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

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

  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

  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

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

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 **Designation**: Head, Research Department
2) Mr. Rahul Mankad **Designation**: Head, UG committee
3) Ms. Shivani Joshi **Designation**: Head, PG committee
4) Mr. Amit Joshi **Designation:** Accounting Head
5) Mr. Kanan Sharma **Designation:** Registrar
6) Mr. Kiran Kataria **Designation**: Warden(Men's Hostel)

Contact Details: deepak_kabra@eit.ac.in
rahul_mankad@eit.ac.in
shivani_vaidya@eit.ac.in
amit_joshi@eit.ac.in
kanan_sharma@eit.ac.in
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 **Designation**: Head, Research Department
2. Mr. Rahul Mankad **Designation**: Head, UG committee
3. Ms. Shivani Vaidya **Designation**: Head, PG committee
4. Mr. Amit Joshi **Designation:** Accounting Head
5. Mr. Kanan Sharma **Designation:** Registrar
6. Mr. Kiran Kataria **Designation**: Warden(Men's Hostel)

Contact Details: deepak_kabra@eit.ac.in
rahul_mankad@eit.ac.in
shivani_vaidya@eit.ac.in
amit_joshi@eit.ac.in
kanan_sharma@eit.ac.in
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 **Designation**: Sr. professor(DIP)
2) Dr. Rahul Patel **Designation**: Professor(Machine Learning)
3) Dr. Ketul Joshi **Designation**: Professor(Embedded Design)

Contact Details: milan_kabra@eit.ac.in
rahul_patel@eit.ac.in
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 **Designation**: Sr. professor(DIP)
2. Dr. Rahul Patel **Designation**: Professor(Machine Learning)
3. Dr. Ketul Joshi **Designation**: Professor(Embedded Design)

Contact Details: milan_kabra@eit.ac.in
rahul_patel@eit.ac.in
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

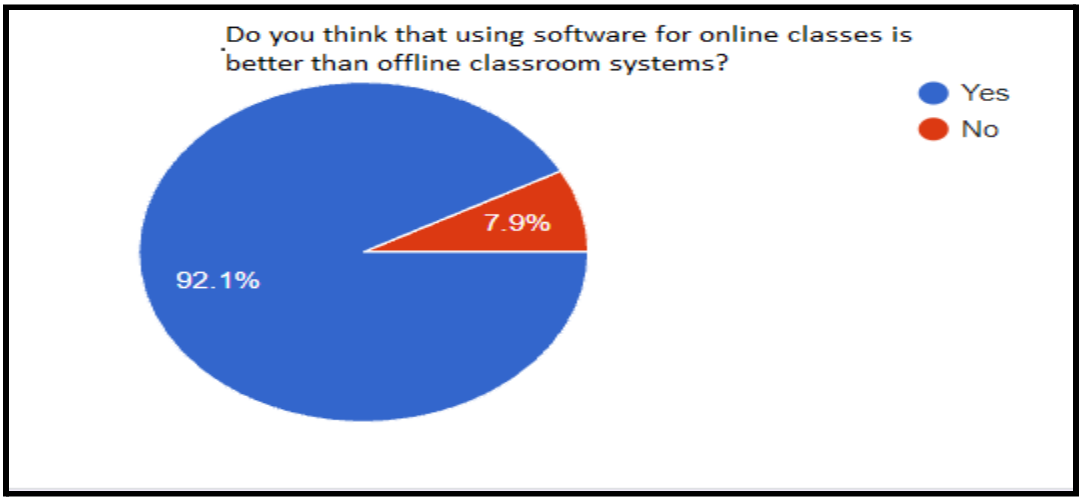
6. **Will it be better if you can track individual students' academics?**
   Yes/No

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

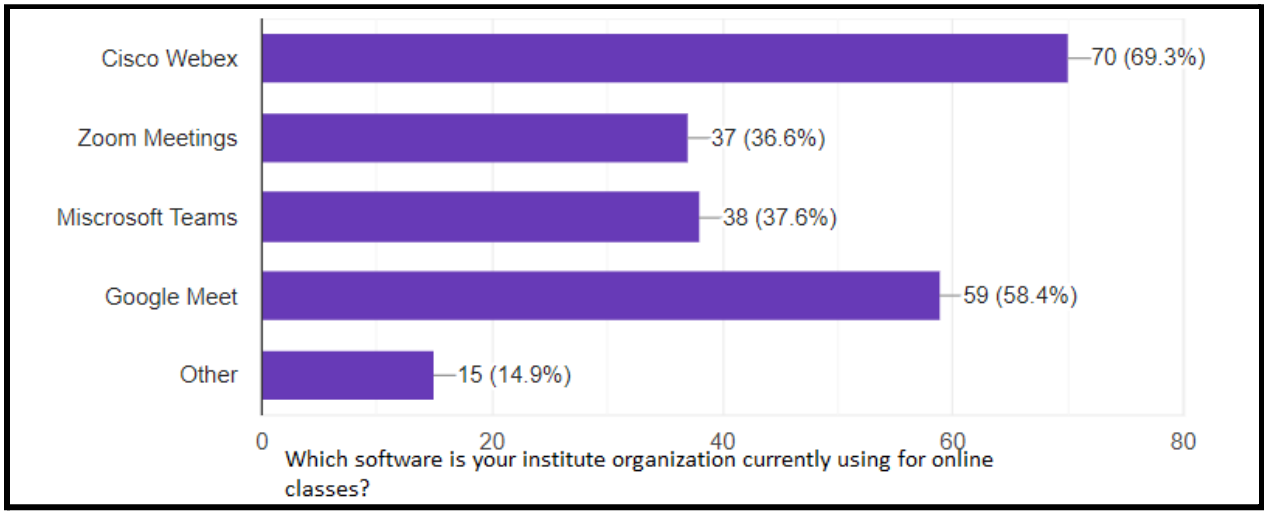
8. **Will you buy software that has all the above functionalities and is available at a reasonable price?**
   Yes/No/Maybe

Summary of Responses

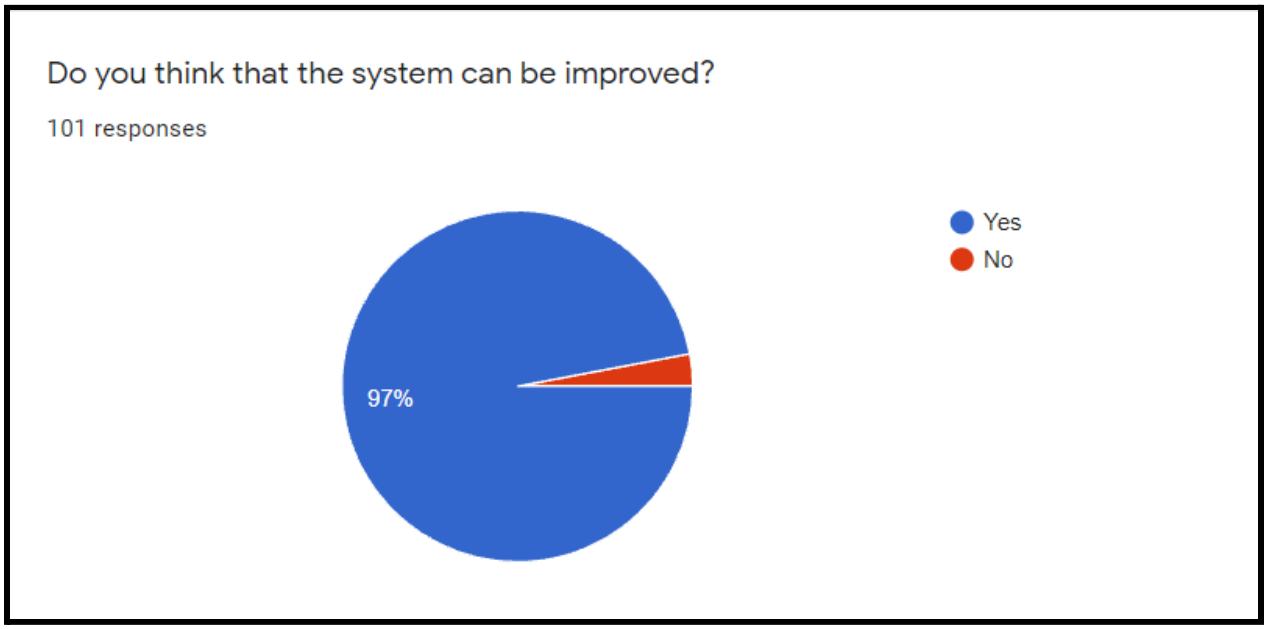
1. **Do you think software-oriented online classes are better than offline classroom systems?**
   92.1% of the people responded with Yes since they believe online classes are better than offline classroom systems.



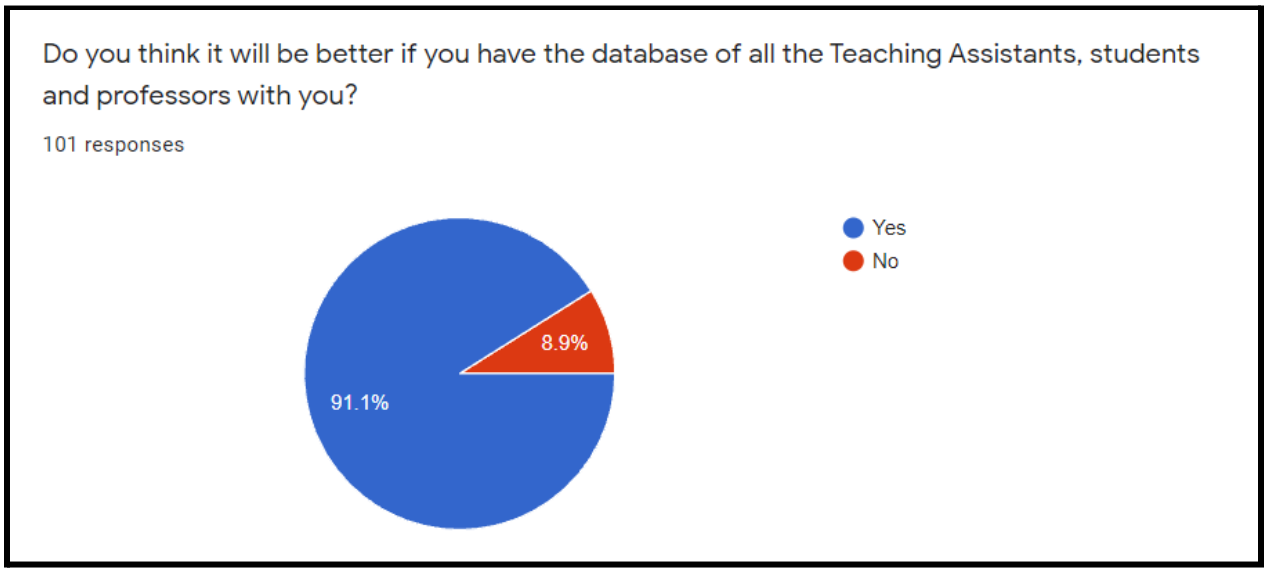
2. **Which software is your institute currently using for online classes?**
As a response to this question, we could conclude that Cisco Webex and google meet are widely used. As a third and fourth choice, zoom meet and Microsoft meet were pretty popular as well. Some organizations also used other Softwares. The detailed distribution is as follows:



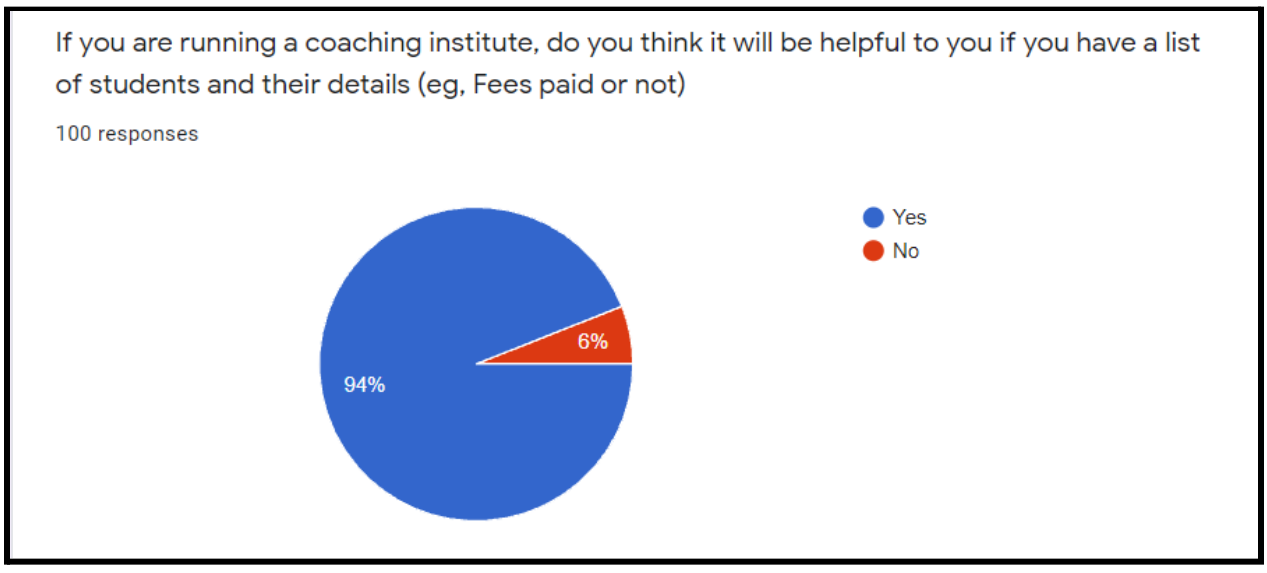
3. **Do you think that the current system can be improved?**
Most of the people answered with Yes since it is a natural tendency for people to expect a better user experience with the advancement of new technologies.



