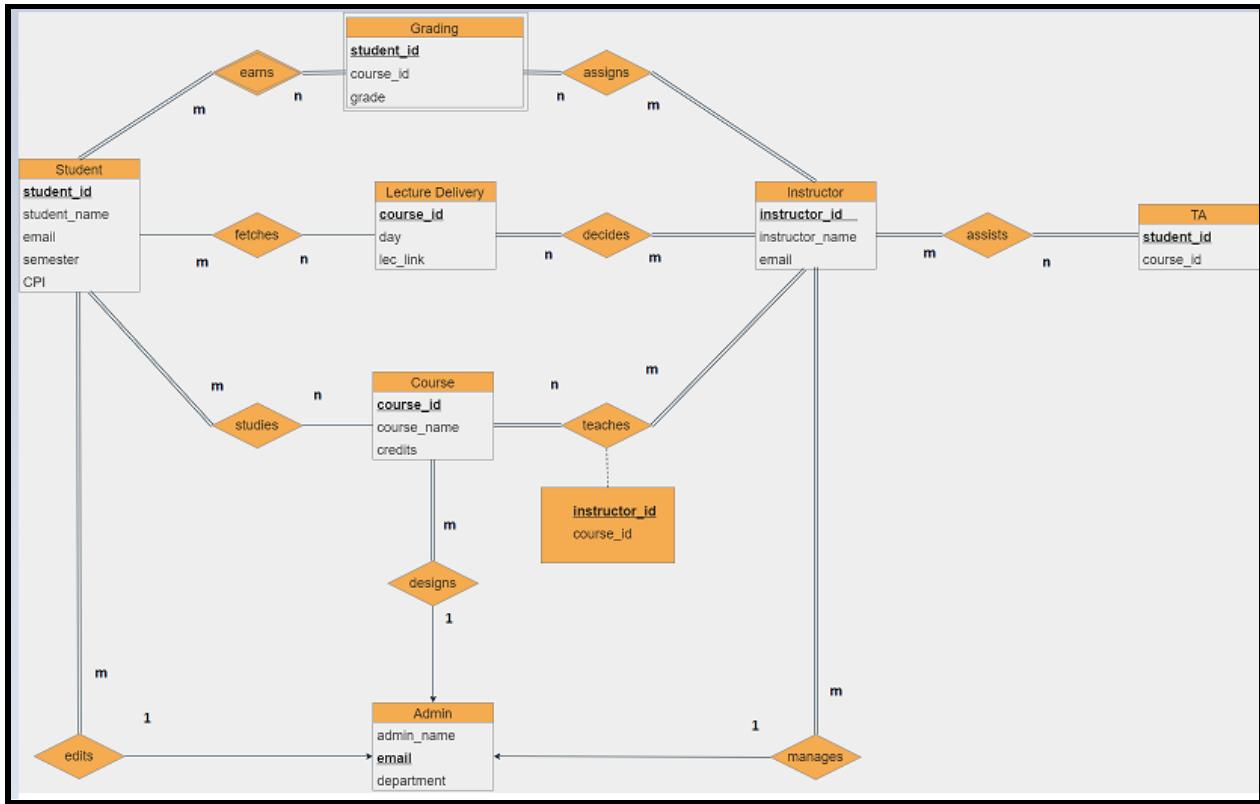
<b><u>Nouns</u></b>	<b><u>Reject Reasons</u></b>
Student (Studies under an Instructor)	Associations
Student (Resides)	Vague
Student (Registers)	Irrelevant
Student (Pays)	Vague
Student (Interns)	Vague
Student (Gets Placed)	Vague
Hostel (Accommodates)	Vague, General
Instructor (Instructs Students)	Associations
Instructor (Assigns TAs)	Irrelevant, Associations
Instructor (Gets paid)	Vague
Teaching Assistant (Groups Students)	Irrelevant
Teaching Assistant (Evaluates)	Duplicates, Irrelevant, Associations
Teaching Assistant (Teaches Course)	Duplicates, General, Associations
Admin (Collects)	Vague
Course (Is Taught)	Vague, General, Attributes

**SECTION 3**  
**Entity Relationship Model (ER)**  
**Diagrams**

## VERSION 1



## VERSION 2



Grading is a weak entity set that is owned by the Student entity set. This is because student and grading share a common attribute namely, Student\_ID. The weak entity set can be represented by a double-lined rectangle and the relationship can be described by a double-lined diamond symbol.

**SECTION 4**

**Conversion of ER-Diagram to**

**Relational Model**

## **FINAL RELATIONS AND ATTRIBUTES**

1. **Student Details** ( Student\_ID, Student\_Name, Email\_Address, Current\_Semester, Current\_CPI )
2. **Teaching Assistant Details** ( Student\_ID, Course\_ID )
3. **Instructor Details** ( Instructor\_ID, Instructor\_Name, Email\_Address )
4. **Admin Details** ( Admin\_Name, Email\_address, Department )
5. **Course Details** ( Course\_ID, Course\_Name, Credits )
6. **Course-Instructor Relationship** ( Instructor\_ID, Course\_ID )
7. **Lecture/Lab Delivery Details** ( Course\_ID, Lecture\_Day, Lecture\_Link )
8. **Grading Details** ( ID, Student\_ID, Course\_ID, Grade )

**SECTION 5**

**Normalization and Schema  
Refinement**

## **Identify and list all types of dependencies ( PK, FK, Functional Dependencies) for each relation**

- **Student Details**

**Primary Key** - Student\_ID, Email\_Address

**Foreign Key** -

**Dependencies** -

Student\_ID → Student\_Name

Student\_ID → Email\_Address

Student\_ID → Current\_Semester

Student\_ID → Current\_CPI

- **Teaching Assistant Details**

**Primary Key** - Student\_ID

**Foreign Key** - Student\_ID, Course\_ID

**Dependencies** -

Student\_ID → Course\_ID

- **Instructor Details**

**Primary Key** - Instructor\_ID, Email\_Address

**Foreign Key** -

**Dependencies** -

Instructor\_ID → Instructor\_Name

Instructor\_ID → Email\_Address

- **Admin Details**

**Primary Key** - Email\_Address

**Foreign Key** -

**Dependencies** -

Email\_Address → Admin\_name

Email\_Address → Department

- **Course Details**

**Primary Key** - Course\_ID, Course\_Name

**Foreign Key** -

**Dependencies** -

Course\_ID → Course\_Name

Course\_ID → Credits

- **Course-Instructor Relationship**

**Primary Key** - Instructor\_ID, Course\_ID

**Foreign Key** - Instructor\_ID, Course\_ID

**Dependencies** -

Instructor\_ID → Course\_ID

Course\_ID → Instructor\_ID

- **Lecture/Lab Delivery Details**  
**Primary Key** - Course\_ID, Lecture\_Link  
**Foreign Key** - Course\_ID  
**Dependencies** -  
Course\_ID → Lecture\_Day  
Course\_ID → Lecture\_Link
- **Grading**  
**Primary Key** - ID  
**Foreign Key** - Student\_ID, Course\_ID  
**Dependencies** -  
ID → Student\_ID  
ID → Course\_ID  
ID → Grade

### **Investigate every schema for the following**

1. **List of redundancies existing for every schema which is part of the database (document it).**  
There were around 80 tuples in the course schema, out of which only 40 tuples were used in other schemas.  
Hence we removed the unnecessary courses not taken by students or not assigned to TA.
2. **List of updates, delete and insert anomalies for every schema (document it).**  
None

### **Normalize the database up to 1NF (scalar values)**

All tables are 1NF. There are no multivalued attributes in any schema. There are no composite attributes in our schemas.

### **Normalize the database further to 2NF (Remove Partial Dependencies)**

Here the database is already in 2 NF because there is no partial dependency existing in the schemas.  
After finding the candidate keys and all proper subsets, they all are not dependent on the Non-Primary Attributes(NPA).

### **Identify (and document) a List of redundancies existing for the schema in 2NF**

As explained above, there are no redundancies present as the schemas are already in 2NF.

### **Normalize it further to 3NF/BCNF (Remove Transitive Dependencies)**

We didn't find any transitive dependencies. So we can say the given schemas are normalized to 3NF.  
For every dependency, we discovered that every determinant in a schema is a super key, so we can also say that it is already normalized to BCNF.

# **SECTION 6**

# **SQL**

## **DDL SCRIPTS**

### **Student Details**

```
CREATE TABLE mydata.Student14
(
    s_id bigint,
    s_name varchar(100),
    s_email varchar(100),
    sem bigint,
    cpi float,
    PRIMARY KEY(s_id)
);
```

### **Course Details**

```
CREATE TABLE mydata.Course14
(
    c_id varchar(10),
    c_name varchar(100),
    credits float,
    PRIMARY KEY(c_id)
);
```

### **Grading Details**

```
CREATE TABLE mydata.Grades14
(
    id bigint,
    s_id bigint,
    c_id varchar(10),
    grade varchar(2),
    FOREIGN KEY(s_id) REFERENCES Student14,
    FOREIGN KEY(c_id) REFERENCES Course14,
    PRIMARY KEY(id)
);
```

### **Lecture Delivery Details**

```
CREATE TABLE mydata.Leclinks14
(
    c_id varchar(10),
    l_day varchar(10),
    l_link varchar(100),
    FOREIGN KEY(c_id) REFERENCES Course14,
    PRIMARY KEY(c_id)
);
```

## **Instructor Details**

```
CREATE TABLE mydata.Instructor14
(
    i_id bigint,
    i_email varchar(100),
    i_name varchar(100),
    PRIMARY KEY(i_id)
);
```

## **Teaching Assistants Details**

```
CREATE TABLE mydata.Ta14
(
    s_id bigint,
    c_id varchar(10),
    FOREIGN KEY(s_id) REFERENCES Student14,
    FOREIGN KEY(c_id) REFERENCES Course14,
    PRIMARY KEY(s_id)
);
```

## **Admin Details**

```
CREATE TABLE mydata.Admin14
(
    a_name varchar(100),
    a_email varchar(100),
    dept varchar(30),
    PRIMARY KEY(a_email)
);
```

## **Course-Instructor Relationship**

```
CREATE TABLE mydata.Courseinstructor14
(
    i_id bigint,
    c_id varchar(10),
    FOREIGN KEY(i_id) REFERENCES Instructor14,
    FOREIGN KEY(c_id) REFERENCES Course14,
    PRIMARY KEY(i_id)
);
```

## **INSERT (IMPORT) STATEMENTS**

```
COPY mydata.Student14 (s_id, s_name, s_email, sem, cpi) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Student14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Instructor14 (i_id, i_name, i_email) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Instructor14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Admin14 (a_name, a_email, dept) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Admin14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Course14 (c_id, c_name, credits) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Course14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Ta14 (s_id, c_id) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL script+schemas\Ta14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Courseinstructor14 (i_id, c_id) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Courseinstructor14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Leclinks14 (c_id, l_day, l_link) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Leclinks14.csv'  
DELIMITER ',' CSV HEADER;
```

```
COPY mydata.Grades14 (id, s_id, c_id, grade) FROM 'E:\D lect\sem-V\IT214\Labs\lab_8\DDL  
script+schemas\Grades14.csv'  
DELIMITER ',' CSV HEADER;
```

## **TRIGGERS AND FUNCTIONS**

### **Function**

#### **Function to calculate average CPI of students in a particular semester**

```
create or replace function semwise_cpi(seme_ster integer)
returns Table(tot_stude bigint,avg_cpi float) as $body$
begin
return query select count(*),avg(cpi) from Student14 where sem = seme_ster;
end
$body$ LANGUAGE plpgsql;
```

### **Triggers**

#### **1. Trigger to check validity of inserted values in the student table**

```
create or replace function student_info()
returns trigger as $body$
begin
if(new.s_id<=0) then
raise exception 'invalid student id entered';
elseif(new.s_email not like '%Educat.ac.in%') then
raise exception 'email ID does not belong of institution';
elseif (new.sem<1 or new.sem>8) then
raise exception 'invalid semester entered';
elseif(new.cpi>10.00 or new.cpi<3.00) then
raise exception 'invalid cpi entered';
else
raise notice 'successfully entered student details';
return new;
end if;
end
$body$ LANGUAGE plpgsql;
```

```
create trigger Trigger_student_info
after Insert or update
on Student14
for each row
execute procedure student_info();
```

**2. Trigger to check validity of inserted values in the course table**

```
create or replace function course_info()
returns trigger as $body$
begin
if(new.c_id='NULL') then
raise exception ' course id cannot be null and cannot be an integer value';
elseif (new.credits<1.5 or new.credits>4.5 ) then
raise exception 'invalid credits of a course entered';
elseif(new.c_name='NULL') then
raise exception 'course name cannot be null ';
else
raise notice 'successfully entered course details';
return new;
end if;
end
$body$ LANGUAGE plpgsql;
```

```
create trigger Trigger_course_info
after Insert or update
on Course14
for each row
execute procedure course_info();
```

**3. Trigger to check validity of inserted values in the grading details table**

```
create or replace function grading_info()
returns trigger as $body$
begin
raise notice 'make sure you write grade between DE to AA. Write inside single quotes ';
if(new.grade='AA' or new.grade='AB' or new.grade='BB' or new.grade='BC' or new.grade='CD' or
new.grade='DD'
or new.grade='CC' or new.grade='DE' or new.grade='F') then
raise notice 'successfully inserted grade details';
return new;
else
raise exception 'invalid grade entered';
end if;
end
$body$ LANGUAGE plpgsql;
```

```
create trigger Trigger_grading_info
after Insert or update
on Grades14
for each row
execute procedure grading_info();
```

**4. Trigger to check validity of inserted values in the admin table**

```
create or replace function admin_info()
returns trigger as $body$
begin
if((new.dept='Research' or new.dept='Registrar'
or new.dept='Placement' or new.dept='Medical'
or new.dept='Maintenance' or new.dept='Library'
or new.dept='IT' or new.dept='Finance'
or new.dept='Counselling' or new.dept='Academic')
and (new.a_email like '%Educat.ac.in%')) then
raise notice 'Successfully entered Admin details';
return new;
else
raise exception 'invalid email address entered or department does not exist in institution';
end if;
end
$body$ LANGUAGE plpgsql;

create trigger Trigger_admin_info
after Insert or update
on Admin14
for each row
execute procedure admin_info();
```