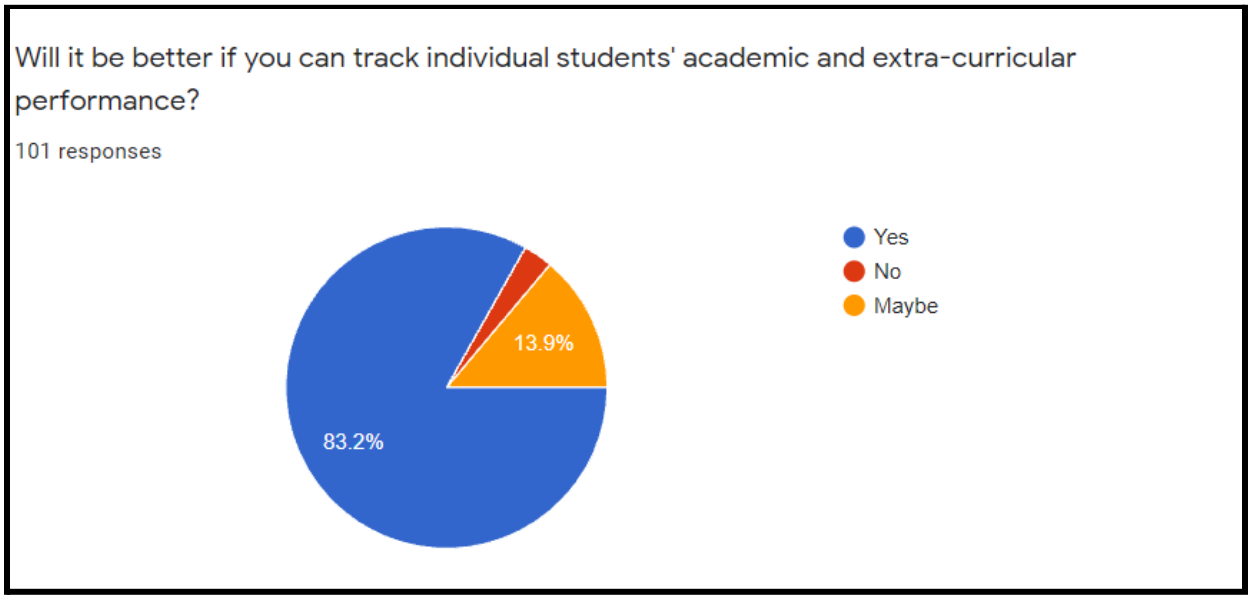
4. **Do you think it would be better if you have data of all the Professors/Teaching Assistants(TAs)/Students with you?**
   Most of the institutions responded with Yes since it is always helpful to have as much data as possible.



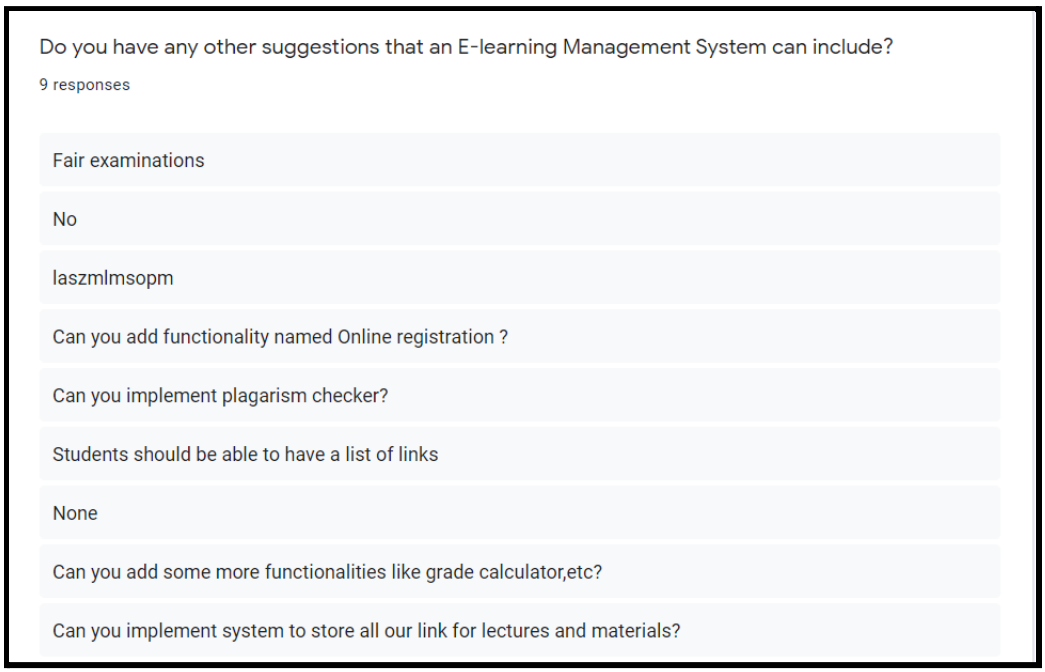
5. **If you are running a coaching institute, do you think it will be helpful to you if you have a list of students and their details?**
   Most of the coaching institutions responded with Yes since again it is helpful to have as much data as possible.



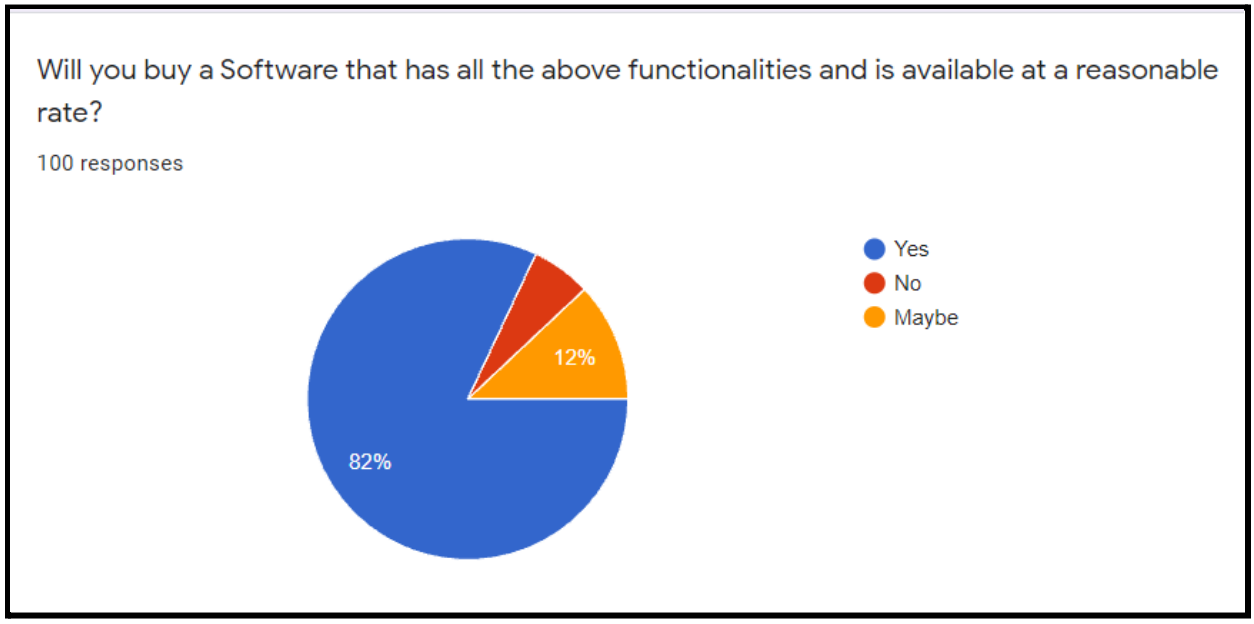
6. **Will it be better if you can track individual students' academics?**
   It would be better if you can track individual performance as it would also be better for students as the professors would be able to concentrate on individuals.



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?**
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

Objective	Technique	Subject(s)	Time Commitment
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

- **<u>Privileges</u>**

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

Nouns	Verbs
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

Candidate Entity Set	**Candidate Attribute Set**	**Candidate Relationship Set**
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	