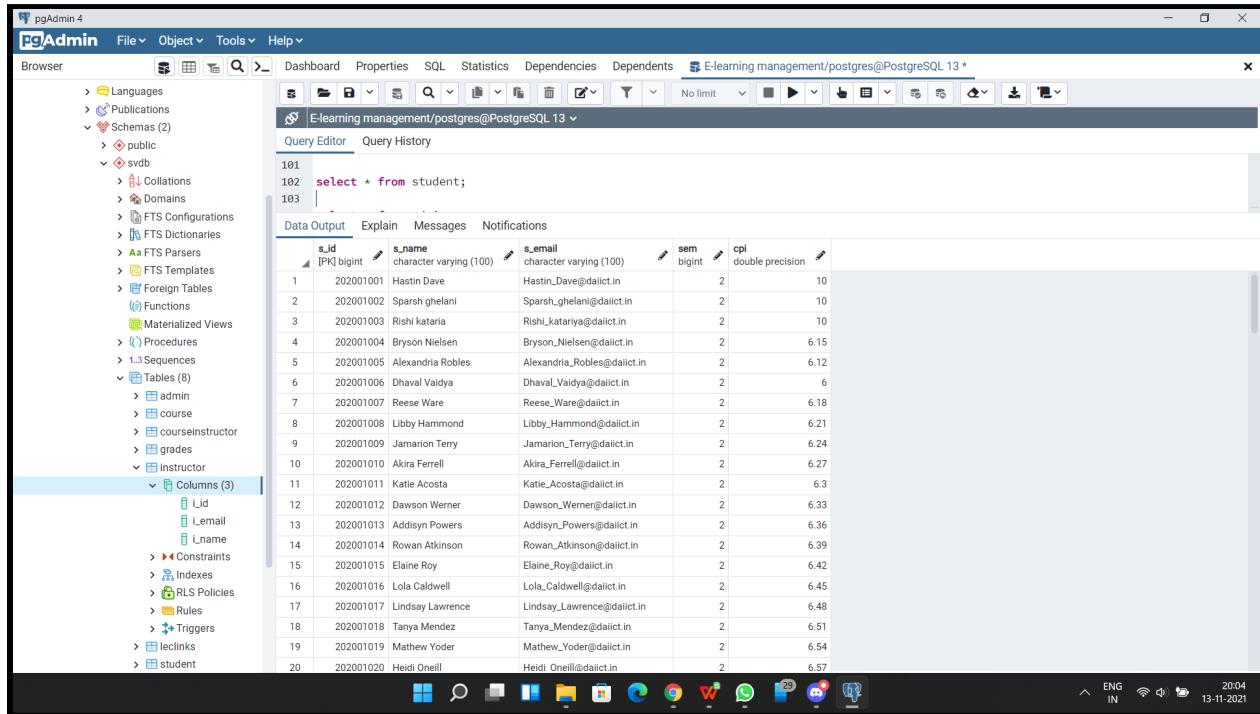
**5. Trigger to check validity of inserted values in the grading details table**

```
create or replace function instructor_info()
returns trigger as $body$
begin
if(new.i_email like '%Educat.ac.in%' and (new.i_id)>0) then
raise notice 'successfully entered instructor details with valid instructor id';
return new;
else
raise exception 'invalid email address entered or it does not belong to institution';
end if;
end
$body$ LANGUAGE plpgsql;

create trigger Trigger_instructor_info
after Insert or update
on Instructor14
for each row
execute procedure instructor_info();
```

## TABLES IN SQL

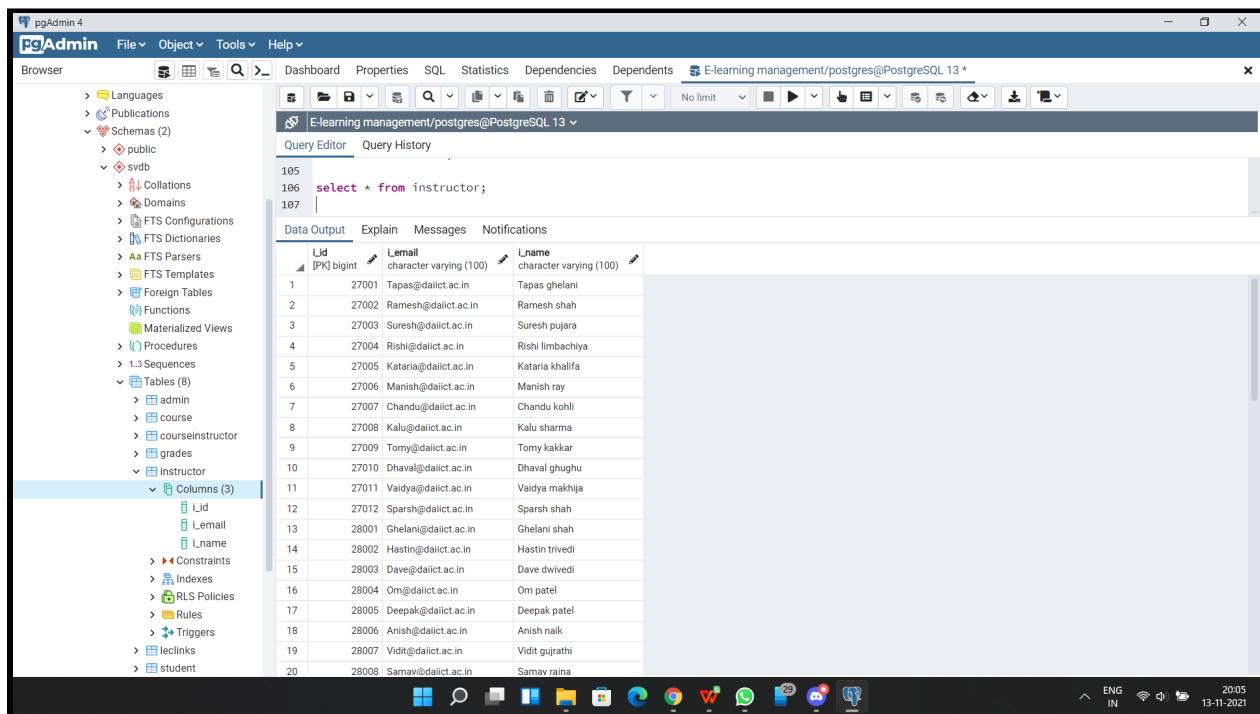
### Student Details



The screenshot shows the pgAdmin 4 interface with the 'student' table selected in the 'Tables' section of the left sidebar. The main pane displays the results of the query `select * from student;`. The table has columns: s\_id, s\_name, s\_email, sem, and cpi. The data shows 20 rows of student information.

s_id	s_name	s_email	sem	cpi
1	Hastin Dave	Hastin_Dave@daiict.in	2	10
2	Sparsh ghelani	Sparsh_ghelani@daiict.in	2	10
3	Rishi kataria	Rishi_Kataria@daiict.in	2	10
4	Bryson Nielsen	Bryson_Nielsen@daiict.in	2	6.15
5	Alexandria Robles	Alexandria_Robles@daiict.in	2	6.12
6	Dhaval Vaidya	Dhaval_Vaidya@daiict.in	2	6
7	Reese Ware	Reese_Ware@daiict.in	2	6.18
8	Libby Hammond	Libby_Hammond@daiict.in	2	6.21
9	Jamarion Terry	Jamarion_Terry@daiict.in	2	6.24
10	Akira Ferrell	Akira_Ferrell@daiict.in	2	6.27
11	Katie Acosta	Katie_Acosta@daiict.in	2	6.3
12	Dawson Werner	Dawson_Werner@daiict.in	2	6.33
13	Addisyn Powers	Addisyn_Powers@daiict.in	2	6.36
14	Rowan Atkinson	Rowan_Atkinson@daiict.in	2	6.39
15	Elaine Roy	Elaine_Roy@daiict.in	2	6.42
16	Lola Caldwell	Lola_Caldwell@daiict.in	2	6.45
17	Lindsay Lawrence	Lindsay_Lawrence@daiict.in	2	6.48
18	Tanya Mendez	Tanya_Mendez@daiict.in	2	6.51
19	Mathew Yoder	Mathew_Yoder@daiict.in	2	6.54
20	Heidi O'Neill	Heidi_O'Neill@daiict.in	2	6.57

### Instructor Details



The screenshot shows the pgAdmin 4 interface with the 'instructor' table selected in the 'Tables' section of the left sidebar. The main pane displays the results of the query `select * from instructor;`. The table has columns: l\_id, l\_email, and lname. The data shows 20 rows of instructor information.

l_id	l_email	lname
1	Tapas@daiict.ac.in	Tapas ghelani
2	Ramesh@daiict.ac.in	Ramesh shah
3	Suresh@daiict.ac.in	Suresh pulera
4	Rishi@daiict.ac.in	Rishi limbachiya
5	Kataria@daiict.ac.in	Kataria khilifa
6	Manish@daiict.ac.in	Manish ray
7	Chandu@daiict.ac.in	Chandu kohli
8	Kalu@daiict.ac.in	Kalu sharma
9	Tomy@daiict.ac.in	Tomy kakkar
10	Dhaval@daiict.ac.in	Dhaval ghoghu
11	Vaidya@daiict.ac.in	Vaidya makhlisa
12	Sparsh@daiict.ac.in	Sparsh shah
13	Ghelani@daiict.ac.in	Ghelani shah
14	Hastin@daiict.ac.in	Hastin trivedi
15	Dave@daiict.ac.in	Dave dwivedi
16	Om@daiict.ac.in	Om patel
17	Deepak@daiict.ac.in	Deepak patel
18	Anish@daiict.ac.in	Anish naik
19	Vidit@daiict.ac.in	Vidit gurathi
20	Samav@daiict.ac.in	Samav raina

## Admin Details

The screenshot shows the pgAdmin 4 interface with the 'admin' table selected in the 'Tables' section of the left sidebar. The table has three columns: a\_name, a\_email, and dept. The data consists of 20 rows of student information, all belonging to the 'Research' department.

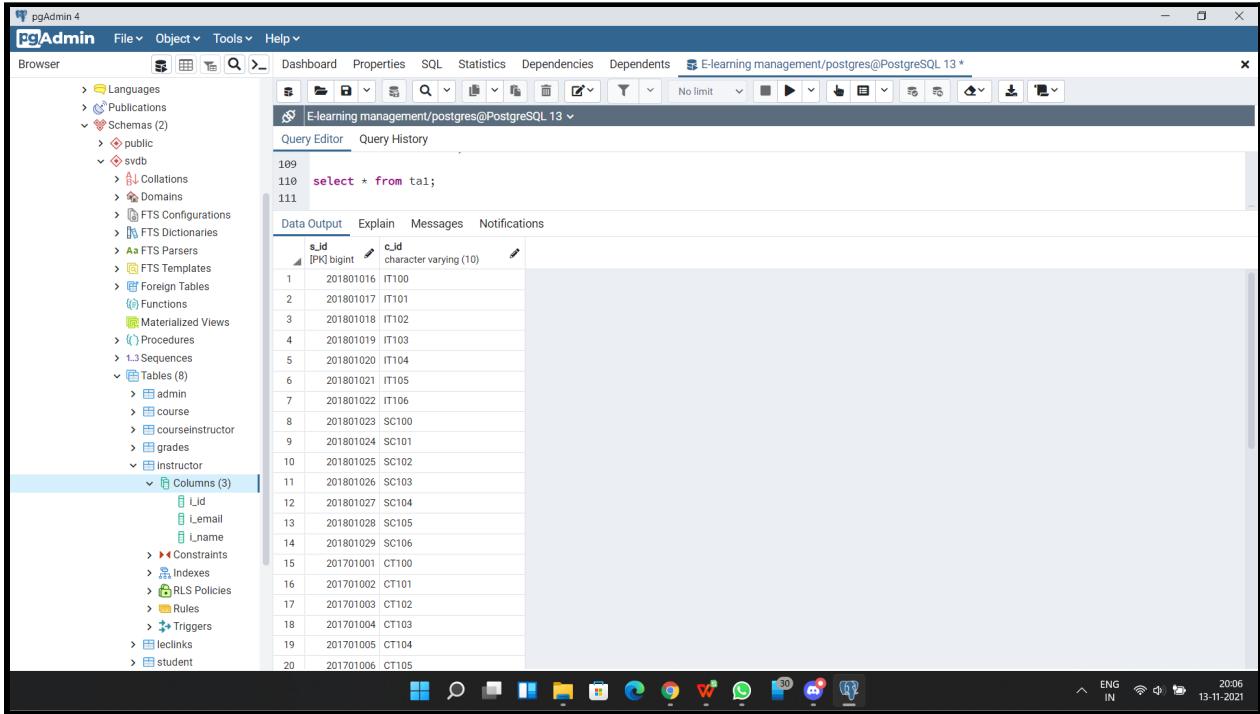
a_name	a_email	dept
1. Hastin Sharma	270042_Research@Educat.ac.in	Research
2. Jeremiah Jordan	270043_Research@Educat.ac.in	Research
3. Kenya Onell	270044_Research@Educat.ac.in	Research
4. Michaela Terry	270045_Research@Educat.ac.in	Research
5. Kyle Roberson	270046_Research@Educat.ac.in	Research
6. Marin Strong	270047_Research@Educat.ac.in	Research
7. Kinsley Todd	270048_Research@Educat.ac.in	Research
8. Quintin Jacobson	270049_Research@Educat.ac.in	Research
9. Anahi Benson	270050_Research@Educat.ac.in	Research
10. Shweta Tripathi	270051_Research@Educat.ac.in	Research
11. Minal Qureshi	270052_Research@Educat.ac.in	Research
12. Sparsh Tendulkar	260012_Registrar@Educat.ac.in	Registrar
13. Akshita Jindal	260013_Registrar@Educat.ac.in	Registrar
14. Vishwesh Jindal	260014_Registrar@Educat.ac.in	Registrar
15. Virat Kohli	260015_Registrar@Educat.ac.in	Registrar
16. Rishi Vihari	350031_placement@Educat.ac.in	Placement
17. Deja Larson	350032_placement@Educat.ac.in	Placement
18. Aliyah Baldwin	350033_placement@Educat.ac.in	Placement
19. Amari Jenkins	350034_placement@Educat.ac.in	Placement
20. Richa Kothari	350035_placement@Educat.ac.in	Placement