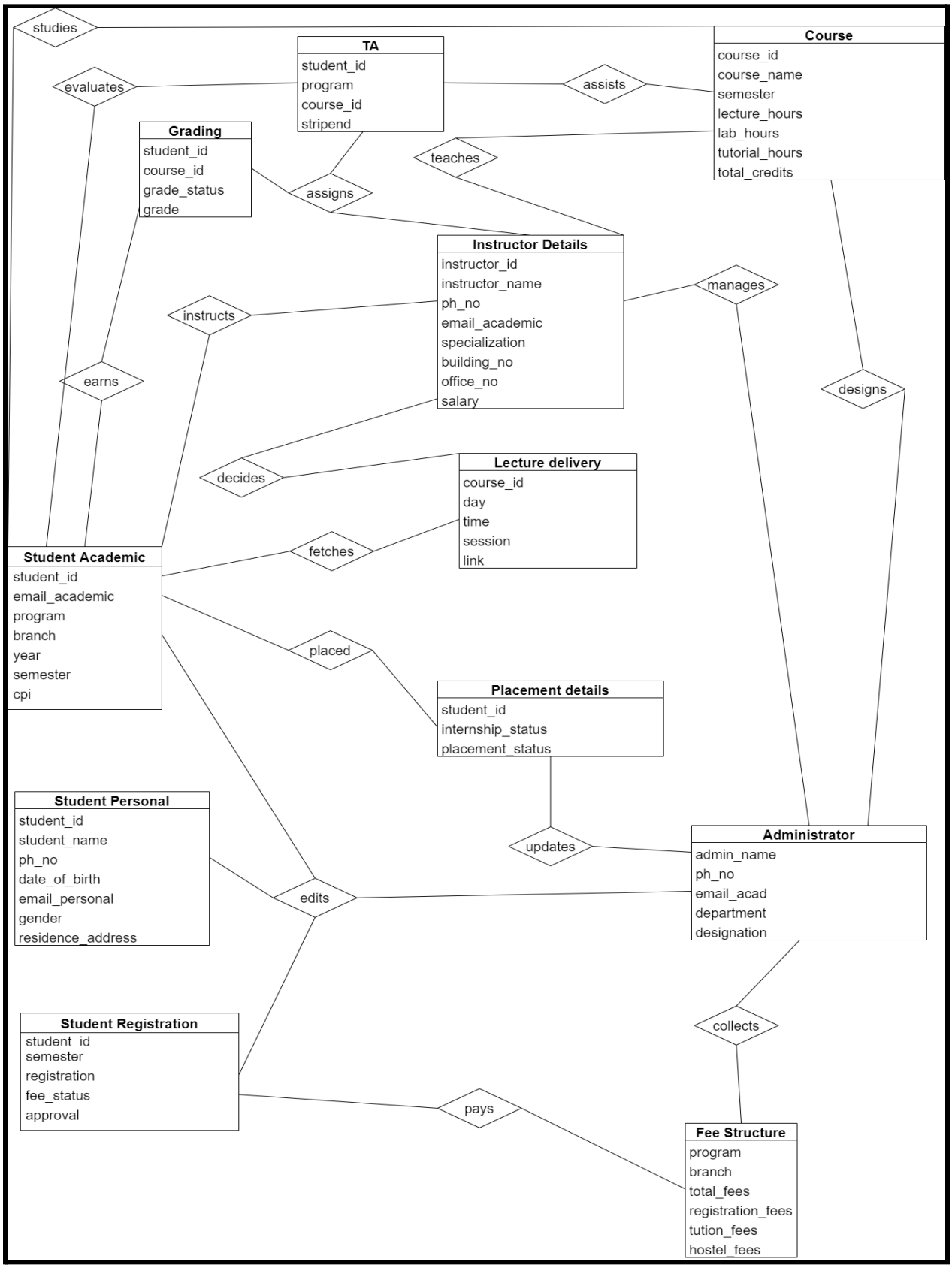
Table 3

Rejected Nouns and Verbs List

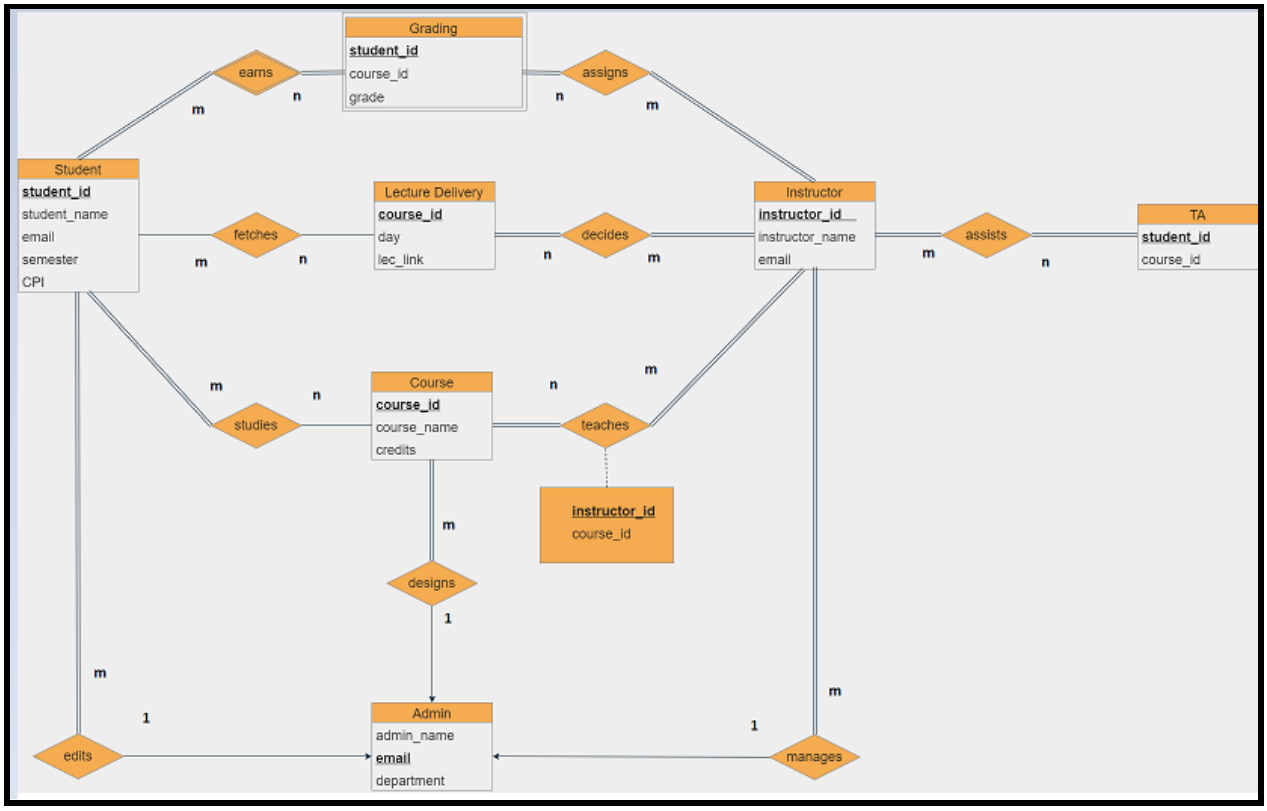
Nouns	Reject Reasons
Student (Studies under an Instructor)	Associations
Student (Resides)	Vague
Student (Registers)	Irrelevant
Student (Pays)	Vague
Student (Interns)	Vague
Student (Gets Placed)	Vague
Hostel (Accommodates)	Vague, General
Instructor (Instructs Students)	Associations
Instructor (Assigns 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


studies
TA
student_id
program
course_id
stripend
Course
course_id
course_name
semester
lecture_hours
lab_hours
tutorial_hours
total_credits
evaluates
assists
Grading
student_id
course_id
grade_status
grade
teaches
assigns
Instructor Details
instructor_id
instructor_name
ph_no
email_academic
specialization
building_no
office_no
salary
manages
instructs
earns
designs
decides
Lecture delivery
course_id
day
time
session
link
fetches
Student Academic
student_id
email_academic
program
branch
year
semester
cpi
placed
Placement details
student_id
internship_status
placement_status
Student Personal
student_id
student_name
ph_no
date_of_birth
email_personal
gender
residence_address
updates
Administrator
admin_name
ph_no
email_acad
department
designation
edits
collects
Student Registration
student_id
semester
registration
fee_status
approval
pays
Fee Structure
program
branch
total_fees
registration_fees
tution_fees
hostel_fees

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**

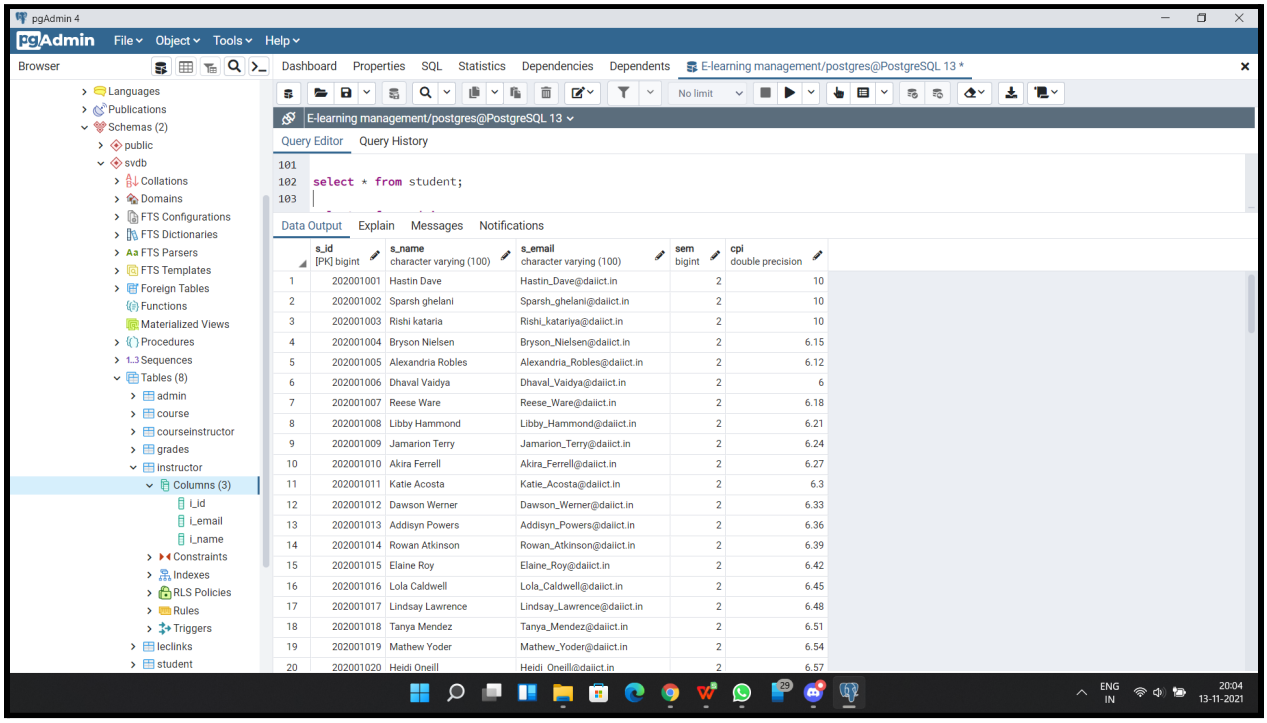
```
create or replace function admin_info()
returns trigger as $body$
begin
if((new.dept='Research' or new.dept='Registrar'
  or new.dept='Placement' or new.dept='Medical'
  or new.dept='Maintenance' or new.dept='Library'
  or new.dept='IT' or new.dept='Finance'
  or new.dept='Counselling' or new.dept='Academic')
   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**

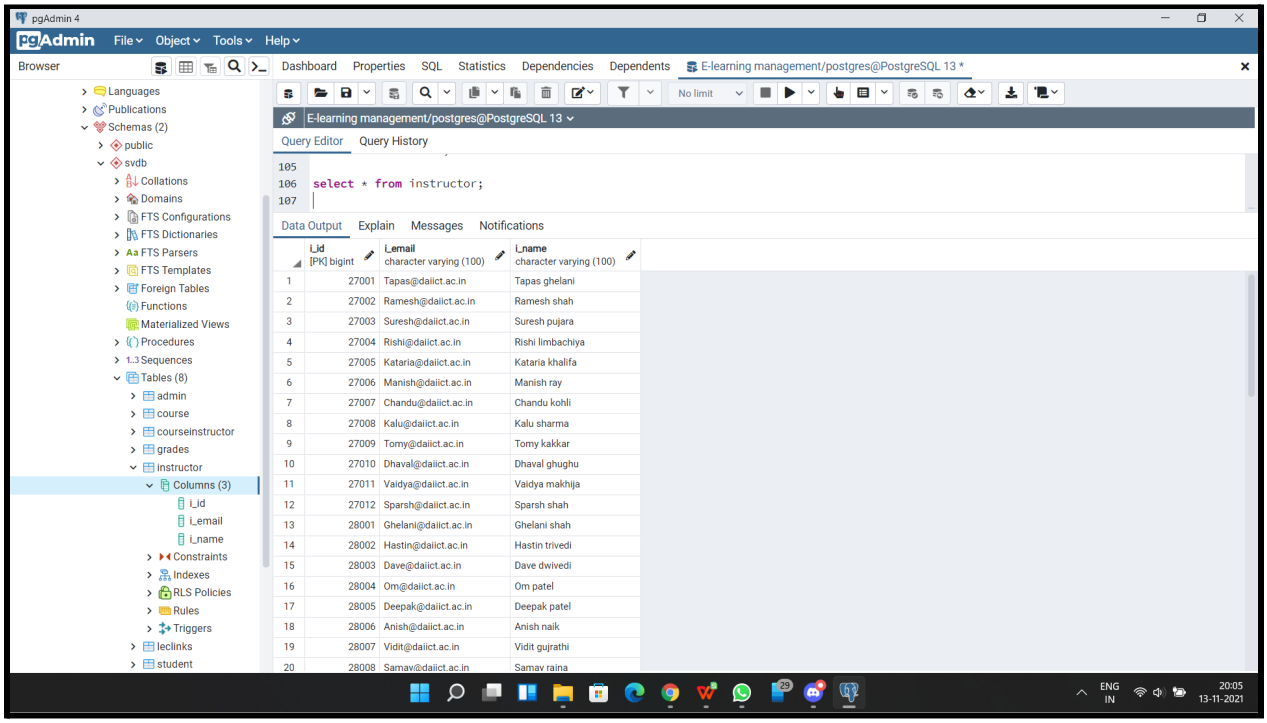
```
create or replace function instructor_info()
returns trigger as $body$
begin
if(new.i_email like '%Educat.ac.in%' and (new.i_id)>0) then
raise notice 'successfully entered instructor details with valid instructor id';
return new;
else
raise exception 'invalid email address entered or it does not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