## Course Details

The screenshot shows the pgAdmin 4 interface with the 'course' table selected in the 'Tables' section of the left sidebar. The table has three columns: c\_id, c\_name, and credits. The data consists of 20 rows of course offerings, with credits ranging from 3.5 to 4.5.

c_id	c_name	credits
1 IT100	Programming1	3.5
2 IT101	Programming2	4
3 IT102	Programming3	4.5
4 IT103	Programming4	3.5
5 IT104	Programming5	4
6 IT105	Programming6	4.5
7 IT106	Programming7	3.5
8 IT107	Programming8	4
9 IT108	Programming9	4
10 IT109	Programming10	4
11 SC100	Mathematics1	4
12 SC101	Mathematics2	4
13 SC102	Mathematics3	4
14 SC103	Mathematics4	4.5
15 SC104	Mathematics5	3.5
16 SC105	Mathematics6	4.5
17 SC106	Mathematics7	4
18 SC107	Mathematics8	4
19 SC108	Mathematics9	3.5
20 SC109	Mathematics10	3.5

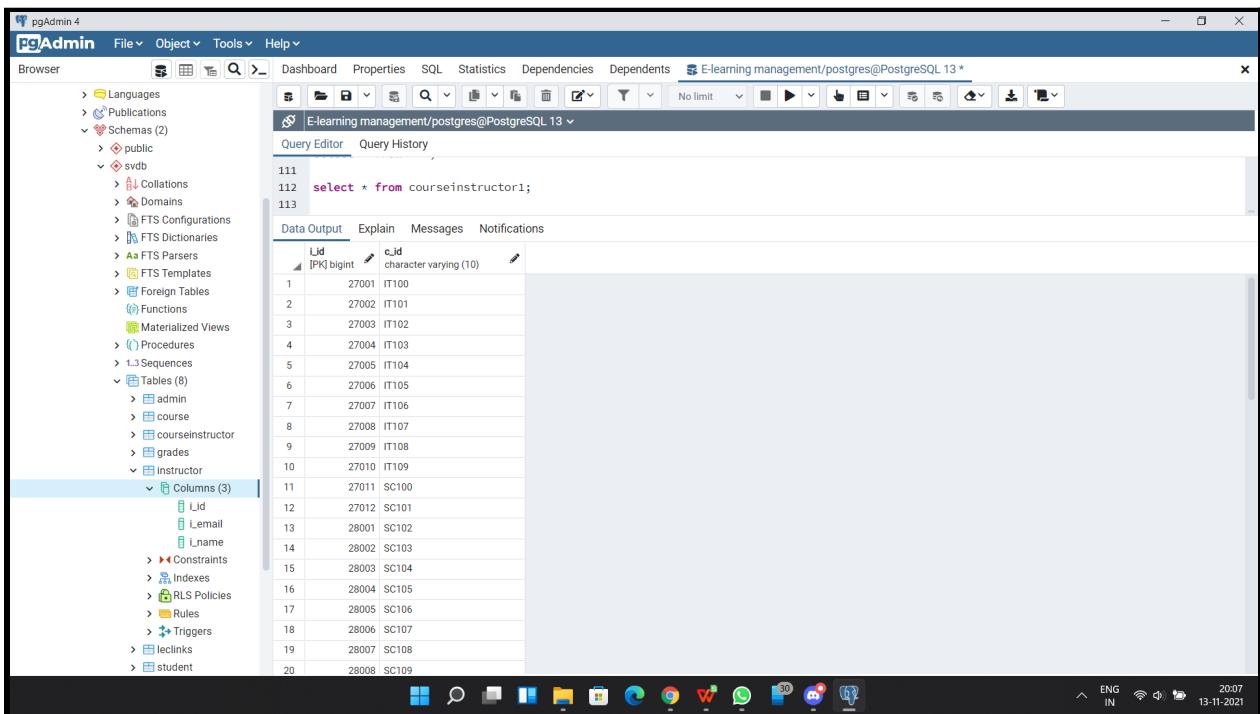
## Teaching Assistant Details



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'Languages', 'Publications', 'Schemas (2)', 'svb', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', 'Tables (8)', 'admin', 'course', 'courseinstructor', 'grades', and 'instructor'. The 'instructor' table is currently selected, showing its columns: 's\_id' (PK), 'c\_id', 'I\_id', 'I\_email', and 'I\_name'. The main pane shows the results of the query 'select \* from ta1;'. The results table has 20 rows, each containing values for s\_id, c\_id, I\_id, I\_email, and I\_name.

s_id	c_id	I_id	I_email	I_name
1	IT100	201801016		
2	IT101	201801017		
3	IT102	201801018		
4	IT103	201801019		
5	IT104	201801020		
6	IT105	201801021		
7	IT106	201801022		
8	SC100	201801023		
9	SC101	201801024		
10	SC102	201801025		
11	SC103	201801026		
12	SC104	201801027		
13	SC105	201801028		
14	SC106	201801029		
15	CT100	201701001		
16	CT101	201701002		
17	CT102	201701003		
18	CT103	201701004		
19	CT104	201701005		
20	CT105	201701006		

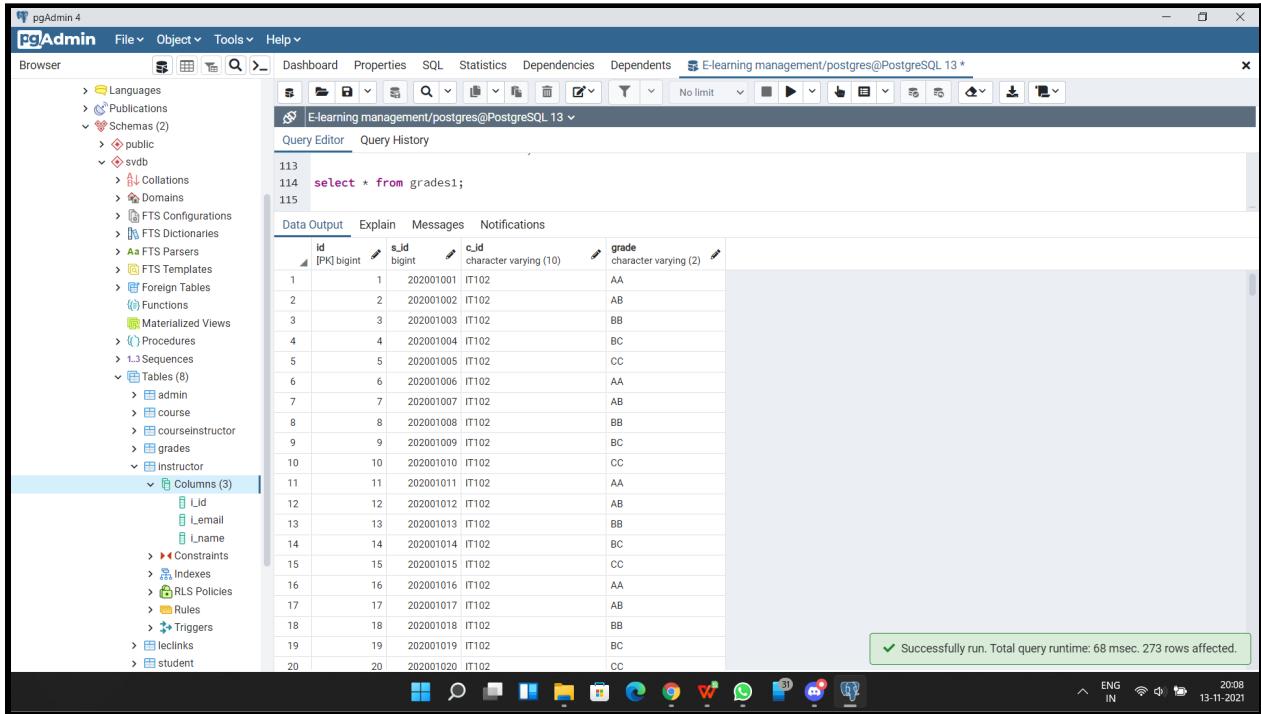
## Course-Instructor Relationship



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'Languages', 'Publications', 'Schemas (2)', 'svb', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', 'Tables (8)', 'admin', 'course', 'courseinstructor', 'grades', and 'instructor'. The 'instructor' table is currently selected, showing its columns: 'I\_id' (PK), 'c\_id', 'I\_id', 'I\_email', and 'I\_name'. The main pane shows the results of the query 'select \* from courseinstructor1;'. The results table has 20 rows, each containing values for I\_id, c\_id, I\_id, I\_email, and I\_name.

I_id	c_id	I_id	I_email	I_name
1	IT100	27001		
2	IT101	27002		
3	IT102	27003		
4	IT103	27004		
5	IT104	27005		
6	IT105	27006		
7	IT106	27007		
8	IT107	27008		
9	IT108	27009		
10	IT109	27010		
11	SC100	27011		
12	SC101	27012		
13	SC102	28001		
14	SC103	28002		
15	SC104	28003		
16	SC105	28004		
17	SC106	28005		
18	SC107	28006		
19	SC108	28007		
20	SC109	28008		

## Grading Details

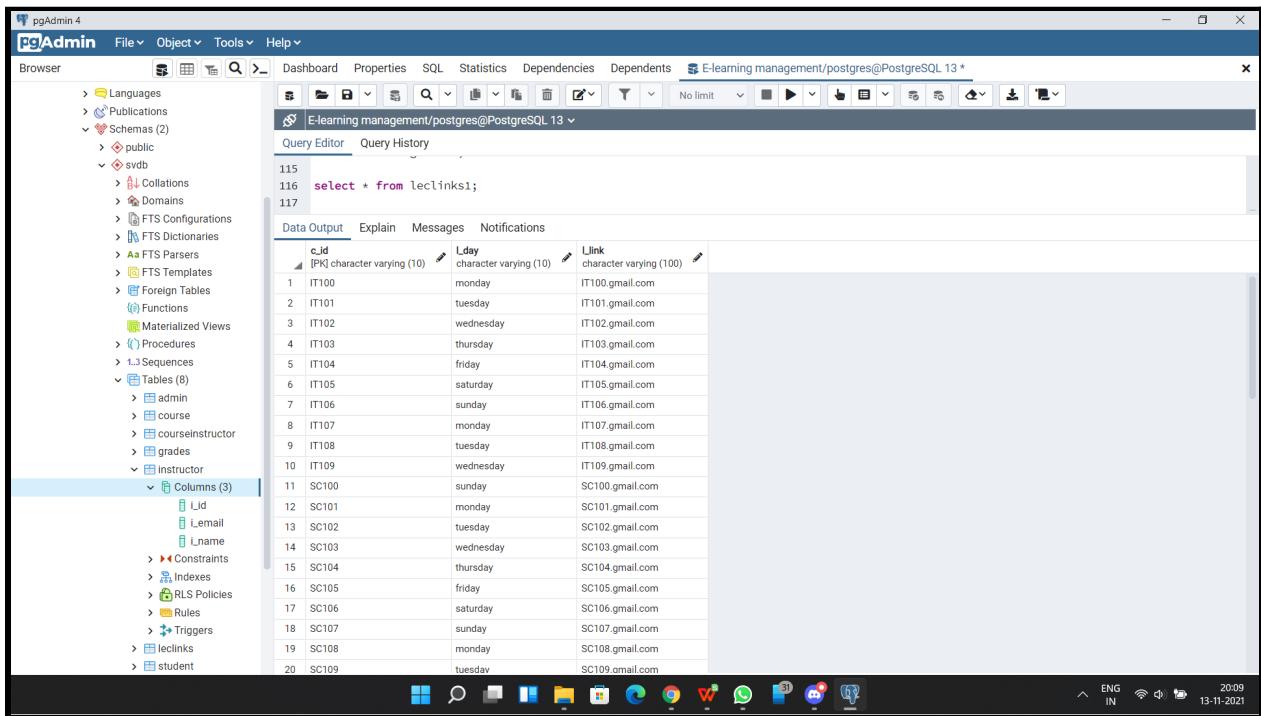


The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure with Schemas (2) containing public and svdb.
- Query Editor:** Displays the SQL query: `select * from grades1;`
- Data Output:** A table showing 27 rows of data from the 'grades1' table. The columns are id, s\_id, c\_id, and grade.
- Message Bar:** Shows a green message: "Successfully run. Total query runtime: 68 msec. 273 rows affected."
- System Bar:** Shows system icons for language (ENG IN), network, battery, and date/time (13-11-2021 20:08).

	<code>id</code>	<code>s_id</code>	<code>c_id</code>	<code>grade</code>
1	1	202001001	IT102	AA
2	2	202001002	IT102	AB
3	3	202001003	IT102	BB
4	4	202001004	IT102	BC
5	5	202001005	IT102	CC
6	6	202001006	IT102	AA
7	7	202001007	IT102	AB
8	8	202001008	IT102	BB
9	9	202001009	IT102	BC
10	10	202001010	IT102	CC
11	11	202001011	IT102	AA
12	12	202001012	IT102	AB
13	13	202001013	IT102	BB
14	14	202001014	IT102	BC
15	15	202001015	IT102	CC
16	16	202001016	IT102	AA
17	17	202001017	IT102	AB
18	18	202001018	IT102	BB
19	19	202001019	IT102	BC
20	20	202001020	IT102	CC

## Lecture Delivery Details



The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure with Schemas (2) containing public and svdb.
- Query Editor:** Displays the SQL query: `select * from leclinks1;`
- Data Output:** A table showing 20 rows of data from the 'leclinks1' table. The columns are c\_id, L\_day, and L\_link.
- System Bar:** Shows system icons for language (ENG IN), network, battery, and date/time (13-11-2021 20:09).