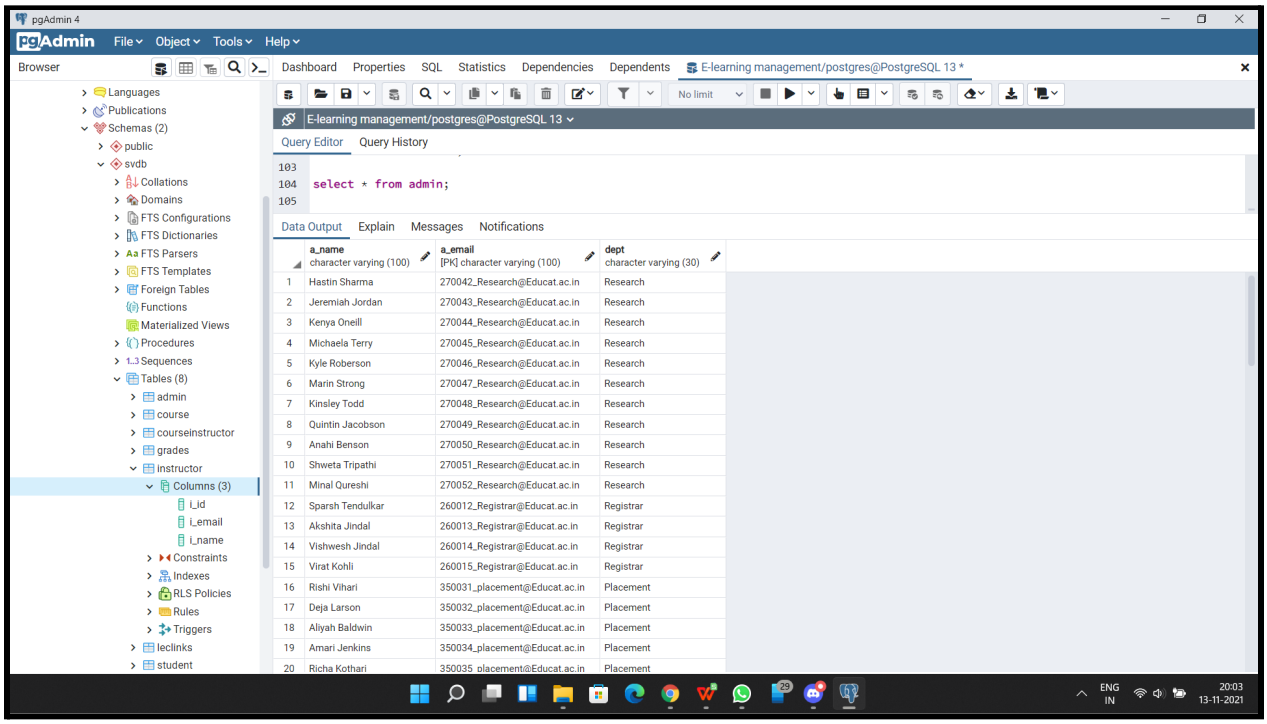
Student Details



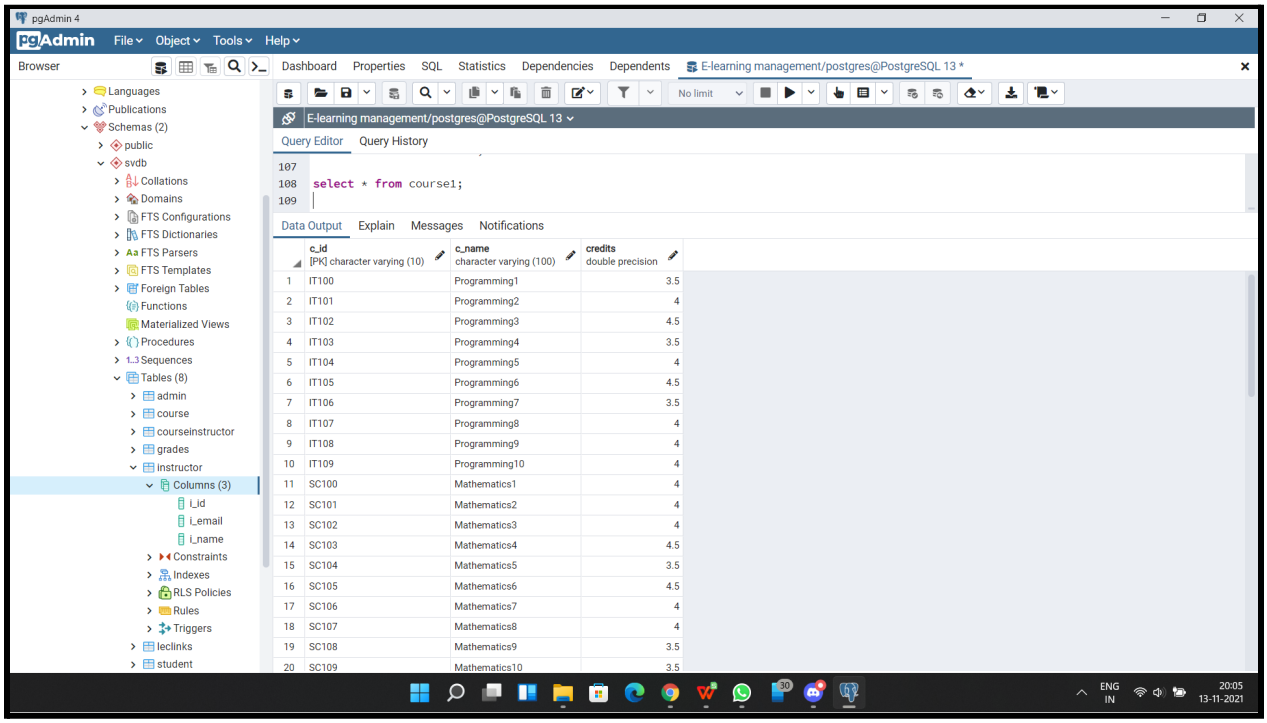
Instructor Details



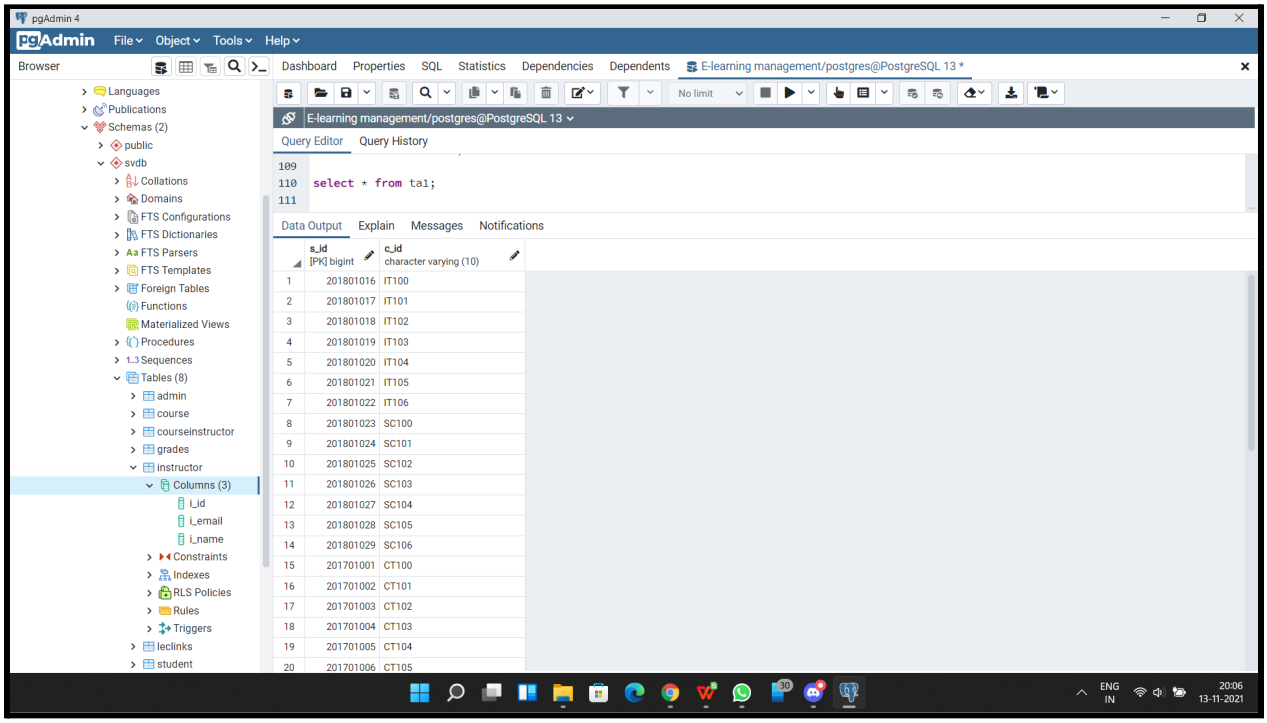
Admin Details



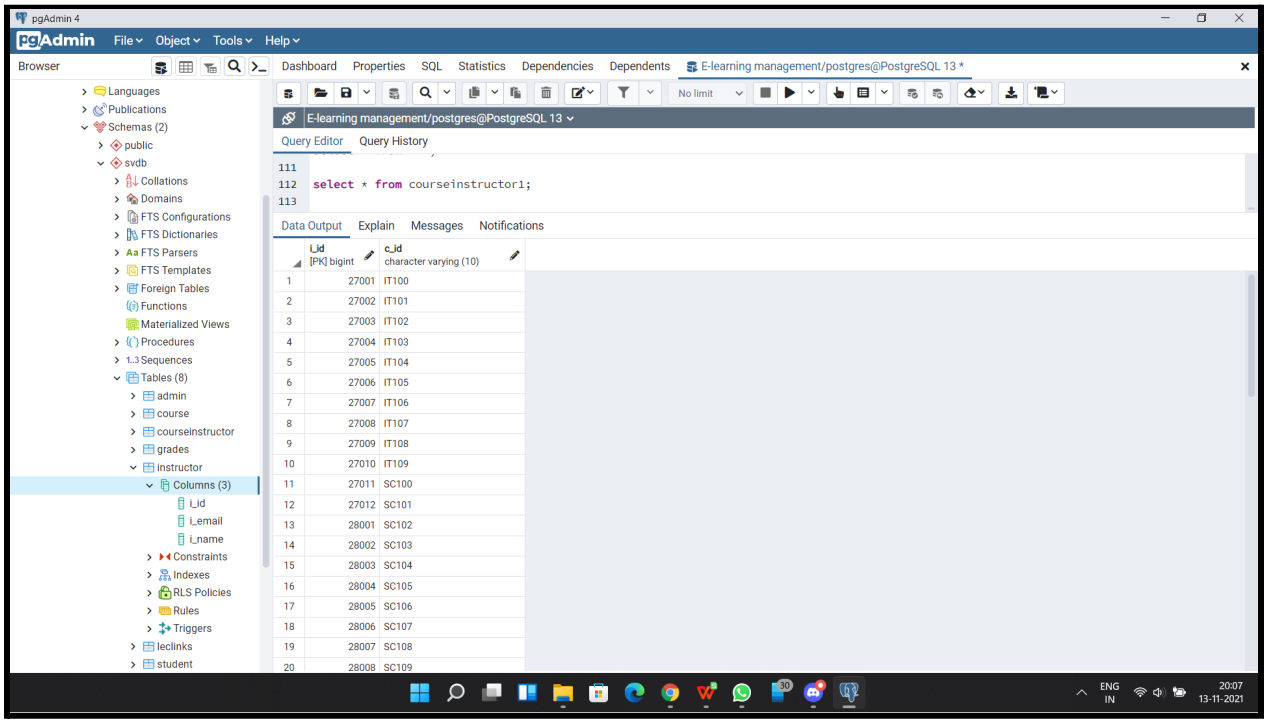
Course Details



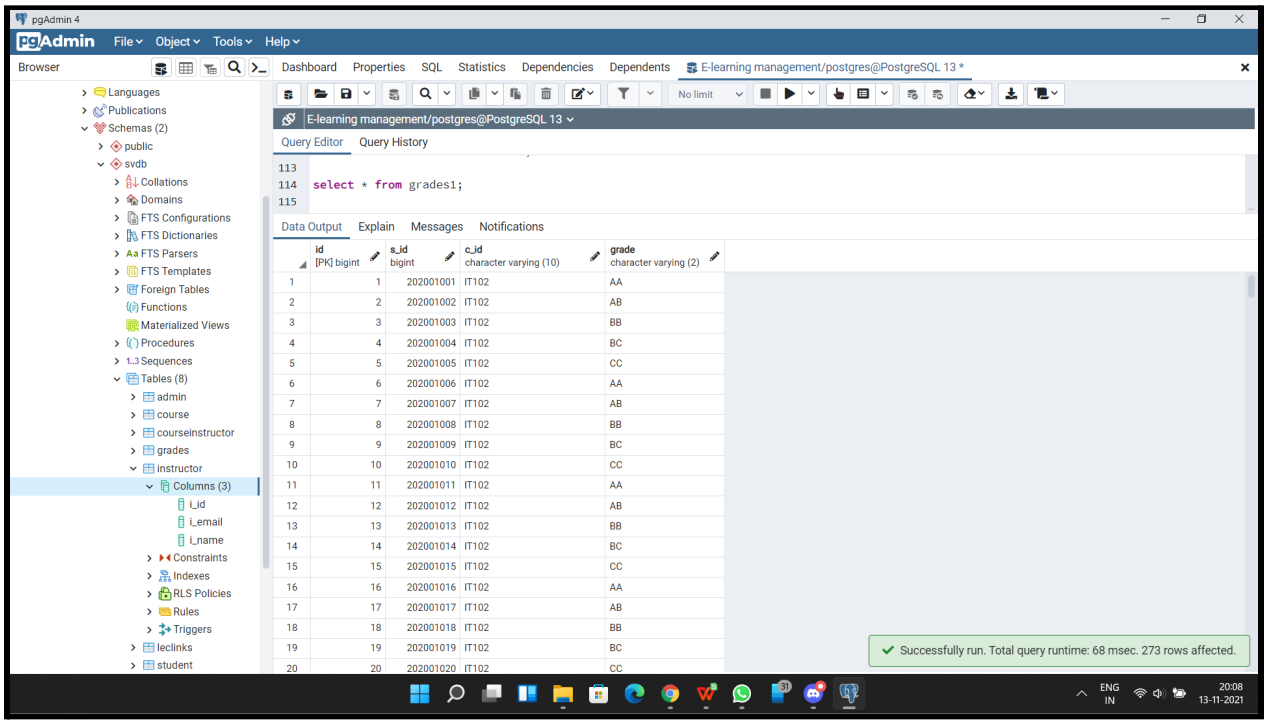
Teaching Assistant Details



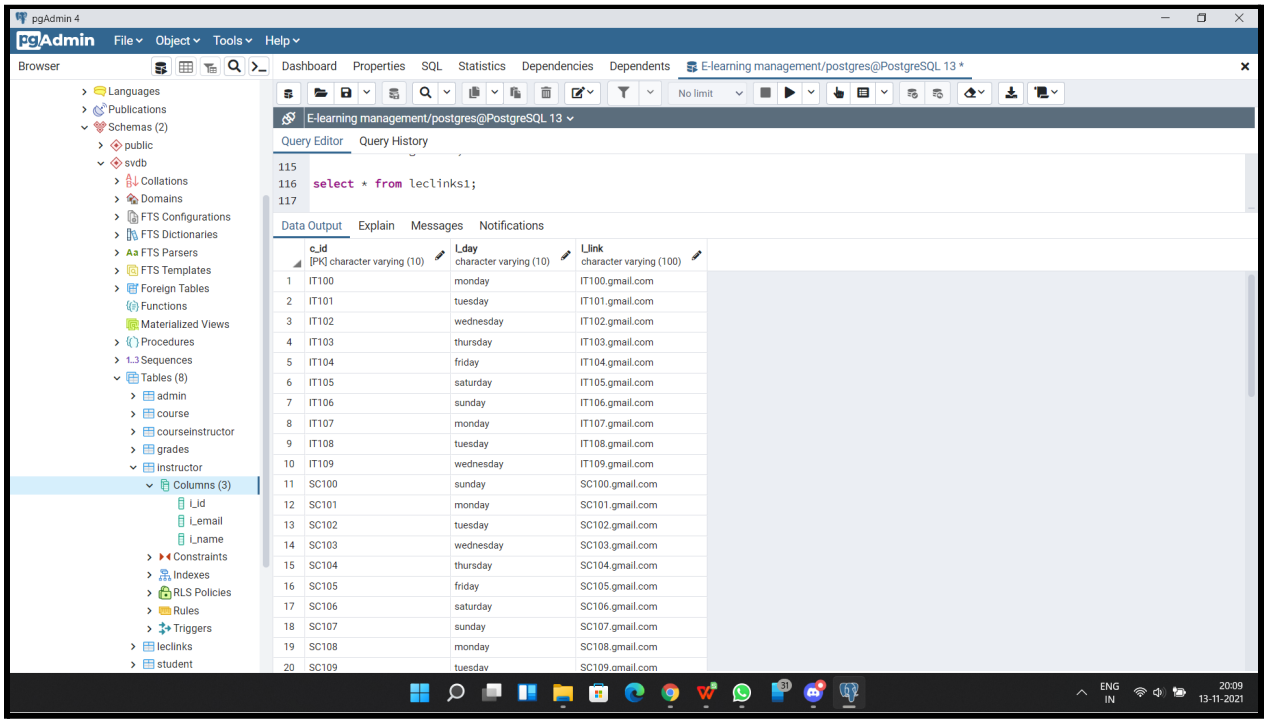
Course-Instructor Relationship



Grading Details

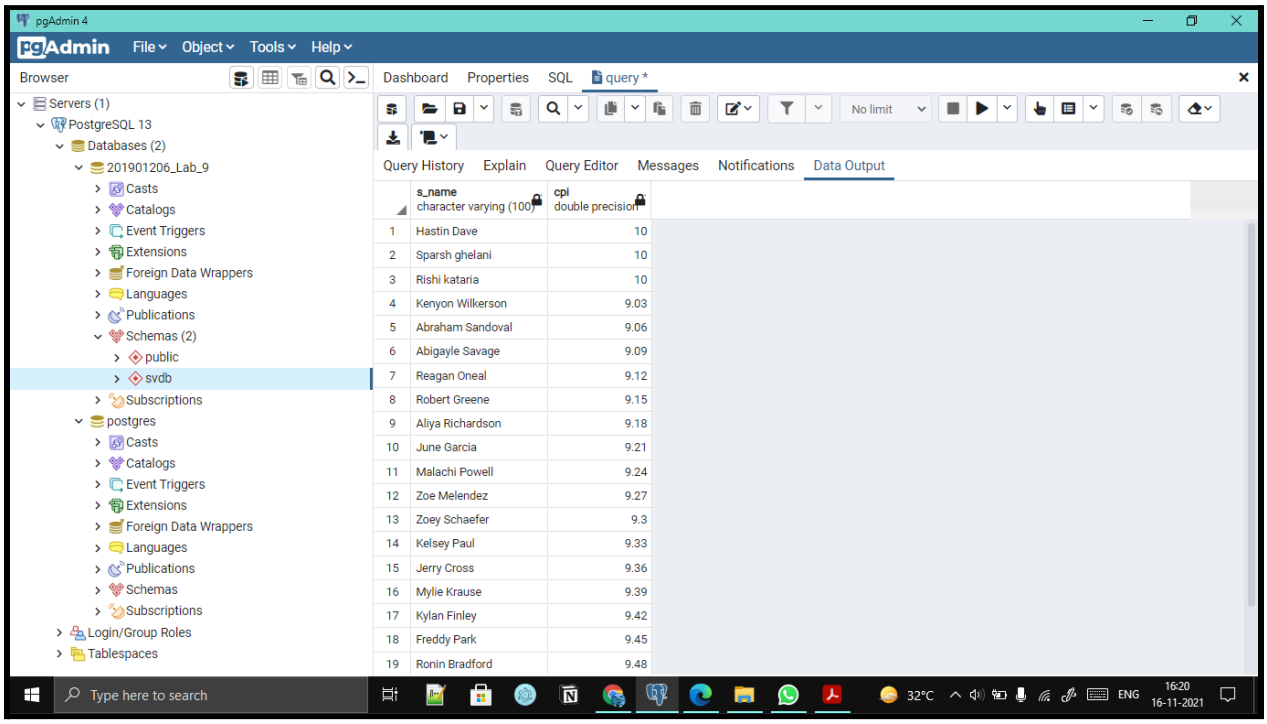


Lecture Delivery Details