	<code>c_id</code>	<code>L_day</code>	<code>L_link</code>
1	IT100	monday	IT100.gmail.com
2	IT101	tuesday	IT101.gmail.com
3	IT102	wednesday	IT102.gmail.com
4	IT103	thursday	IT103.gmail.com
5	IT104	friday	IT104.gmail.com
6	IT105	saturday	IT105.gmail.com
7	IT106	sunday	IT106.gmail.com
8	IT107	monday	IT107.gmail.com
9	IT108	tuesday	IT108.gmail.com
10	IT109	wednesday	IT109.gmail.com
11	SC100	sunday	SC100.gmail.com
12	SC101	monday	SC101.gmail.com
13	SC102	tuesday	SC102.gmail.com
14	SC103	wednesday	SC103.gmail.com
15	SC104	thursday	SC104.gmail.com
16	SC105	friday	SC105.gmail.com
17	SC106	saturday	SC106.gmail.com
18	SC107	sunday	SC107.gmail.com
19	SC108	monday	SC108.gmail.com
20	SC109	tuesday	SC109.gmail.com

## QUERIES

### 1. View names of the students with CPI > 9.

SQL - select s\_name,cpi from student where cpi>9

The screenshot shows the pgAdmin 4 interface with the 'Data Output' tab selected. The results of a query are displayed in a table with two columns: 's\_name' and 'cpi'. The data consists of 19 rows of student information.

s_name	cpi
Hastin Dave	10
Sparsh ghelani	10
Rishi kataria	10
Kenyon Wilkerson	9.03
Abraham Sandoval	9.06
Abigayle Savage	9.09
Reagan Oneal	9.12
Robert Greene	9.15
Aliya Richardson	9.18
June Garcia	9.21
Malachi Powell	9.24
Zoe Melendez	9.27
Zoey Schaefer	9.3
Kelsey Paul	9.33
Jerry Cross	9.36
Mylie Krause	9.39
Kylan Finley	9.42
Freddy Park	9.45
Ronin Bradford	9.48

### 2. View Student IDs having a grade of AA in IT102.

SQL - select s\_id from grades1 where grade='AA' and c\_id='IT102'

The screenshot shows the pgAdmin 4 interface with the 'Data Output' tab selected. The results of a query are displayed in a table with one column: 's\_id'. The data consists of 7 rows of student IDs.

s_id
202001001
202001006
202001011
202001016
202001021
202001026
202001031

### 3. View courses worth three credits.

SQL - select c\_id from course1 where credits=3

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (including public and svdb), Tables (8), and so on. The main window has two tabs: 'Query Editor' and 'Query History'. The 'Query Editor' tab contains the following SQL code:

```
1 set search_path to svdb
2
3 select c_id from course1 where credits=3
```

The 'Data Output' tab shows the results of the query:

c_id
CT107
EL105
IE100

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 342 msec. 3 rows affected."

### 4. View all students studying IT102.

SQL - select s\_id from grades1 where c\_id='IT102';

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with Servers (1), PostgreSQL 13, Databases (3), Dhaval, and so on. The main window has two tabs: 'Query Editor' and 'Query History'. The 'Query Editor' tab contains the following SQL code:

```
1 set search_path to svdb;
2
3 select s_id from grades1 where c_id='IT102';
4
```

The 'Data Output' tab shows the results of the query:

s_id
202001001
202001002
202001003
202001004
202001005
202001006
202001007
202001008
202001009
202001010
202001011
202001012
202001013
202001014
202001015
202001016
202001017
202001018
202001019
202001020
202001021

## 5. View all lectures on Monday.

SQL - select c\_id from leclinks1 where l\_day='monday'

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Casts, Catalogs, Event Triggers, etc. The main area shows a query editor with the following SQL code:

```
1 set search_path to svdb
2
3 select c_id from leclinks1 where l_day='monday'
```

The Data Output tab shows the results of the query:

c_id
IT100
IT107
SC101
SC108
CT102
CT109
EL106
IE103
HM100

A message at the bottom right indicates: "Successfully run. Total query runtime: 100 msec. 10 rows affected."

## 6. View student ID and names whose grade is AB. Also, mention the corresponding course id in which the student got this grade.

SQL - select s\_id,s\_name,c\_id from student natural join grades1 where student.s\_id=grades1.s\_id and grade='AB'

The screenshot shows the pgAdmin 4 interface with a different database connection. The left sidebar shows servers and databases, including 'PostgreSQL 13' and '201901206\_Lab\_9'. The main area shows a query editor with the following SQL code:

```
1 select s_id,s_name,c_id from student natural join grades1 where student.s_id=grades1.s_id and grade='AB'
```

The Data Output tab shows the results of the query:

s_id	s_name	c_id
202001002	Sparsh ghelani	IT102
202001007	Reese Ware	IT102
202001012	Dawson Werner	IT102
202001017	Lindsay Lawrence	IT102
202001022	Lawson Harris	IT102
202001027	Tania Oliver	IT102
202001032	Omar Schaefer	IT102
202001005	Alexandria Robles	SC102
202001010	Akira Ferrell	SC102
202001015	Elaine Roy	SC102
202001020	Heidi O'Neill	SC102
202001025	Darrell Cohen	SC102
202001030	Lilia Blackburn	SC102
202001003	Rishi kataria	IE102
202001008	Libby Hammond	IE102
202001013	Addisyn Powers	IE102
202001018	Tanya Méndez	IE102
202001023	Sawyer Vaughan	IE102
202001028	Marin Kemp	IE102

A message at the bottom right indicates: "Successfully run. Total query runtime: 1 sec."

## 7. View students who are either in semester two or semester six.

SQL - select s\_id,s\_name,sem from student where sem=2 or sem=6;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The 'Query Editor' tab is active, containing the following SQL code:

```
5 select s_id,s_name,sem from student where sem=2 or sem=6;
```

The 'Data Output' tab shows the results of the query:

s_id	s_name	sem
1	Hastin Dave	2
2	Sparsh ghelani	2
3	Rishi kataria	2
4	Bryson Nielsen	2
5	Alexandria Robles	2
6	Dhaval Vaidya	2
7	Reese Ware	2
8	Libby Hammond	2
9	Jamariion Terry	2
10	Akira Ferrell	2
11	Katie Acosta	2
12	Dawson Werner	2
13	Addieyn Powers	2
14	Rowan Atkinson	2
15	Elaine Roy	2
16	Lola Caldwell	2
17	Lindsay Lawrence	2
18	Tanya Mendez	2
19	Mathew Yoder	2
20	Heidi O'Neill	2

## 8. Find students studying in semester six and having a CPI of 8+.

SQL - select s\_id,s\_name,sem,cpi from student where sem=6 and cpi>=8;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Servers (1)'. The 'Query Editor' tab is active, containing the following SQL code:

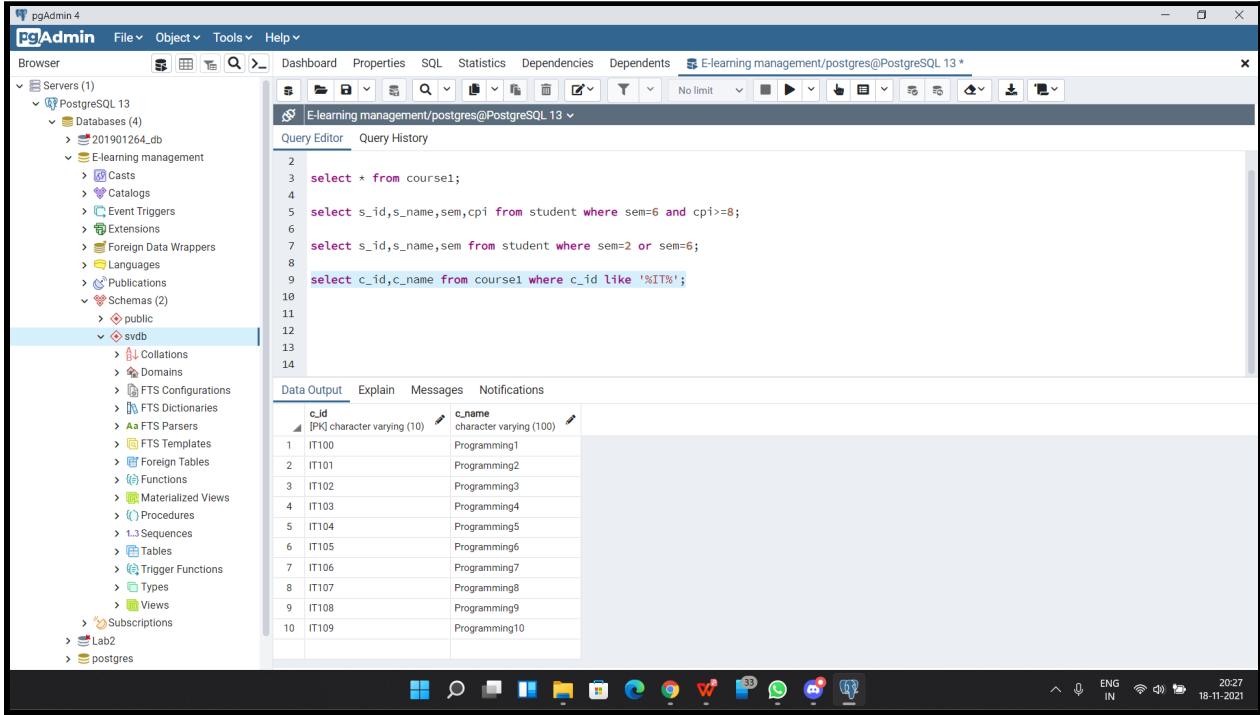
```
4 select s_id,s_name,sem,cpi from student where sem=6 and cpi>=8;
```

The 'Data Output' tab shows the results of the query:

s_id	s_name	sem	cpi
1	Sonia Jimenez	6	8.01
2	Kellen Cuevas	6	8.04
3	Jaylin Levine	6	8.07
4	Ximena Chung	6	8.1
5	Jada Adams	6	8.13
6	Camila Holt	6	8.16
7	Alijah Cobb	6	8.19
8	Laurn Kennedy	6	8.22
9	Jakobe Townsend	6	8.25
10	Lukas Shields	6	8.28
11	Laura Meyer	6	8.31
12	Rebekah Manning	6	8.34
13	Gretchen Santiago	6	8.37
14	Ayden Cunningham	6	8.4
15	Cynthia Short	6	8.43
16	Amir Chandler	6	8.46
17	Makala Terrell	6	8.49
18	Jimmy Juarez	6	8.52
19	Camille McGrath	6	8.55
20	Riley Owens	6	8.58

## 9. Print all the Programming courses.

SQL - select c\_id,c\_name from course1 where c\_id like '%IT%';



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'PostgreSQL 13' and 'E-learning management'. The main window shows a query editor with the following SQL code:

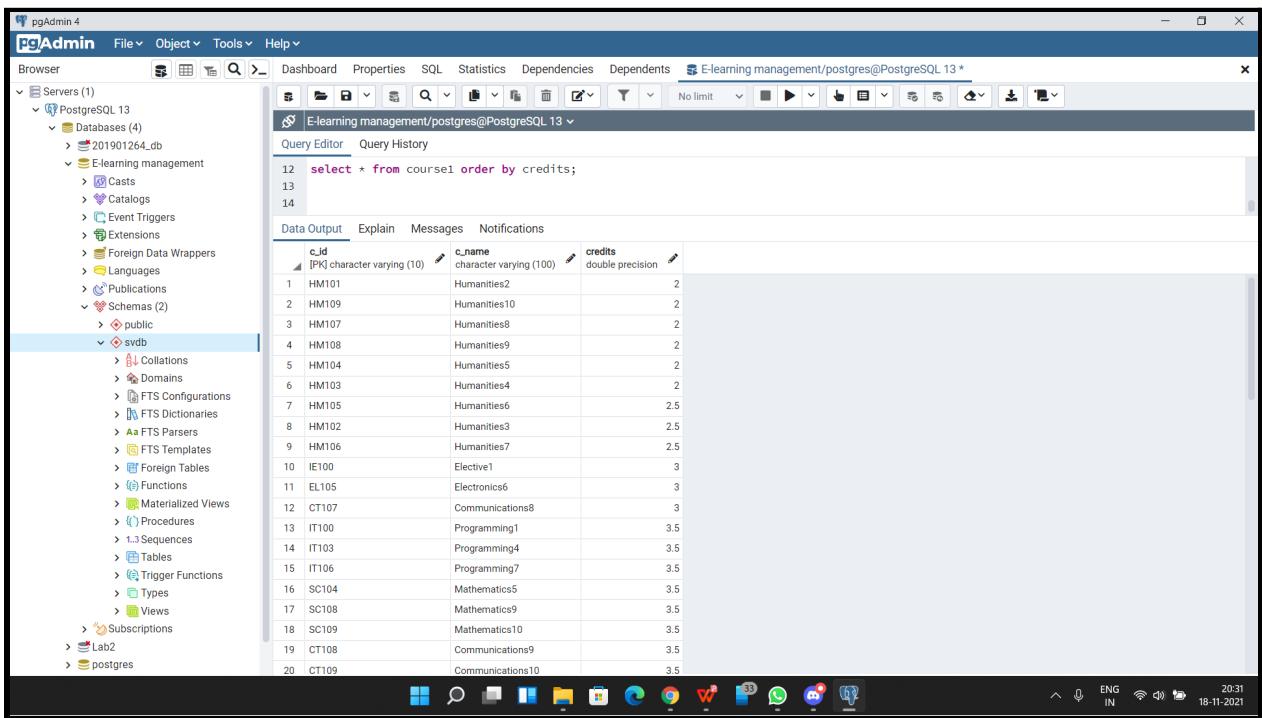
```
2
3 select * from course1;
4
5 select s_id,s_name,sem,cpi from student where sem=6 and cpi>=8;
6
7 select s_id,s_name,sem from student where sem=2 or sem=6;
8
9 select c_id,c_name from course1 where c_id like '%IT%';
10
11
12
13
14
```