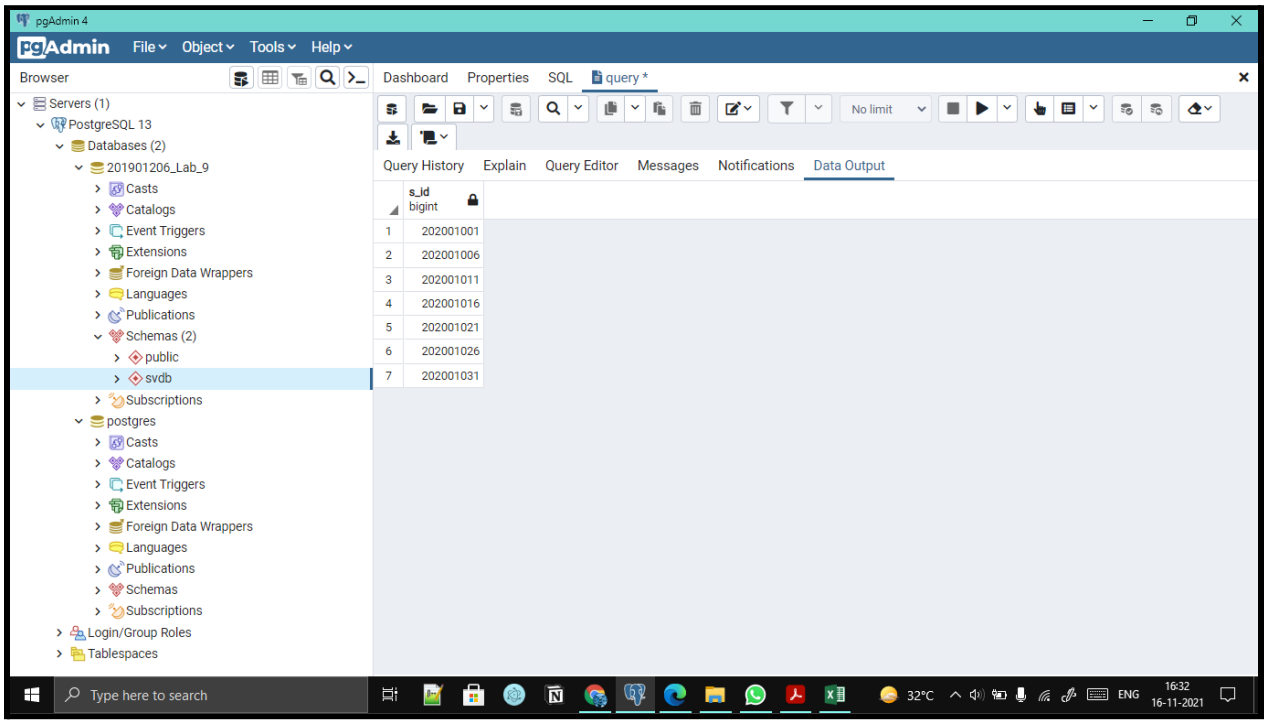


QUERIES

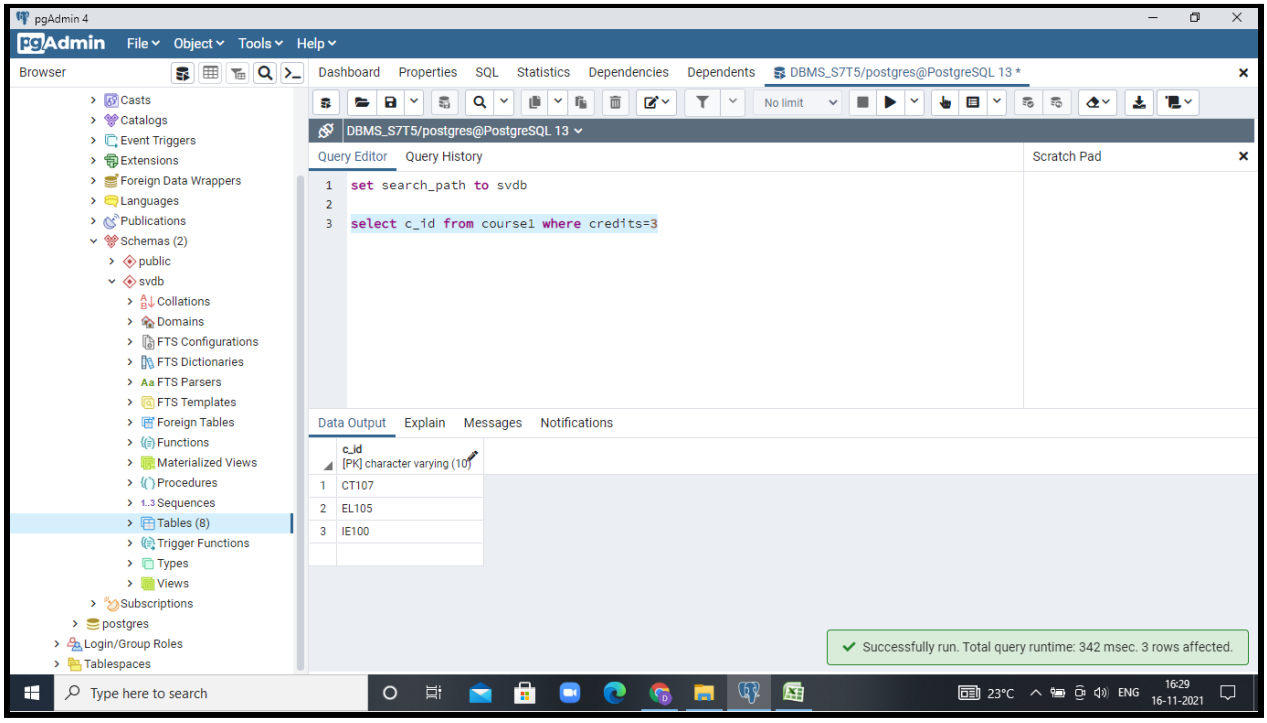
1. **View names of the students with CPI > 9.**
   SQL - select s_name,cpi from student where cpi>9



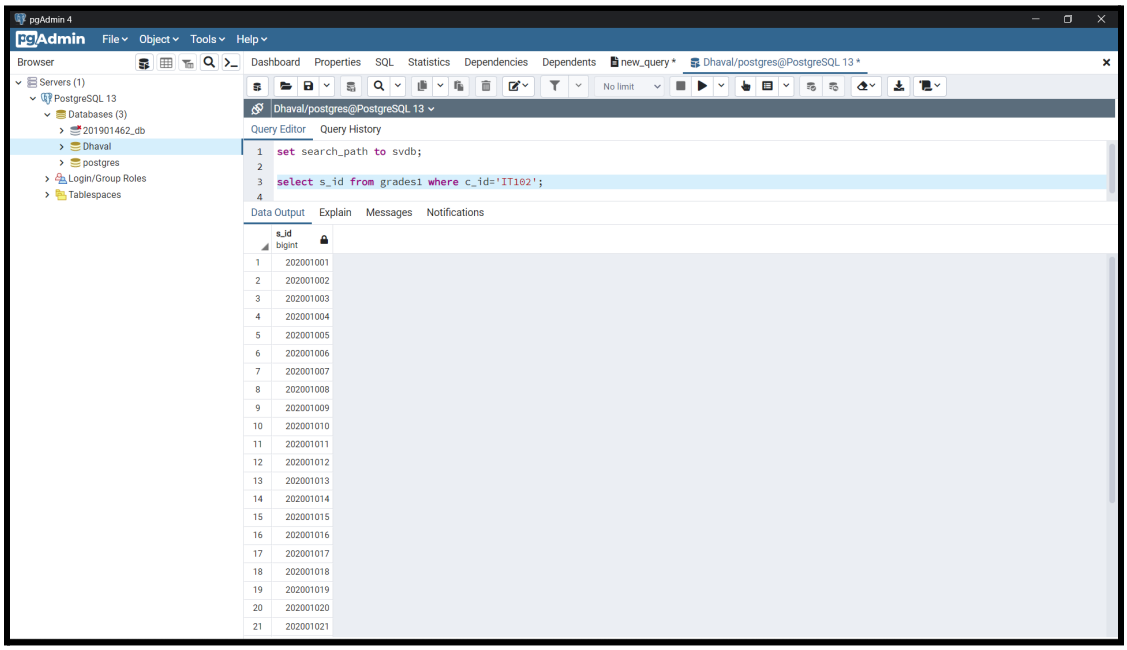
2. **View Student IDs having a grade of AA in IT102.**
   SQL - select s_id from grades1 where grade='AA' and c_id='IT102'



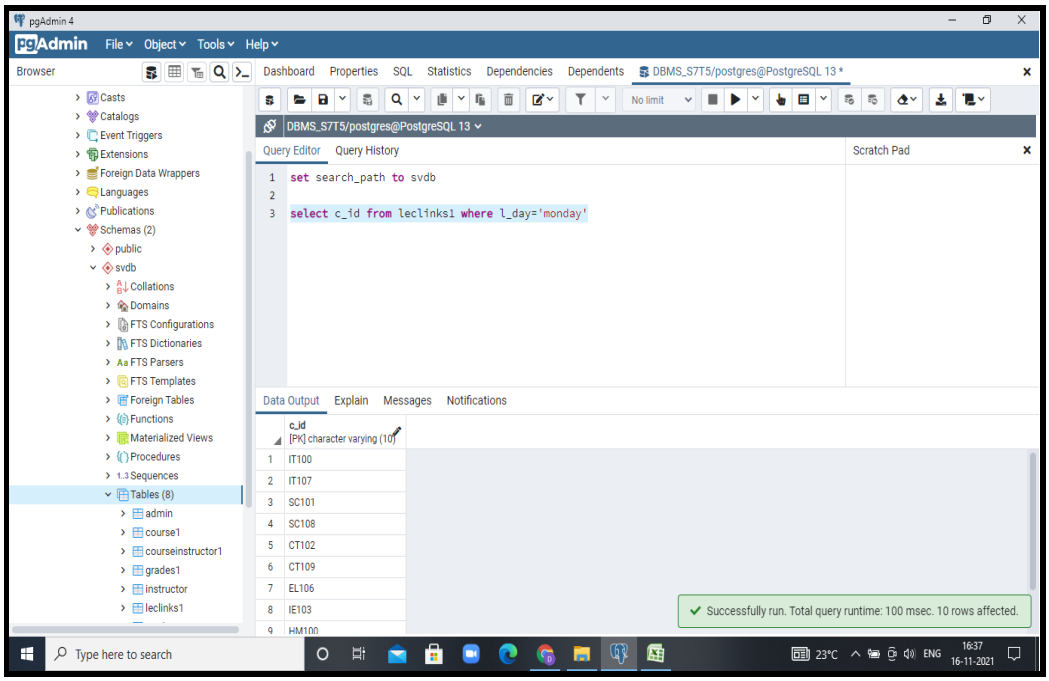
3. **View courses worth three credits.**
   SQL - select c_id from course1 where credits=3



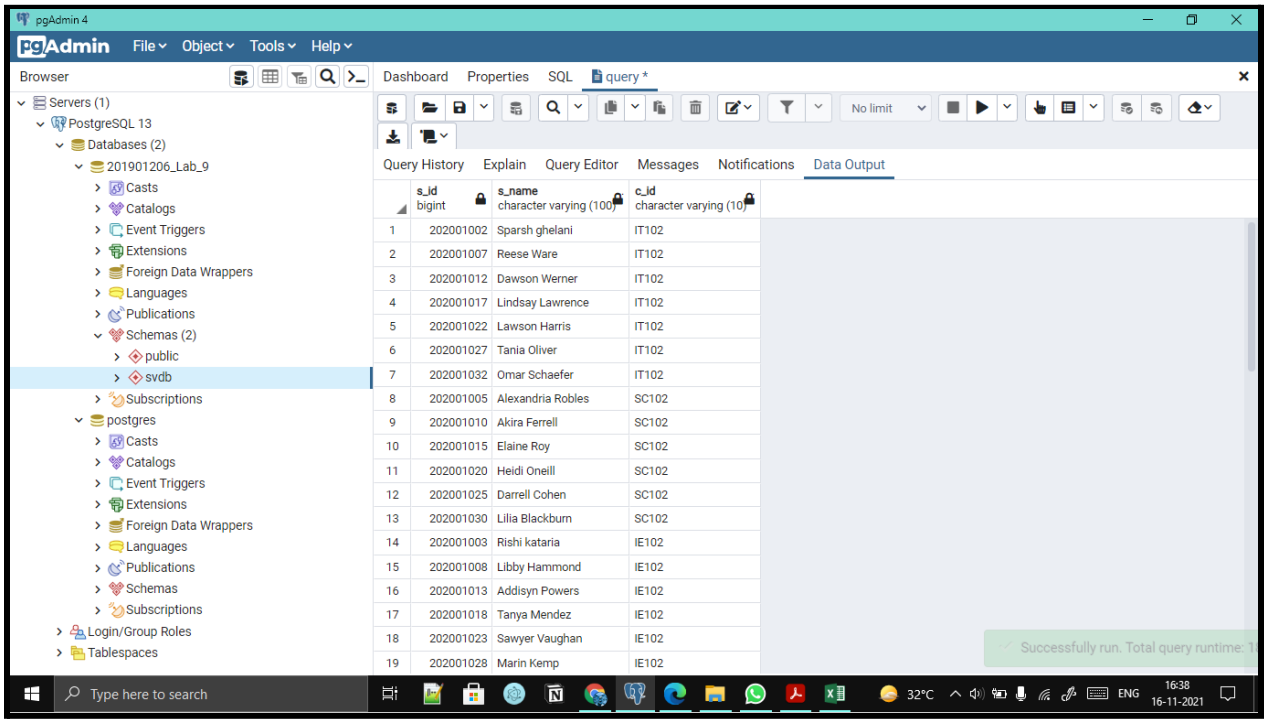
4. **View all students studying IT102.**
   SQL - select s_id from grades1 where c_id=’IT102’;



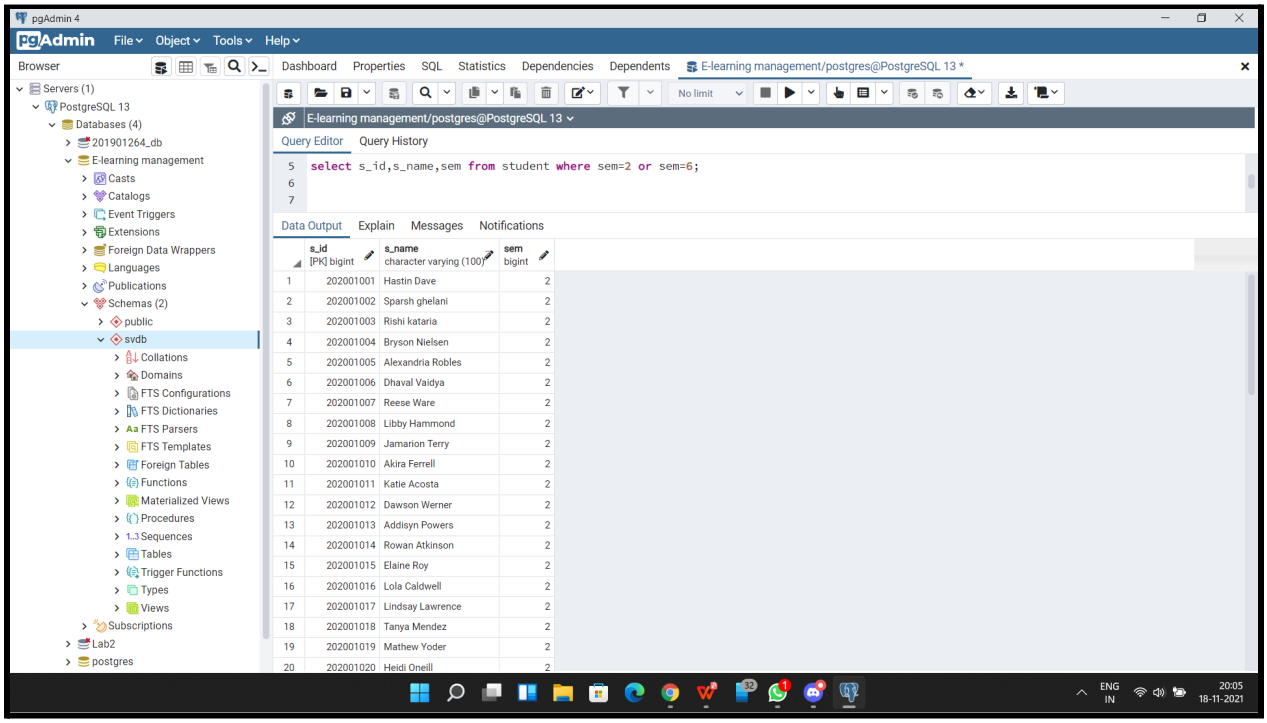
5. **View all lectures on Monday.**
SQL - select c_id from leclinks1 where l_day='monday'



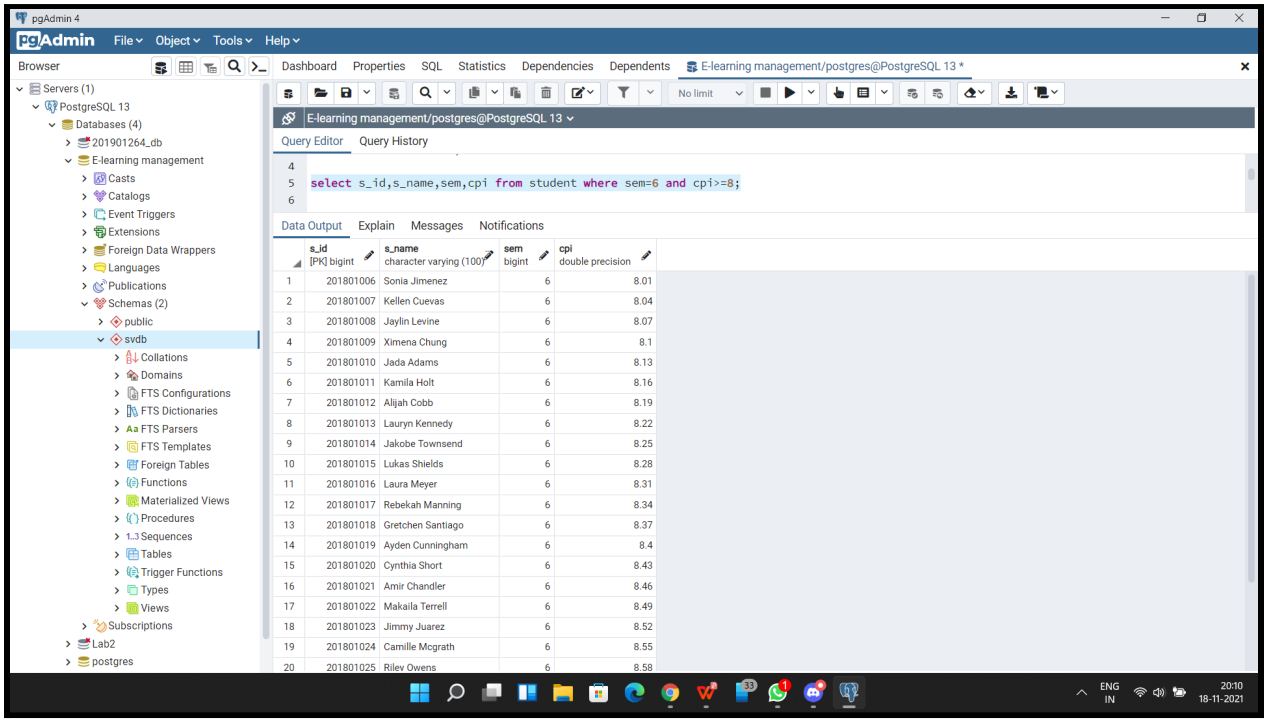
6. **View student ID and names whose grade is AB. Also, mention the corresponding course id in which the student got this grade.**
SQL - select s_id,s_name,c_id from student natural join grades1 where student.s_id=grades1.s_id and grade='AB'



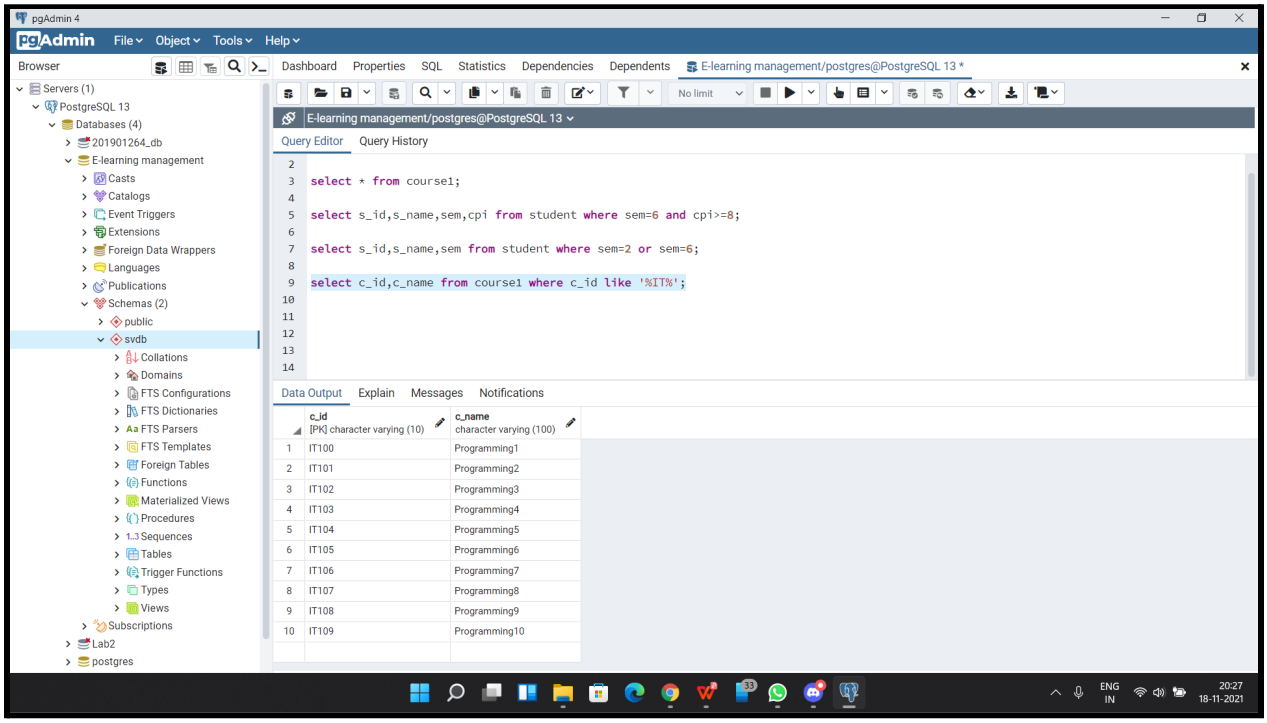
7. **View students who are either in semester two or semester six.**
   SQL - select s_id,s_name,sem from student where sem=2 or sem=6;