The results are displayed in a table titled 'Data Output' with columns 'c\_id', 'c\_name', and 'credits'. The data is as follows:

c_id	c_name	credits
IT100	Programming1	
IT101	Programming2	
IT102	Programming3	
IT103	Programming4	
IT104	Programming5	
IT105	Programming6	
IT106	Programming7	
IT107	Programming8	
IT108	Programming9	
IT109	Programming10	

## 10. Sort all the courses in ascending order of their credits.

SQL - select \* from course1 order by credits;



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'PostgreSQL 13' and 'E-learning management'. The main window shows a query editor with the following SQL code:

```
12 select * from course1 order by credits;
13
14
```

The results are displayed in a table titled 'Data Output' with columns 'c\_id', 'c\_name', and 'credits'. The data is as follows:

c_id	c_name	credits
HM101	Humanities2	2
HM109	Humanities10	2
HM107	Humanities8	2
HM108	Humanities9	2
HM104	Humanities5	2
HM103	Humanities4	2
HM105	Humanities6	2.5
HM102	Humanities3	2.5
HM106	Humanities7	2.5
IE100	Elective1	3
EL105	Electronics6	3
CT107	Communications8	3
IT100	Programming1	3.5
IT103	Programming4	3.5
IT106	Programming7	3.5
SC104	Mathematics5	3.5
SC108	Mathematics9	3.5
SC109	Mathematics10	3.5
CT108	Communications9	3.5
CT109	Communications10	3.5

## 11. Find all the courses taught in semester 2.

SQL - select c\_id,c\_name,credits from course1 where c\_id like '%102%';

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under 'PostgreSQL 13'. In the center, the 'Query Editor' pane contains the following SQL code:

```
12 select * from course1 order by credits;
13
14 select c_id,c_name,credits from course1 where c_id like '%102%';
```

The 'Data Output' tab is selected, showing the results of the second query:

c_id	c_name	credits
IT102	Programming3	4.5
SC102	Mathematics3	4
CT102	Communications3	4.5
EL102	Electronics3	3.5
IE102	Elective3	3.5
HM102	Humanities3	2.5

## 12. Mention the names of people who work in the Placement department

SQL - select a\_name from admin where dept='Placement';

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under 'PostgreSQL 13'. In the center, the 'Query Editor' pane contains the following SQL code:

```
16 select * from admin;
17
18 select a_name from admin where dept='Placement';
```

The 'Data Output' tab is selected, showing the results of the second query:

a_name
Rishi Vihari
Deja Larson
Aliyah Baldwin
Amari Jenkins
Richa Kothari
Harry Panchal
Aman Dhatterwal
Mukesh Hiranandani

### 13. Mention the names of people who work in the library department

SQL - select a\_name from admin where dept='Library';

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the 'svdb' schema, including Publications, Schemas, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (8), Trigger Functions, Types, Views, Subscriptions, and PostgreSQL roles. The 'Tables' node is selected. The main window shows the results of a query in the 'Query Editor' tab:

```
select a_name from admin where dept='Placement'
```

The results table contains three rows:

a_name
Nicobar Tesla
Bryce McCullough
Taimur Khan

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 116 msec. 3 rows affected."

### 14. Find s\_ids of TAs who assist in an Electronics course.

SQL - select s\_id from ta1 where c\_id like '%EL%'

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the 'E-learning management' schema, including Databases, Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Tables (13). The 'Tables' node is selected. The main window shows the results of a query in the 'Query Editor' tab:

```
select * from ta1;
```

```
select s_id from ta1 where c_id like '%EL%'
```

The results table contains seven rows:

s_id
201701008
201701009
201701010
201701011
201701012
201701013
201701014

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 68 msec. 7 rows affected."

## 15. Find all the TAs who study in semester 6.

SQL - select s\_id from ta1 where s\_id>201799999 and s\_id<201900000

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'E-learning management'. The main area shows a query editor with the following SQL code:

```
16 select * from ta1;
17
18 select s_id from ta1 where s_id>201799999 and s_id<201900000;
```

The results pane displays a table with 14 rows, each containing an 's\_id' value ranging from 201801016 to 201801029. A green message at the bottom right indicates the query was successfully run with a total runtime of 75 msec and 14 rows affected.

## 16. Find instructor id of instructors that teaches Electronic courses

SQL - select \* from courseinstructor1 where c\_id like'%EL%';

The screenshot shows the pgAdmin 4 interface with a different database connection. The left sidebar shows tables like 'admin', 'course1', 'courseinstructor1', etc. The main area shows a query editor with the following SQL code:

```
21. A
22. A
```

The results pane displays a table with 10 rows, each containing an 'i\_id' value (29006 to 29015) and a corresponding 'c\_id' value ('EL100' to 'EL109').

**17. Find the name/s of the student/s having the lowest cpi.**

SQL- select s\_name from student where cpi=(select min(cpi) from student);

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays the database schema, including tables like course1, courseinstructor1, grades1, instructor, leclinks1, and student. The student table is expanded, showing its columns: s\_id, s\_name, s\_email, sem, and cpi. A query result table is open in the center, showing one row: Dhaval Vaidya. The top bar shows the connection details: 201901206\_Lab\_9/postgres@PostgreSQL 13\*.

s_name
Dhaval Vaidya

**18. Find the student ID of students who got BB grades in the IT course.**

SQL - select s\_id from grades1 where grade='BB' and c\_id like '%IT%';

The screenshot shows the pgAdmin 4 interface. The schema browser is visible on the left. A query result table is shown in the center, listing student IDs (s\_id) from the student table. The results are as follows:

s_id
202001003
202001008
202001013
202001018
202001023
202001028
201901002
201901007
201901012
201901017
201901022
201901027
201801002
201801007
201801012

**19. Find the total number of students having CPI between 7 and 8.**

SQL - select COUNT(\*) from student where cpi<=8 and cpi>=7;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'student' schema, including tables like course1, courseinstructor1, grades1, instructor, leclinks1, and student. The 'student' table is selected, showing its columns: s\_id, s\_name, s\_email, sem, and cpi. The main pane contains a query editor with the following SQL code:

```
count
bigin
1 33
```

A message box at the bottom right indicates: "Successfully run. Total query runtime: 178 msec. 1 rows affected."

**20. Count the total number of students having grades either 'AA' or 'AB' in any subject.**

SQL - select count(\*) from grades1 where grade='AA' or grade='AB'

The screenshot shows the pgAdmin 4 interface with a more complex query in the Query Editor. The code includes multiple SELECT statements and subqueries. The results pane shows the output of the final COUNT(\*) query:

```
count
bigin
1 110
```

**21. Find the names of instructors who have surnames as 'shah.'**

SQL - select i\_name from instructor where i\_name like '%shah%';

	Lname	s_id	s_name	s_email	sem	cpi
1	Ramesh shah					
2	Sparsh shah					
3	Ghelani shah					
4	Sagar shah					

**22. Find names of instructors that teach in SC ( mathematics) courses.**

SQL - select \* from instructor natural join courseinstructor1 where instructor.i\_id=courseinstructor1.i\_id and c\_id like '%SC%';

	L_id	L_email	Lname	c_id
1	27011	Vaidya@EduCat.ac.in	Vaidya makhija	SC100
2	27012	Sparsh@EduCat.ac.in	Sparsh shah	SC101
3	28001	Ghelani@EduCat.ac.in	Ghelani shah	SC102
4	28002	Hastin@EduCat.ac.in	Hastin trivedi	SC103
5	28003	Dave@EduCat.ac.in	Dave dwivedi	SC104
6	28004	Om@EduCat.ac.in	Om patel	SC105
7	28005	Deepak@EduCat.ac.in	Deepak patel	SC106
8	28006	Anish@EduCat.ac.in	Anish naik	SC107
9	28007	Vidit@EduCat.ac.in	Vidit gujrathi	SC108
10	28008	Samay@EduCat.ac.in	Samay raina	SC109

**23. Find names of students and their cpi who assist in the IT course.**

SQL- select s\_name,cpi from student natural join ta1 where student.s\_id=ta1.s\_id and c\_id like '%IT%';

	s_name	cpi
1	Laura Meyer	8.31
2	Rebekah Manning	8.34
3	Gretchen Santiago	8.37
4	Ayden Cunningham	8.4
5	Cynthia Short	8.43
6	Amir Chandler	8.46
7	Makaila Terrell	8.49

**24. Find the number of students and average cpi of sem2 students.**

```
SQL - create or replace function semwise_cpi(seme_ster integer)
returns Table(tot_stude bigint,avg_cpi float) as $body$
begin
return query select count(*),avg(cpi) from Student14 where sem = seme_ster;
end
$body$ LANGUAGE plpgsql;
```

```
select * from semwise_cpi(2);
```

tot_stude	avg_cpi
32	6.834375

**25. Check if the output of the trigger for inserting tuples in the student table is working or not. If inserted, show the entry in the student table.**

**Insert statement :**

```
insert into Student14(s_id,s_name,s_email,sem,cpi) values(202001033, 'ManishPathak',  
'manish\_pathak@gmail.com', 6, 8.45);
```



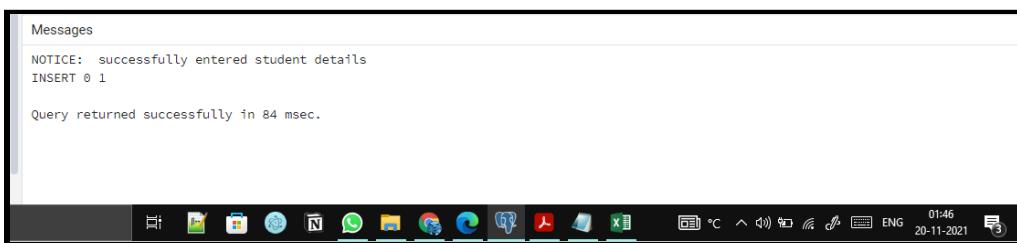
The screenshot shows a terminal window titled 'Messages' with the following error output:

```
139  
Messages  
  
ERROR: invalid email address  
CONTEXT: PL/pgSQL function student_info() line 10 at RAISE  
SQL state: P0001
```

The system tray at the bottom right shows the date as 20-11-2021 and the time as 01:43.

**Corrected insert statement :**

```
insert into Student14(s_id,s_name,s_email,sem,cpi) values(202001033,'Manish  
Pathak','manish\_pathak@Educat.ac.in',6,8.45);
```

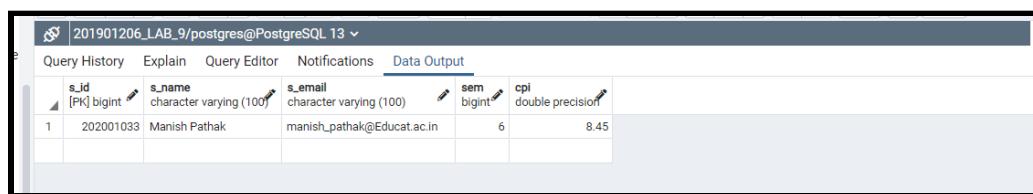


The screenshot shows a terminal window titled 'Messages' with the following success output:

```
NOTICE: successfully entered student details  
INSERT 0 1  
  
Query returned successfully in 84 msec.
```

The system tray at the bottom right shows the date as 20-11-2021 and the time as 01:46.

**SQL - select \* from Student14 where s\_id = 202001033**



The screenshot shows a PostgreSQL Data Output window with the following table:

s_id	s_name	s_email	sem	cpi
202001033	Manish Pathak	manish_pathak@Educat.ac.in	6	8.45

**26. Find the course ID in SC in which the students got maximum BC grade.**

SQL - create view SC\_freq as

```
select c_id,count(grade)  
as freq_BC from Grades14 where c_id like '%SC%'  
and grade = 'BC' group by c_id;  
Select * From SC_freq;
```

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Servers (1) > PostgreSQL 13 > Databases (2) > 201901206\_LAB\_9 > Views', there is a list of objects: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, and Publications. The 'Data Output' tab is selected in the top right, showing a table with two columns: 'c\_id' and 'freq\_bc'. The data rows are: 1 SC102 (freq\_bc: 7), 2 SC104 (freq\_bc: 6), and 3 SC106 (freq\_bc: 6).

c_id	freq_bc
1	7
2	6
3	6

```
select c_id from SC_freq where freq_BC=(select max(freq_BC) from SC_freq);
```

The screenshot shows the pgAdmin 4 interface with the 'Data Output' tab selected. It displays a single row with the value '1 SC102'.

c_id
1 SC102

**27. Print IT grades of all students.**

SQL - select student.s\_id,grades1.c\_id,grades1.grade

```
from student left join grades1 on student.s_id=grades1.s_id  
where c_id like '%IT%'
```

The screenshot shows the pgAdmin 4 interface with the 'Data Output' tab selected. It displays a table with three columns: 's\_id', 'c\_id', and 'grade'. The data rows are: 1 202001001 IT102 AA, 2 202001002 IT102 AB, 3 202001003 IT102 BB, 4 202001004 IT102 BC, 5 202001005 IT102 CC, 6 202001006 IT102 AA, 7 202001007 IT102 AB, and 8 202001008 IT102 BB.

s_id	c_id	grade
1	202001001	IT102
2	202001002	IT102
3	202001003	IT102
4	202001004	IT102
5	202001005	IT102
6	202001006	IT102
7	202001007	IT102
8	202001008	IT102

**28. Find the output for insertion of the following values in admin.**

**Insert statement.**

```
insert into Admin14(a_name,a_email,dept)
values('aniket roy','aniket_roy@Educat.ac.in','Anti ragging');
```

The screenshot shows the pgAdmin interface. On the left is a tree view of database objects under 'mydata'. The 'Messages' panel displays an error message: 'ERROR: invalid email address entered or department does not exist in institution'. The 'Query Editor' tab contains the original insert statement with the error. The 'Notifications' tab is also visible.

**Corrected insert statement :**

```
insert into Admin14(a_name,a_email,dept)
values('aniket pathak','aniket\_pathak@Educat.ac.in','academic');
```

The screenshot shows the pgAdmin interface. The 'Messages' panel displays a 'NOTICE: Successfully entered Admin details' message and 'INSERT 0 1'. The 'Query Editor' tab shows the corrected insert statement and its execution. The 'Notifications' tab is also visible.

**SQL -** select \* from Admin14 where a\_email='aniket\_pathak@Educat.ac.in';

The screenshot shows the pgAdmin interface. The 'Data Output' panel displays the results of the query: a single row with columns 'a\_name', 'a\_email', and 'dept', containing the values 'aniket pathak', '[aniket\\_pathak@Educat.ac.in](mailto:aniket_pathak@Educat.ac.in)', and 'academic' respectively.

**29. Find the course with the maximum number of 'AA' grades.**

**SQL -** create view FREQ1 as

```
select c_id,count(grade)
as freq from grades1 where grade = 'AA' group by c_id;
```

```
select * From FREQ1;
```

```
select c_id,freq from FREQ1 where freq = (select max(freq) from FREQ1);
```

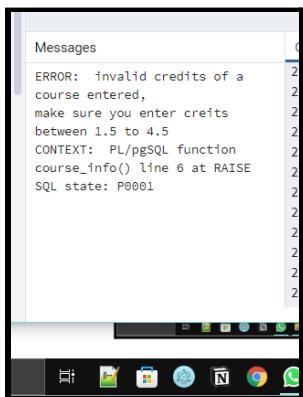
The screenshot shows the pgAdmin interface. The 'Data Output' panel displays the results of the final query: two rows with columns 'c\_id' and 'freq', containing the values 'IE102' and 'IT102' with a frequency of 7 each.

**30. Check if the output of the trigger for inserting tuple in the Course table is working or not. If inserted, show the entry in the Course table.**

**SQL -**

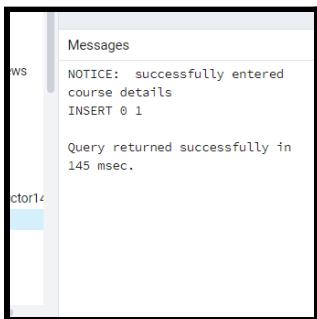
**Insert statement :**

```
insert into Course14(c_id,c_name,credits) values('SC555','Mathematical Supremacy',5)
```



**Corrected insert statement :**

```
insert into Course14(c_id,c_name,credits)
values('SC555','Mathematical Supremacy',4);
```



**SQL -** select \* from Course14 where c\_id='SC555';

A screenshot of the pgAdmin III interface. The left sidebar shows a tree view of database objects under the schema "mydata". The main area is titled "Data Output" and displays a table with one row of data:

	c_id	c_name	credits
1	SC555	Mathematical Supremacy	4

- 31. Check if the output of the trigger for inserting the tuple in the Grades table is working or not. If inserted, show the entry in the Grades table.**

**Insert statement :**

```
insert into Grades14(id,s_id,c_id,grade) values(274,202001033,'IT106','A');
```

Messages

NOTICE: make sure you write grade between DE to AA. Write inside single quotes

ERROR: invalid\_grade entered  
CONTEXT: PL/pgSQL function grading\_info() line 9 at RAISE  
SQL state: P0001

**Corrected insert statement :**

```
insert into Grades14(id,s_id,c_id,grade) values(274,202001033,'IT106','AA');
```

Messages

NOTICE: make sure you write grade between DE to AA. Write inside single quotes

NOTICE: successfully inserted grade details

INSERT 0 1

Query returned successfully in 109 msec.

**SQL - select \* from Grades14 where s\_id = 202001033**

	<b>[PK] bigint</b>	<b>s_id</b> <b>bigrnt</b>	<b>c_id</b> <b>character varying (10)</b>	<b>grade</b> <b>character varying (2)</b>
1	274	202001033	IT106	AA

- 32. Find the course ID in IT in which the student got maximum AA grade.**

**SQL - create view FREQ as**

```
select c_id,count(grade)
as freq_AA from Grades14 where c_id like '%IT%'  
and grade = 'AA' group by c_id;
```

Select \* From FREQ;

<b>c_id</b> <b>character varying (10)</b>	<b>freq_aa</b> <b>bigrnt</b>
1 IT104	6
2 IT106	5
3 IT102	7

```
select c_id from FREQ where freq_AA=(select max(freq_AA) from FREQ);
```

c_id	character varying (10)
1	IT102

### 33. Find out frequency of BB grade in IE104 and IE106 courses.

SQL - create view FREQ as select c\_id,count(grade)

as freq\_BB from Grades14 where (c\_id='IE104' and grade='BB') or (c\_id='IE106' and grade = 'BB') group by c\_id

c_id	freq_bb
1	IE104
2	IE106

### 34. Find the grade that appears most times in the grade table.

SQL - create view Grade\_Freq as

select grade,count(grade) as freq from Grades14 group by grade;

Select \* from Grade\_Freq;

select grade,freq from Grade\_Freq

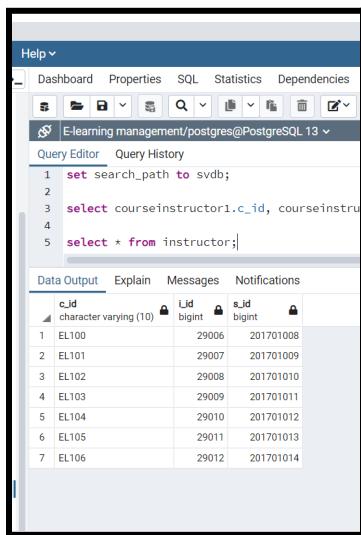
grade	freq
AB	55
BB	55
AA	56
CC	54
BC	54

where freq=(select max(freq) from Grade\_Freq);

grade	freq
AA	56

**35. Find the instructors who teach EL courses and their Teaching Assistants.**

**SQL** - select courseinstructor1.c\_id, courseinstructor1.i\_id, ta1.s\_id from courseinstructor1 natural join ta1 where courseinstructor1.c\_id=ta1.c\_id and c\_id like '%EL%';



The screenshot shows the pgAdmin 4 interface. In the Query Editor tab, the following SQL code is entered:

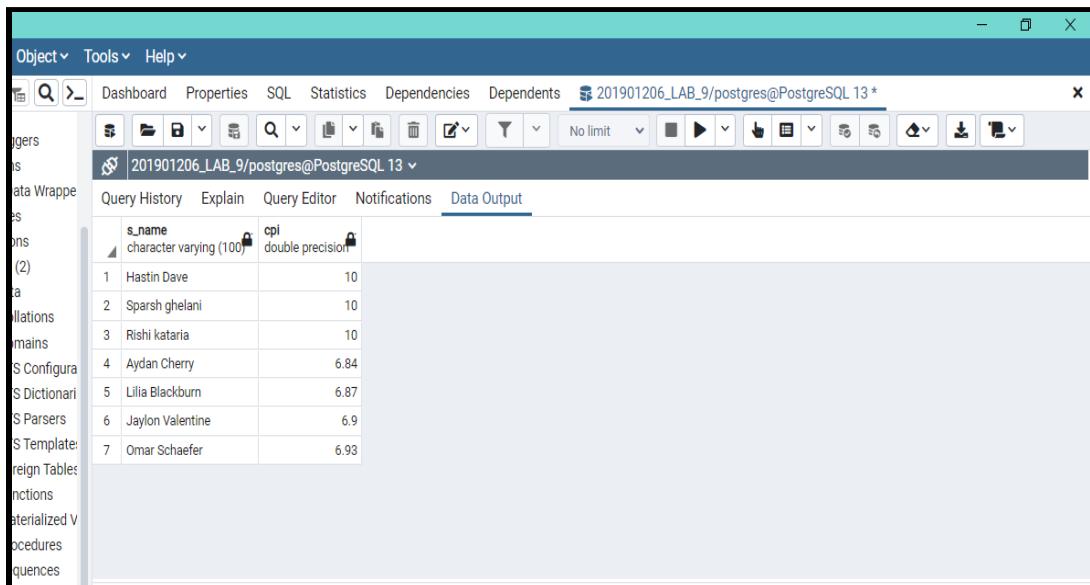
```
1 set search_path to svdb;
2
3 select courseinstructor1.c_id, courseinstructor1.i_id, ta1.s_id
4
5 select * from instructor;
```

The Data Output tab displays the results of the query:

c_id	i_id	s_id
EL100	29006	201701008
EL101	29007	201701009
EL102	29008	201701010
EL103	29009	201701011
EL104	29010	201701012
EL105	29011	201701013
EL106	29012	201701014

**36. Find the names of students studying in sem 2 whose CPI is more than or equal to the average cpi of sem 2.**

**SQL** - select s\_name,cpi from Student14 where sem=2 and cpi>=(select avg\_cpi from semwise\_cpi(2))



The screenshot shows the pgAdmin 4 interface. In the Query History tab, the following SQL code is selected:

```
1 select s_name, cpi
2 from Student14
3 where sem=2 and cpi>=(select avg_cpi from semwise_cpi(2));
```

The Data Output tab displays the results of the query:

s_name	cpi
Hastin Dave	10
Sparsh ghelani	10
Rishi kataria	10
Aydan Cherry	6.84
Lilia Blackburn	6.87
Jaylon Valentine	6.9
Omar Schaefer	6.93

**37. Find the instructors who teach the SC course and the ID of TA's who are assisting the course.**

SQL - select c\_id,i\_id,i\_name,s\_id from Instructor14  
natural join Courseinstructor14 natural join Ta14  
where c\_id like '%SC%' and Instructor14.i\_id = Courseinstructor14.i\_id;

	c_id	i_id	i_name	s_id
1	SC100	27011	Vaidya makhija	201801023
2	SC101	27012	Sparsh shah	201801024
3	SC102	28001	Ghelani shah	201801025
4	SC103	28002	Hastin trivedi	201801026
5	SC104	28003	Dave dwivedi	201801027
6	SC105	28004	Om patel	201801028
7	SC106	28005	Deepak patel	201801029

**38. Find the names of students studying in sem 8 whose CPI is less than the average cpi of sem 8.**

SQL - select s\_name,cpi from Student14 where sem=8 and cpi<(select avg\_cpi from semwise\_cpi(8));

	s_name	cpi
1	Tyree Booth	8.73
2	Tobias Callahan	8.76
3	Amir Garrett	8.79
4	Albert Vazquez	8.82
5	Giana Carrillo	8.85
6	Heidy Tanner	8.88
7	Trace Morgan	8.91
8	Alia Golden	8.94
9	Esteban Sanchez	8.97
10	Lizeth Bryant	9
11	Kenyon Wilkerson	9.03

Messages  
Successfully run. Total query runtime: 92 msec.  
14 rows affected.

**39. Print all the number of students who got AB grade.**

SQL - create view FREQ2 as

select s\_id,count(grade)

as freq from grades1 where grade = 'AB' group by s\_id;

select \* From FREQ2;

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, the 'grades1' table is selected. In the main query editor window, the following SQL code is written:

```
1 set search_path to svdb
2
3 create view FREQ2 as
```

Below the code, the 'Data Output' tab is selected, showing the results of the query:

s_id	freq
201801018	1
201801027	1
201901026	3
201801001	1
201801007	1
201801006	1
202001013	1
202001032	1
201801011	1
202001002	1
201801022	1
201801012	1
201801013	1
202001025	1
202001022	1

**40. Display the number of 'BB' grades received by every student in the batch.**

SQL - select s\_id,count(grade) as no\_BB from Grades14 where grade='BB' group by s\_id;

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, the 'Grades14' table is selected. In the main query editor window, the following SQL code is written:

```
1 set search_path to mydata
2
3 select s_id,count(grade) as no_BB from Grades14 where grade='BB' group by s_id;
```

Below the code, the 'Data Output' tab is selected, showing the results of the query:

s_id	no_BB
201801019	1
201801029	1
201801018	1
201801027	1
202001004	1
202001021	1
201801009	1
201801007	1
201901012	3
201901027	3
202001013	1
202001031	1
201801022	1

- 41. Find the highest grade that every student has attained. For example, if a student ID 201901455 has got AA, AB, and BB grades in 3 different courses then display 201901455 student ID and AA grades and likewise for other student IDs.**

SQL - select student.s\_id,min(grades1.grade) as maximum\_grade  
from student natural join grades1  
where student.s\_id=grades1.s\_id  
group by student.s\_id

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with tables like student, courseinstructor1, course1, etc. The main area shows a query editor with the following SQL code:

```
1 set search_path to svdb
2
3 select student.s_id,min(grades1.grade) as maximum_grade
4 from student natural join grades1
5 where student.s_id=grades1.s_id
6 group by student.s_id
7
```

The results are displayed in a Data Output table:

s_id	maximum_grade
1	BB
2	AA
3	AA
4	AA
5	AB
6	AA
7	AA
8	AA
9	CC
10	AA
11	AB
12	AB

## **SECTION 7**

## **Front-End Development**

## OVERVIEW

- We did the complete coding of the database in Java Programming Language. We used JDBC connection to connect to the PostgreSQL database and completely coded in Eclipse IDE for Java Developers.
- We have created a menu driven program using else-if ladder which will continue to execute till you enter exit