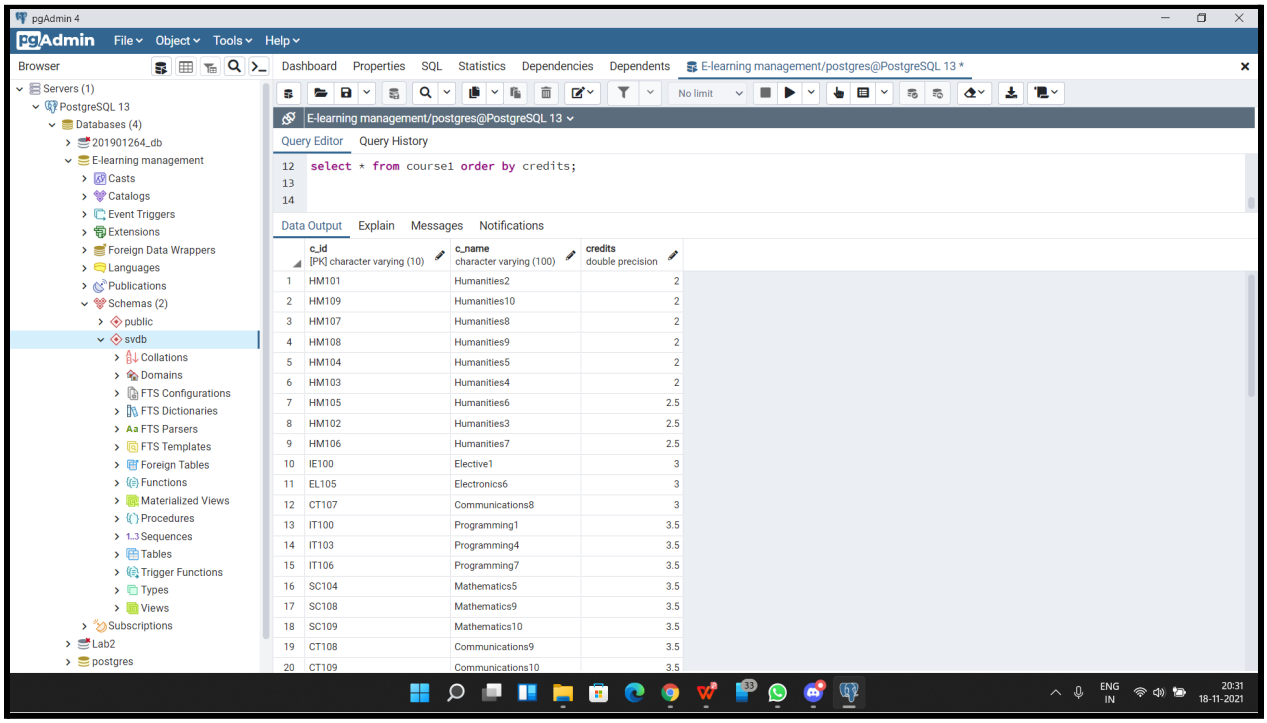
8. **Find students studying in semester six and having a CPI of 8+.**
   SQL - select s_id,s_name,sem,cpi from student where sem=6 and cpi>=8;



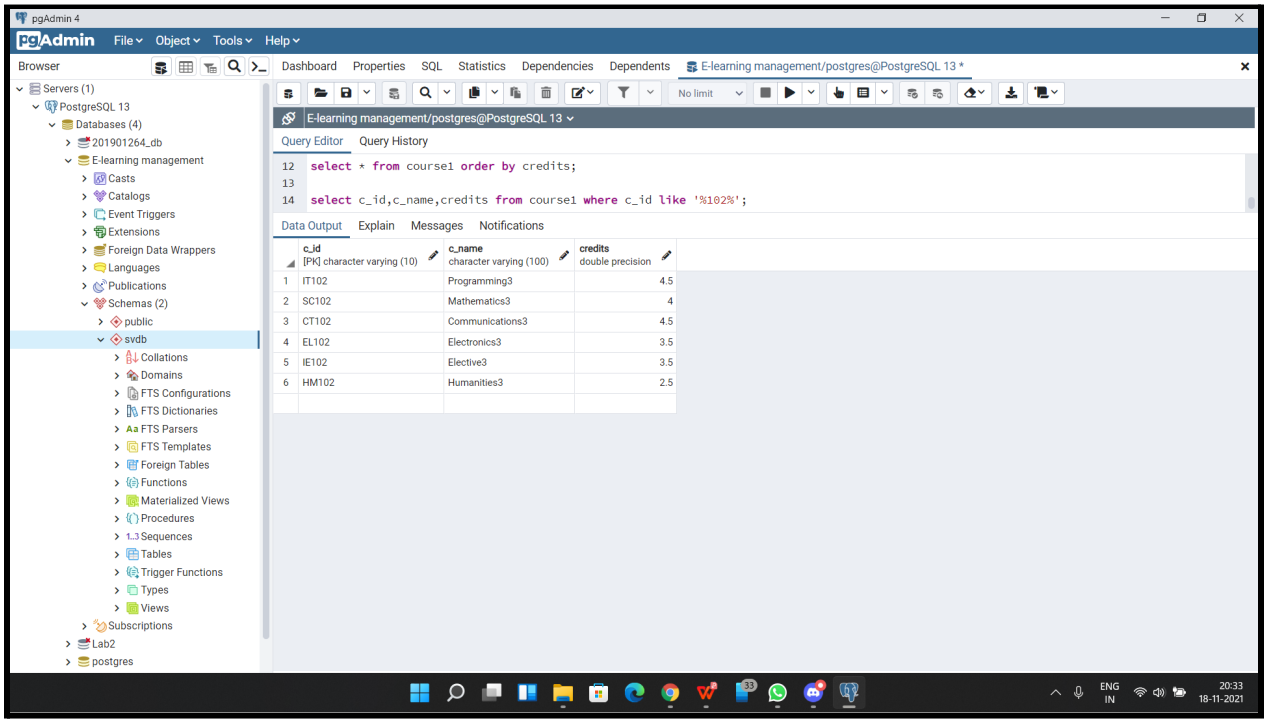
9. **Print all the Programming courses.**
   SQL - select c_id,c_name from course1 where c_id like '%IT%';



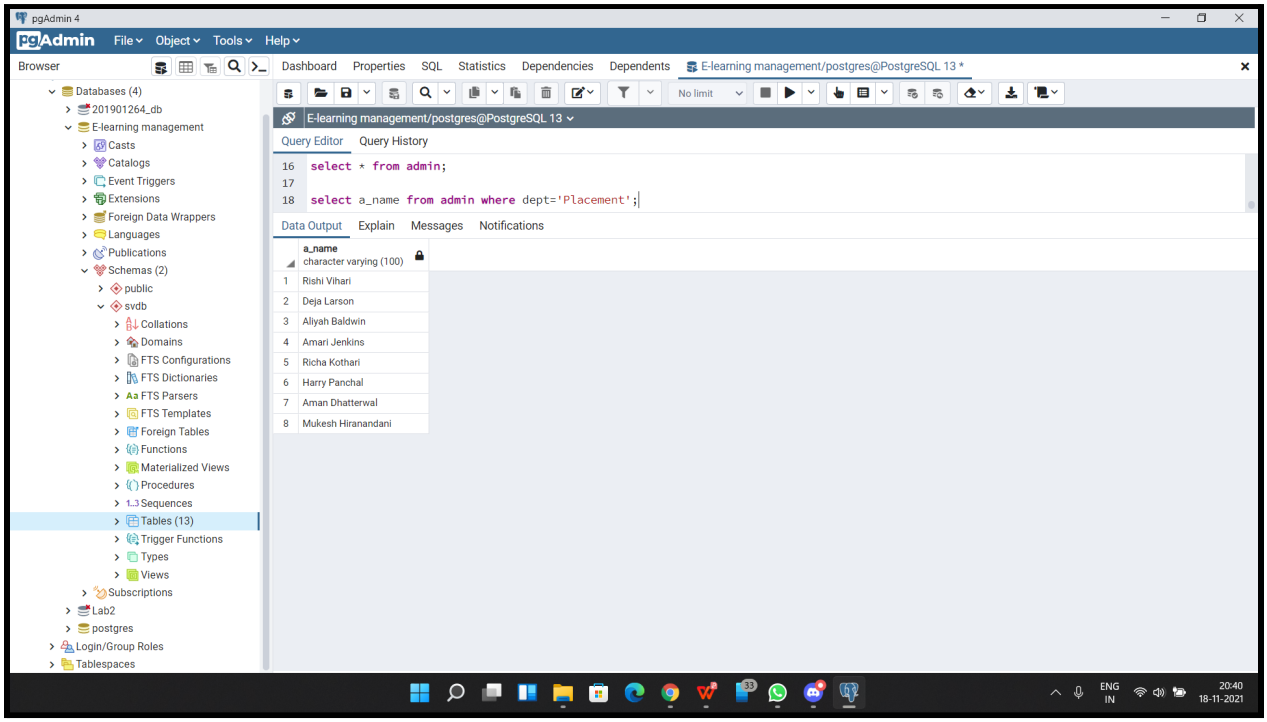
10. **Sort all the courses in ascending order of their credits.**
    SQL - select * from course1 order by credits;



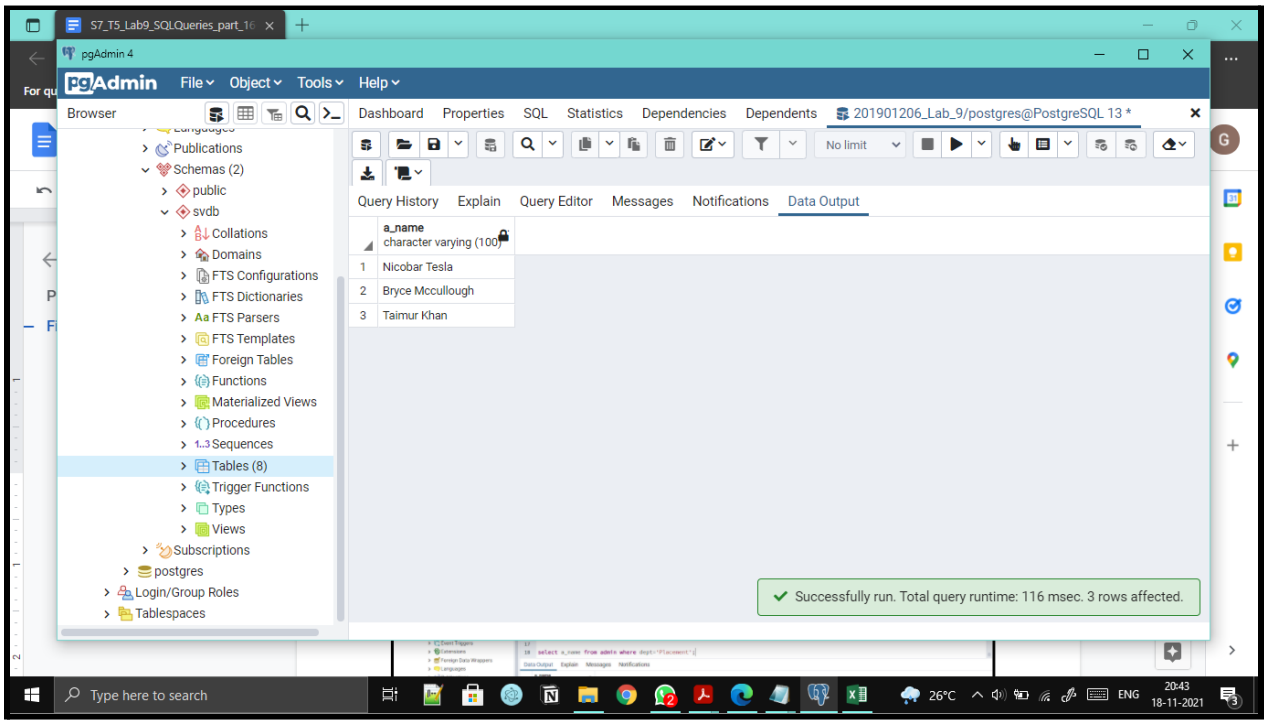
11. **Find all the courses taught in semester 2.**
    SQL - select c_id,c_name,credits from course1 where c_id like '%102%';