## JAVA CODE

```
*****  
/* CASE STUDY :: E-Learning Management System */  
*****  
  
package com.P;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.InputStream;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.math.BigDecimal;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.ResultSetMetaData;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Scanner;  
import java.awt.Desktop;  
import java.io.File; // Import the File class  
import java.io.FileNotFoundException; // Import this class to handle errors  
import java.util.Scanner; // Import the Scanner class to read text files  
  
public class main  
{  
  
    private final String url = "jdbc:postgresql://localhost/201901206_LAB_9";  
    private final String user = "postgres";  
    private final String password = "admin";
```

```

/* function to connect to the postgresql database */
public Connection connect()
{
    Connection conn = null;
    try {
        conn = DriverManager.getConnection(url, user, password);
        System.out.println("Connected to the PostgreSQL server successfully.");
    }
    catch (SQLException e)
    {
        System.out.println(e.getMessage());
    }

    return conn;
}

/* This function inserts the data into the respective Table */
/*
In case of invalid entries,the triggers made corresponding to the schemas
will raise the exception
*/
/* Here we passed argument c in functions so that only authorized people can perform changes in the database */
/* Only Instructor can make changes to the lecture link table and grades table ,rest other changes in tables excluding these two can only be
performed by admin */
/* Student cannot make any changes to the database */

public void Insertion(String str, int c)
{
    if(str.compareTo("student") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            String q = "insert into mydata.Student14(s_id,s_name,s_email,sem,cpi) values(?,?,?,?,?)";

            PreparedStatement pst = conn.prepareStatement(q);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter student ID of student");
            BigDecimal ID = in.nextBigDecimal();
            System.out.println("Enter name of student");
            String name = br.readLine();
            System.out.println("Enter email address of student");
            String email = br.readLine();
            System.out.println("Enter current semester of student ");
            int sem = in.nextInt();
            System.out.println("Enter CPI of student");
            float cpi = in.nextFloat();
        }
    }
}

```

```

        pst.setBigDecimal(1, ID);
        pst.setString(2, name);
        pst.setString(3, email);
        pst.setInt(4, sem);
        pst.setFloat(5, cpi);

        pst.executeUpdate();
        System.out.println("student details successfully inserted");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

else if (str.compareTo("admin") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        String q = "insert into mydata.Admin14(a_name,a_email,dept) values(?, ?, ?)";

        PreparedStatement pst = conn.prepareStatement(q);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter name of the person ");
        String name = br.readLine();
        System.out.println("Enter email of the person");
        String email = br.readLine();
        System.out.println("Enter the department");
        String dept = br.readLine();

        pst.setString(1, name);
        pst.setString(2, email);
        pst.setString(3, dept);

        pst.executeUpdate();
        System.out.println("admin details successfully inserted");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

else if (str.compareTo("course") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

```

```

Connection conn = DriverManager.getConnection(url, user, password);
String q = "insert into mydata.Course14(c_id,c_name,credits) values(?, ?, ?)";

PreparedStatement pst = conn.prepareStatement(q);
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
Scanner in = new Scanner(System.in);

System.out.println("Enter Course ID of the course ");
String cid = br.readLine();
System.out.println("Enter the name of the course");
String cname = br.readLine();
System.out.println("Enter total credits of the course");
float credits = in.nextFloat();

pst.setString(1, cid);
pst.setString(2, cname);
pst.setFloat(3, credits);

pst.executeUpdate();
System.out.println("course details successfully inserted");
}

catch (Exception e)
{
    e.printStackTrace();
}

}

else if (str.compareTo("courseinstructor") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        String q = "insert into mydata.Courseinstructor14(i_id,c_id) values(?, ?)";

        PreparedStatement pst = conn.prepareStatement(q);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Instructor ID of the instructor");
        String iid = br.readLine();
        System.out.println("Enter the course ID that you are going to teach");
        String cid = br.readLine();

        pst.setString(1, iid);
        pst.setString(2, cid);

        pst.executeUpdate();
        System.out.println("course details successfully inserted");
    }
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
    else if (str.compareTo("grades") == 0 && c == 3)
    {

        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            String q = "insert into mydata.Grades14(id,s_id,c_id,grade) values(?, ?, ?, ?)";

            PreparedStatement pst = conn.prepareStatement(q);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter ID");
            BigDecimal ID = in.nextBigDecimal();
            System.out.println("Enter the student ID of the student");
            BigDecimal SID = in.nextBigDecimal();
            System.out.println("Enter Course ID of the course");
            String cid = br.readLine();
            System.out.println("Enter appropriate grade assigned to the student for this course");
            String grade = br.readLine();

            pst.setBigDecimal(1, ID);
            pst.setBigDecimal(2, SID);
            pst.setString(3, cid);
            pst.setString(4, grade);

            pst.executeUpdate();
            System.out.println("grade details of student successfully inserted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
    else if (str.compareTo("instructor") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            String q = "insert into mydata.Instructor14(i_id,i_email,i_name) values(?, ?, ?)";

```

```

PreparedStatement pst = conn.prepareStatement(q);
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
Scanner in = new Scanner(System.in);

System.out.println("Enter Instructor ID of the instructor");
BigDecimal insID = in.nextBigDecimal();
System.out.println("Enter email address of the instructor");
String email = br.readLine();
System.out.println("Enter Name of Instructor ");
String namei = br.readLine();

pst.setBigDecimal(1, insID);
pst.setString(2, email);
pst.setString(3, namei);

pst.executeUpdate();
System.out.println("Instructor details of instructor successfully inserted");
}

catch (Exception e)
{
e.printStackTrace();
}

}

else if (str.compareTo("lecturelink") == 0 && c == 3)
{
try {

String url = "jdbc:postgresql://localhost/201901206_LAB_9";
String user = "postgres";
String password = "admin";

Connection conn = DriverManager.getConnection(url, user, password);
String q = "insert into mydata.leclinks14(c_id,l_day,l_link) values(?, ?, ?)";

PreparedStatement pst = conn.prepareStatement(q);
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
Scanner in = new Scanner(System.in);

System.out.println("Enter Course ID of a course");
String cid = br.readLine();
System.out.println("Enter the day the lecture of course is scheduled");
String day = br.readLine();
System.out.println("Enter the lecture link of the scheduled lecture ");
String link = br.readLine();

pst.setString(1, cid);
pst.setString(2, day);
pst.setString(3, link);

pst.executeUpdate();
System.out.println("lecture link details for the course successfully inserted");
}
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("ta") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            String q = "insert into mydata.Ta14(s_id,c_id) values(?,?)";

            PreparedStatement pst = conn.prepareStatement(q);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter Student ID of the Teaching Assistant");
            BigDecimal tID = in.nextBigDecimal();
            System.out.println("Enter course ID of the course that TA is assisting in ");
            String cid = br.readLine();

            pst.setBigDecimal(1, tID);
            pst.setString(2, cid);

            pst.executeUpdate();
            System.out.println("TA details successfully inserted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

}

```

```

/* Performs deletion in the tables */
public void Deletion(String str, int c)
{
    if(str.compareTo("admin") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the email address of the admin that you want to delete");
            String email = br.readLine();

            String q = "delete from mydata.Admin14 where a_email= ?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setString(1, email);
            pst.executeUpdate();
            System.out.println("admin details for the email address successfully deleted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

else if (str.compareTo("course") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter the Course ID of course you want to delete");
        String cid = br.readLine();

        String q = "delete from mydata.Course14 where c_id= ?";

        PreparedStatement pst = conn.prepareStatement(q);
        pst.setString(1, cid);
        pst.executeUpdate();
        System.out.println("course details for the corresponding course ID successfully deleted");
    }
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("courseinstructor") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the Instructor ID of the instructor that you want to delete");
            BigDecimal ID = in.nextBigDecimal();

            String q = "delete from mydata.Courseinstructor14 where i_id= ?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setBigDecimal(1, ID);
            pst.executeUpdate();
            System.out.println("course instructor details successfully deleted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("instructor") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the instructor ID of the instructor that you want to delete");
            BigDecimal ID = in.nextBigDecimal();

            String q = "delete from mydata.Instructor14 where i_id= ?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setBigDecimal(1, ID);
            pst.executeUpdate();
            System.out.println("Instructor details of the instructor ID successfully deleted");
        }
    }
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("lecturelink") == 0 && c == 3)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the course ID of the course for which you want to delete the link");
            String S1 = br.readLine();
            String q = "delete from mydata.leclinks14 where c_id= ?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setString(1, S1);
            pst.executeUpdate();
            System.out.println("lecture link details of the course ID entered are successfully deleted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

}

else if (str.compareTo("student") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter the Student ID of the Student that you want to delete");
        BigDecimal SID = in.nextBigDecimal();

        String q = "delete from mydata.Student14 where s_id= ?";

        PreparedStatement pst = conn.prepareStatement(q);
        pst.setBigDecimal(1, SID);
        pst.executeUpdate();
        System.out.println("Student details of the student ID are successfully deleted");
    }
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("ta") == 0 && c == 2)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the Student ID of the Teaching Assistant that you want to delete");
            BigDecimal tID = in.nextBigDecimal();

            String q = "delete from mydata.Ta14 where s_id= ?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setBigDecimal(1, tID);
            pst.executeUpdate();
            System.out.println("TA details corresponding to student ID tID are successfully deleted");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    else if (str.compareTo("grade") == 0 && c == 3)
    {
        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter the Student ID of the Student for which you want to delete grade detail");
            BigDecimal SID = in.nextBigDecimal();
            System.out.println("Enter the Course ID of course of which you want to delete grade details");
            String cid = br.readLine();

            String q = "delete from mydata.grades14 where s_id= ? and c_id=?";

            PreparedStatement pst = conn.prepareStatement(q);
            pst.setBigDecimal(1, SID);

```

```

        pst.setString(2, cid);
        pst.executeUpdate();

        System.out.println("Grade details of the student ID for the course are successfully deleted");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

}

/* This function updates the already present values. Updation of data is done by authorized personnel only */
public void Updation(String str, int c)
{
    if(str.compareTo("grades") == 0 && c == 3)
    {

        try {
            String url = "jdbc:postgresql://localhost/201901206_LAB_9";
            String user = "postgres";
            String password = "admin";

            Connection conn = DriverManager.getConnection(url, user, password);
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Scanner in = new Scanner(System.in);

            System.out.println("Enter ID");
            BigDecimal ID = in.nextBigDecimal();
            System.out.println("Enter the student ID of the student ");
            BigDecimal SID = in.nextBigDecimal();
            System.out.println("Enter Course ID of the course you want to update");
            String cid = br.readLine();
            System.out.println("Enter appropriate grade that you want to update to the student for this course");
            String grade = br.readLine();

            String q = " update mydata.grades14 SET c_id = ?,grade =? WHERE id = ? and s_id = ? ";
            PreparedStatement pst = conn.prepareStatement(q);

            pst.setString(1, cid);
            pst.setString(2, grade);
            pst.setBigDecimal(3, ID);
            pst.setBigDecimal(4, SID);

            pst.executeUpdate();

            System.out.println("Grade details of the student ID for the course are successfully updated");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

```

else if (str.compareTo("courseinstructor") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Instructor ID of the instructor");
        BigDecimal iID = in.nextBigDecimal();
        System.out.println("Enter Course ID of the course you want to update");
        String cid = br.readLine();

        String q = " update mydata.courseinstructor14 SET c_id = ? WHERE i_id = ?";
        PreparedStatement pst = conn.prepareStatement(q);

        pst.setString(1, cid);
        pst.setBigDecimal(2, iID);

        pst.executeUpdate();

        System.out.println("course instructor details of the instructor successfully updated");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

else if (str.compareTo("admin") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter name of admin");
        String name = br.readLine();
        System.out.println("Enter the email address of the admin");
        String email = br.readLine();
        System.out.println("Enter the department of the admin");
        String dept = br.readLine();
    }
}

```

```

String q = " update mydata.admin14 SET dept = ?, WHERE a_name = ? and a_email = ? ";
PreparedStatement pst = conn.prepareStatement(q);

pst.setString(1, dept);
pst.setString(2, name);
pst.setString(3, email);

pst.executeUpdate();

System.out.println("admin details of admin are successfully updated");
}

catch (Exception e)
{
    e.printStackTrace();
}

}

else if (str.compareTo("lecturelink") == 0 && c == 3)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);

        System.out.println("Enter the course ID");
        String cid = br.readLine();
        System.out.println("Enter the day the lecture for this course is scheduled");
        String day = br.readLine();
        System.out.println("Enter the lecture link ");
        String link = br.readLine();

        String q = "update leclinks14 SET l_day=? ,l_link= WHERE c_id=?;";
        PreparedStatement pst = conn.prepareStatement(q);

        pst.setString(1, day);
        pst.setString(2, link);
        pst.setString(3, cid);

        pst.executeUpdate();
        System.out.println("lecture link details of the course are successfully inserted");
    }

    catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

```
else if (str.compareTo("student") == 0 && c == 2)
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);
        System.out.println("Enter ID of student");
        BigDecimal ID = in.nextBigDecimal();
        System.out.println("Enter name of student");
        String name = br.readLine();
        System.out.println("Enter email of student");
        String email = br.readLine();
        System.out.println("Enter sem of student");
        int sem = in.nextInt();
        System.out.println("Enter cpi of student");
        float cpi = in.nextFloat();
        String q = "update student14 SET cpi=? ,sem=? WHERE s_id=? and s_name=? and s_email=? ;";
        PreparedStatement pst = conn.prepareStatement(q);

        pst.setInt(2, sem);
        pst.setFloat(1, cpi);
        pst.setBigDecimal(3, ID);
        pst.setString(4, name);
        pst.setString(5, email);

        pst.executeUpdate();
        System.out.println("student details of the student are successfully updated");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

}
```

```

/* Display of results obtained according to query written */
    public void display()
{
    try {
        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
        String user = "postgres";
        String password = "admin";
        String q;

        Connection conn = DriverManager.getConnection(url, user, password);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Scanner in = new Scanner(System.in);
        q = br.readLine();

        PreparedStatement pst = conn.prepareStatement(q);
        ResultSet result = pst.executeQuery();
        ResultSetMetaData rsmd = result.getMetaData();

        /* To obtain number of columns in the resultant table of the query*/
        int cn = rsmd.getColumnCount();

        for (int i = 1; i <= cn; i++)
        {
            if (i == 1)
                System.out.print(" ( " + rsmd.getColumnName(i) + " , ");
            else if (i == cn)
                System.out.print(rsmd.getColumnName(i) + " ) ");
            else
                System.out.print(rsmd.getColumnName(i) + " , ");
        }
        System.out.println();

        while (result.next())
        {
            int x = 1;

            while (x <= cn)
            {
                if (x == 1)
                    System.out.print("(" + result.getString(x) + " , ");
                else if (x == cn)
                    System.out.print(result.getString(x) + " ) ");
                else
                    System.out.print(result.getString(x) + " , ");
                x++;
            }
            System.out.println();
        }

    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

```

/* Main function */
public static void main(String[] args)
{
    main app = new main();
    app.connect();      // to establish the connection with postgresql database
    Scanner p = new Scanner(System.in);
    int c;
    System.out.println("Enter the number that you belong to among these choices :");
    System.out.println("1 -- Student");
    System.out.println("2 -- Admin");
    System.out.println("3 -- Instructor");
    c = p.nextInt();
    int n;
    do {
        System.out.println("Enter a value among the given choices:");
        System.out.println(" 1 -- Insertion of data");
        System.out.println(" 2 -- Deletion of data ");
        System.out.println(" 3 -- Updation of existing data");
        System.out.println(" 4 -- Display the data ");
        System.out.println(" 5 -- Terminate ");

        Scanner a = new Scanner(System.in);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        n = a.nextInt();

        if (n == 1 && c != 1)
        {
            main tut = new main();
            String s;
            Scanner scan = new Scanner(System.in);
            System.out.println("In which Table,you want to perform Insertion");
            System.out.println("It should be one of the following 8 tables");
            System.out.println(" 1 -- student");
            System.out.println(" 2 -- course ");
            System.out.println(" 3 -- courseinstructor");
            System.out.println(" 4 -- grades ");
            System.out.println(" 5 -- lecturelink");
            System.out.println(" 6 -- instructor ");
            System.out.println(" 7 -- admin");
            System.out.println(" 8 -- ta ");
            s = scan.nextLine();
            app.Insertion(s, c);
        }
        else if (n == 2 && c != 1)
        {
            main del = new main();
            String input;
            Scanner scan = new Scanner(System.in);
            System.out.println("In which table,you want to delete the record");
            System.out.println("It should be one of the following 8 tables");
            System.out.println(" 1 -- student");
            System.out.println(" 2 -- course ");
            System.out.println(" 3 -- courseinstructor");
            System.out.println(" 4 -- grades ");
            System.out.println(" 5 -- lecturelink");
        }
    }
}

```

```

        System.out.println(" 6 -- instructor ");
        System.out.println(" 7 -- admin");
        System.out.println(" 8 -- ta ");

        input = scan.nextLine();
        app.Deletion(input, c);
    }

    else if (n == 3 && c != 1) {
        main del = new main();
        String input;
        Scanner scan = new Scanner(System.in);

        System.out.println("In which table,you want to update the record");
        System.out.println("It should be one of the following 8 tables");
        System.out.println(" 1 -- student");
        System.out.println(" 2 -- course ");
        System.out.println(" 3 -- courseinstructor");
        System.out.println(" 4 -- grades ");
        System.out.println(" 5 -- lecturelink");
        System.out.println(" 6 -- instructor ");
        System.out.println(" 7 -- admin");
        System.out.println(" 8 -- ta ");

        input = scan.nextLine();
        app.Updation(input, c);
    }

    else if (n == 4) {
        System.out.println("Write a query of your choice to display the data");
        app.display();
    }

    else
    {
        System.out.println("Invalid number entered");
    }

}

} while (n != 5);

System.out.println("Program terminated");
}
}

```

# RUNNING QUERIES

## 1. Print details of students scoring cpi>8.2.

The screenshot shows the Eclipse IDE interface with the Java Project DBMS workspace. The console tab displays the following output:

```
Connected to the PostgreSQL server successfully.
Enter the number that you belong to among these choices :
1 -- Student
2 -- Admin
3 -- Instructor
4
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
5 -- Terminate
4
Write a query of your choice to display the data
select * from mydata.student14 where cpi>8.2
| (s_id , s_name , s_email , sem , cpi )
| (201801081 , Hastin_Dave , Hastin_Dave@educat.in , 2 , 10 )
| (201801082 , Sparsh_ghelani , Sparsh_ghelani@educat.in , 2 , 10 )
| (201801083 , Rishi_Kataria , Rishi_Kataria@educat.in , 2 , 10 )
| (201801084 , Lauryn_Kennedy , Lauryn_Kennedy@educat.in , 6 , 8.22 )
| (201801014 , Jakobe_Townsend , Jakobe_Townsend@educat.in , 6 , 8.25 )
| (201801015 , Lukas_Shields , Lukas_Shields@educat.in , 6 , 8.28 )
| (201801016 , Laura_Meyer , Laura_Meyer@educat.in , 6 , 8.31 )
| (201801017 , Rebekah_Manning , Rebekah_Manning@educat.in , 6 , 8.34 )
| (201801018 , Gretchen_Santiago , Gretchen_Santiago@educat.in , 6 , 8.37 )
| (201801019 , Ayden_Cunningham , Ayden_Cunningham@educat.in , 6 , 8.4 )
| (201801020 , Cynthia_Short , Cynthia.Short@educat.in , 6 , 8.43 )
| (201801021 , Amir_Chandler , Amir_Chandler@educat.in , 6 , 8.46 )
| (201801022 , Makaila_Teccari , Makaila_Teccari@educat.in , 6 , 8.49 )
| (201801023 , Jimmy_Juarez , Jimmy_Juarez@educat.in , 6 , 8.52 )
| (201801024 , Camille_Mcgrath , Camille_Mcgrath@educat.in , 6 , 8.55 )
| (201801025 , Riley_Owens , Riley_Owens@educat.in , 6 , 8.58 )
| (201801026 , Rishi_Kelley , Rishi_Kelley@educat.in , 6 , 8.61 )
| (201801027 , Jane_Duarte , Jane_Duarte@educat.in , 6 , 8.64 )
| (201801028 , Frank_Kennedy , Frank_Kennedy@educat.in , 6 , 8.67 )
| (201801029 , Hayden_Phan , Hayden_Phan@educat.in , 6 , 8.7 )
| (201701001 , Tyree_Booth , Tyree_Booth@educat.in , 8 , 8.73 )
| (201701002 , Tobias_Callahan , Tobias_Callahan@educat.in , 8 , 8.76 )
```

## 2. Find Student ID of students who got AA grade in any IT course.

The screenshot shows the Eclipse IDE interface with the Java Project DBMS workspace. The console tab displays the following output:

```
(201701029 , Heath_Keller , Heath_Keller@educat.in , 8 , 9.57 )
| (202001033 , Manish_Pathak , manish_pathak@educat.ac.in , 6 , 8.45 )
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
5 -- Terminate
4
Write a query of your choice to display the data
select * from mydata.student14 natural join mydata.grades14 where mydata.student14.s_id=mydata.grades14.s_id and grade='AA' and c_id like '%IT%'
| (s_id , s_name , s_email , sem , cpi , id , c_id , grade )
| (202001081 , Hastin_Dave , Hastin_Dave@educat.in , 2 , 10 , 1 , IT102 , AA )
| (202001086 , Dhaval_Vaidya , Dhaval_Vaidya@educat.in , 2 , 6 , 6 , IT102 , AA )
| (202001011 , Katie_Acosta , Katie_Acosta@educat.in , 2 , 6.3 , 11 , IT102 , AA )
| (202001016 , Lola_Caldwell , Lola_Caldwell@educat.in , 2 , 6.45 , 16 , IT102 , AA )
| (202001021 , Craig_McGuire , Craig_McGuire@educat.in , 2 , 6.6 , 21 , IT102 , AA )
| (202001026 , Jamison_Green , Jamison.Green@educat.in , 2 , 6.75 , 26 , IT102 , AA )
| (202001031 , Jaylon_Valentine , Jaylon.Valentine@educat.in , 2 , 6.9 , 31 , IT102 , AA )
| (201901005 , Ignacio_Willis , Ignacio_Willis@educat.in , 4 , 7.08 , 101 , IT104 , AA )
| (201901010 , Corinne_Colon , Corinne_Colon@educat.in , 4 , 7.23 , 106 , IT104 , AA )
| (201901015 , Connor_Winters , Connor_Winters@educat.in , 4 , 7.38 , 111 , IT104 , AA )
| (201901020 , Brendon_Todd , Brendon.Todd@educat.in , 4 , 7.53 , 116 , IT104 , AA )
| (201901025 , Terrence_Sampson , Terrence.Sampson@educat.in , 4 , 7.68 , 121 , IT104 , AA )
| (201901030 , Reece_Hodge , Reece_Hodge@educat.in , 4 , 7.83 , 126 , IT104 , AA )
| (201801005 , Maximus_Bowers , Maximus_Bowers@educat.in , 6 , 7.98 , 191 , IT106 , AA )
| (201801010 , Jada_Adams , Jada_Adams@educat.in , 6 , 8.13 , 196 , IT106 , AA )
| (201801015 , Lukas_Shields , Lukas_Shields@educat.in , 6 , 8.28 , 201 , IT106 , AA )
| (201801020 , Christopher_Knowles , Christopher_Knowles@educat.in , 6 , 8.33 , 206 , IT106 , AA )
| (201801025 , Riley_Owens , Riley_Owens@educat.in , 6 , 8.58 , 211 , IT106 , AA )
| (202001033 , Manish_Pathak , manish_pathak@educat.ac.in , 6 , 8.45 , 274 , IT106 , AA )
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
5 -- Terminate
```

### 3. Insertion of values in the table.

The screenshot shows the Eclipse IDE interface with a Java application running. The console output is as follows:

```
Connected to the PostgreSQL server successfully.
Enter the number that you belong to among these choices :
1 -- Student
2 -- Admin
3 -- Instructor

Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data

1
In which Table,you want to perform Insertion
It should be one of the following 8 tables
1 -- student
2 -- course
3 -- courseinstructor
4 -- grades
5 -- lecturelink
6 -- instructor
7 -- admin
8 -- ta
student
Enter student ID of student
201801454
Enter name of student
Prakash Sheth
Enter email address of student
prakash_sheth@Educat.ac.in
Enter current semester of student
6
Enter CPI of student
8.17
student details successfully inserted
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
```

The screenshot shows the database query results in the Eclipse IDE's SQL perspective. The query is:

```
select * from mydata.student14 where s_id=201801454
```

The result is:

id	s_name	a_email	cpi
201801454	Prakash Sheth	prakash_sheth@Educat.ac.in	6

### 4. Deletion of values from table.

The screenshot shows the Eclipse IDE interface with a Java application running. The console output is as follows:

```
6 -- instructor
7 -- admin
8 -- ta

7
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data

2
In which table,you want to delete the record
It should be one of the following 8 tables
1 -- student
2 -- course
3 -- courseinstructor
4 -- grades
5 -- lecturelink
6 -- instructor
7 -- admin
8 -- ta
admin
Enter the email address of the admin that you want to delete
210040_Fin@Educat.ac.in
admin details for the email address successfully deleted
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data

4
Write a query of your choice to display the data
select * from mydata.admin14 where a_email = '210040_Fin@Educat.ac.in'
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
```

## 5. To ensure the triggers are working perfectly or not.

### Inserting values into admin

The screenshot shows the Eclipse IDE interface with a Java project named "Java\_Project\_DBMS". The code editor displays a main.java file with the following content:

```
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data

In which Table,you want to perform Insertion
It should be one of the following 8 tables
1 -- student
2 -- course
3 -- courseinstructor
4 -- grades
5 -- lecturelink
6 -- instructor
7 -- admin
8 -- ta
admin
Enter name of the person
nillesh_bhai
Enter email of the person
q2@q2
Enter the department
DAC
org.postgresql.util.PSQLException: ERROR: invalid email address entered or department does not exist in institution
Where: PL/pgSQL function mydata.admin_info() line 12 at RAISE
Enter a value among the given choices:
1 -- Insertion of data
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2674)
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:2364)
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:354)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgStatement.executeInternal(PgStatement.java:484)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.execute(PgStatement.java:484)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.executeWithFlags(PgPreparedStatement.java:162)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.executeUpdate(PgPreparedStatement.java:138)
at com.P.main.Insertion(main.java:122)
at com.P.main.main(main.java:849)

2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
```

A red box highlights the error message: "org.postgresql.util.PSQLException: ERROR: invalid email address entered or department does not exist in institution".

### Inserting values into grade table

The screenshot shows the Eclipse IDE interface with a Java project named "Java\_Project\_DBMS". The code editor displays a main.java file with the following content:

```
1
In which Table,you want to perform Insertion
It should be one of the following 8 tables
1 -- student
2 -- course
3 -- courseinstructor
4 -- grades
5 -- lecturelink
6 -- instructor
7 -- admin
8 -- ta
grades
Enter ID
500
Enter the student ID of the student
2020010111
Enter Course ID of the course
IT106
Enter appropriate grade assigned to the student for this course
FF
org.postgresql.util.PSQLException: ERROR: invalid grade entered
Where: PL/pgSQL function mydata.grading_info() line 9 at RAISE
Enter a value among the given choices:
1 -- Insertion of data
2 -- Deletion of data
3 -- Updation of existing data
4 -- Display the data
5 -- Terminate
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2674)
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:2364)
at org.postgresql.jdbc@42.3.1/org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:354)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgStatement.executeInternal(PgStatement.java:484)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.execute(PgStatement.java:484)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.executeWithFlags(PgPreparedStatement.java:162)
at org.postgresql.jdbc@42.3.1/org.postgresql.jdbc.PgPreparedStatement.executeUpdate(PgPreparedStatement.java:138)
at com.P.main.Insertion(main.java:233)
at com.P.main.main(main.java:856)
```

A red box highlights the error message: "org.postgresql.util.PSQLException: ERROR: invalid grade entered".