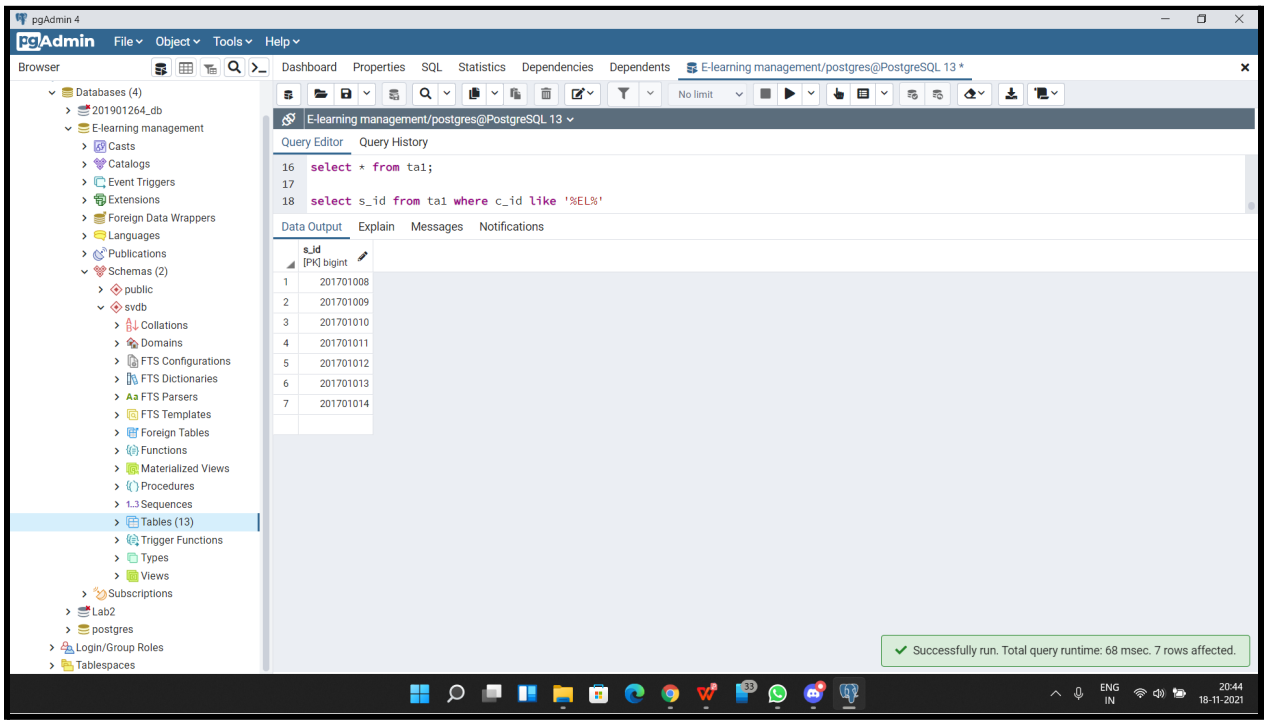
12. **Mention the names of people who work in the Placement department**
    SQL - select a_name from admin where dept='Placement';



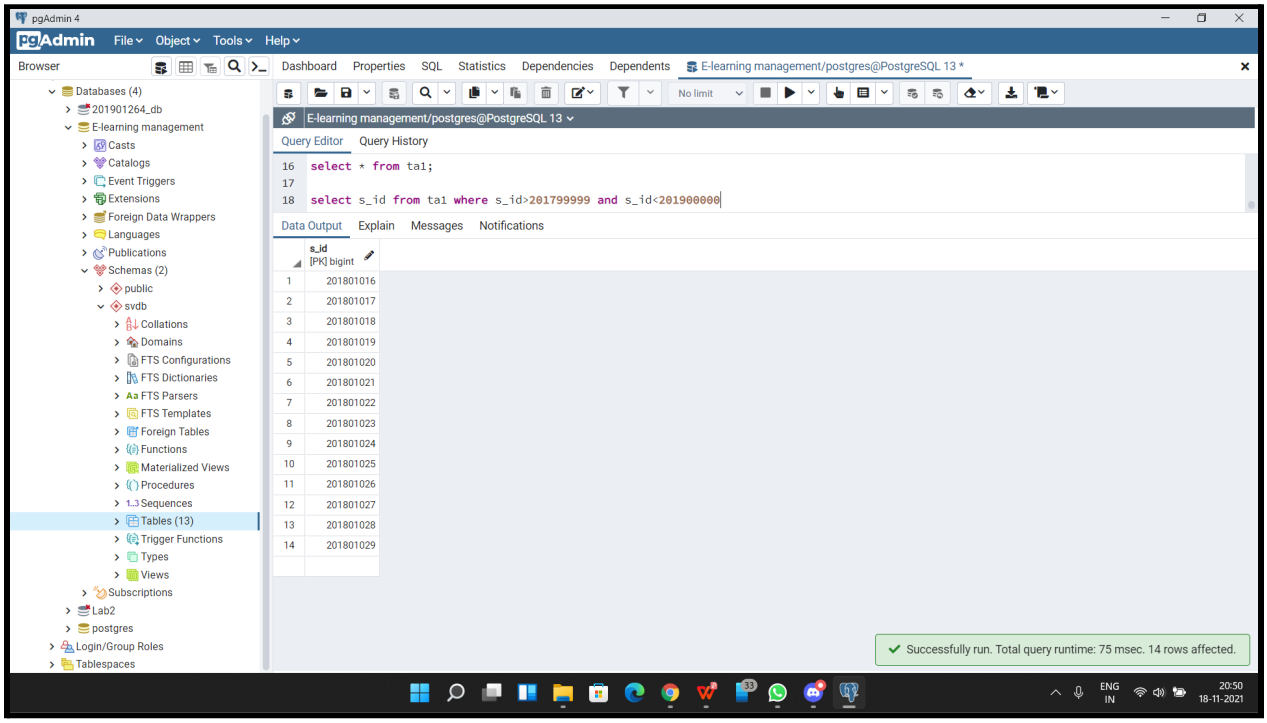
13. **Mention the names of people who work in the library department**
    SQL - select a_name from admin where dept='Library';



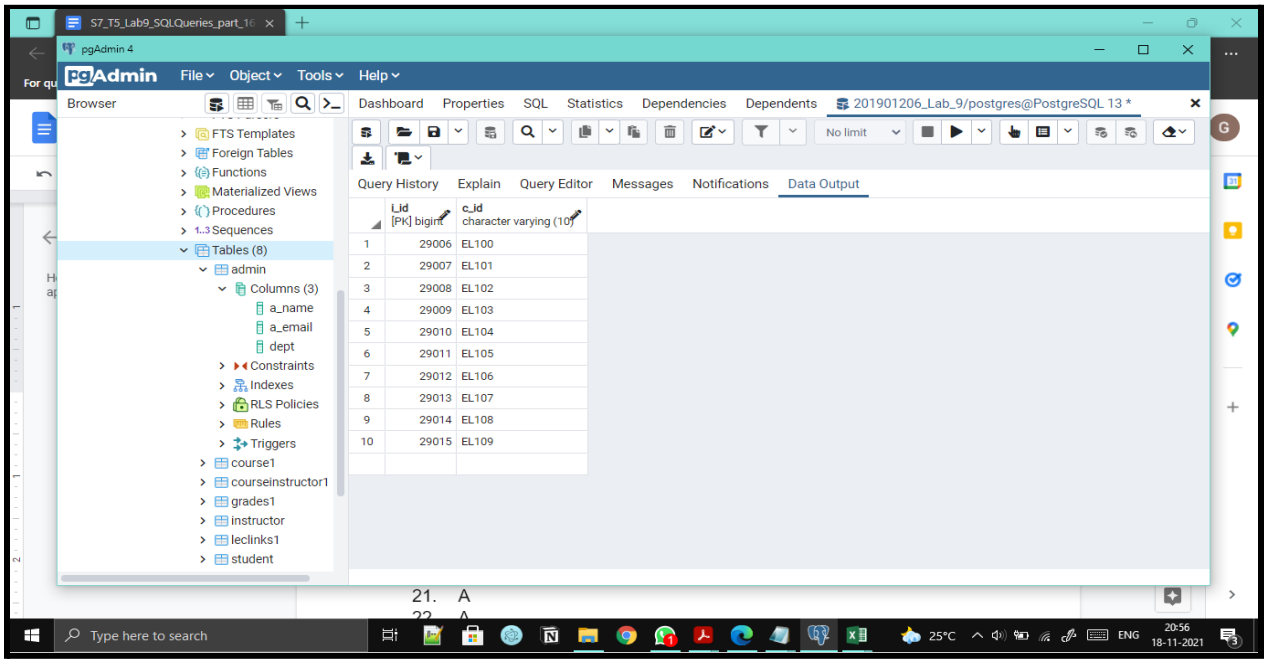
14. **Find s_ids of TAs who assist in an Electronics course.**
    SQL - select s_id from ta1 where c_id like '%EL%'



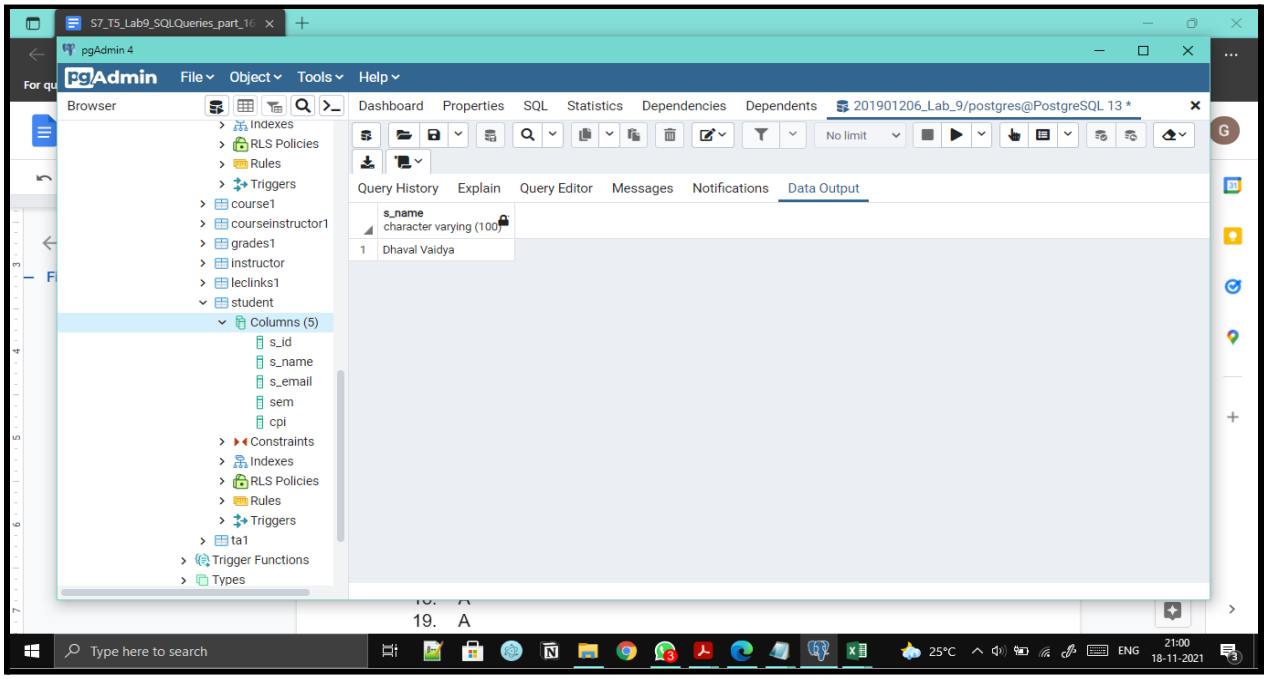
15. **Find all the TAs who study in semester 6.**
    SQL - select s_id from ta1 where s_id>201799999 and s_id<201900000



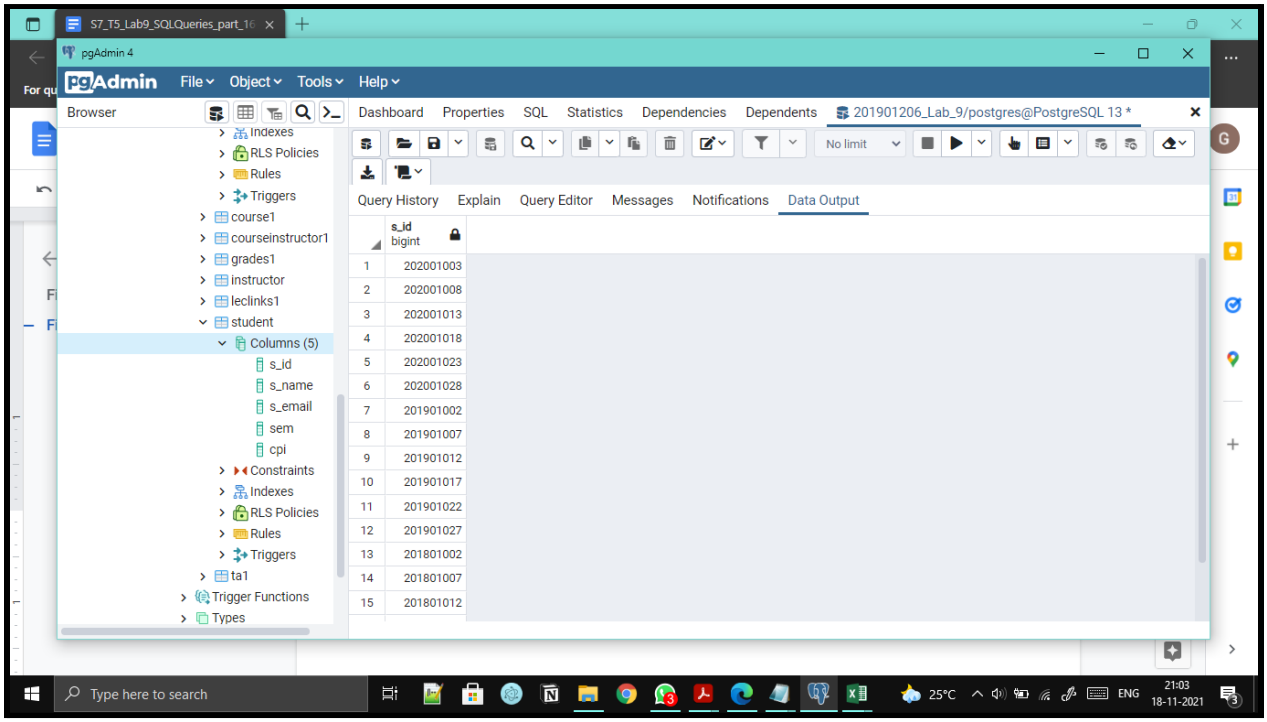
16. **Find instructor id of instructors that teaches Electronic courses**
    SQL - select * from courseinstructor1 where c_id like'%EL%';



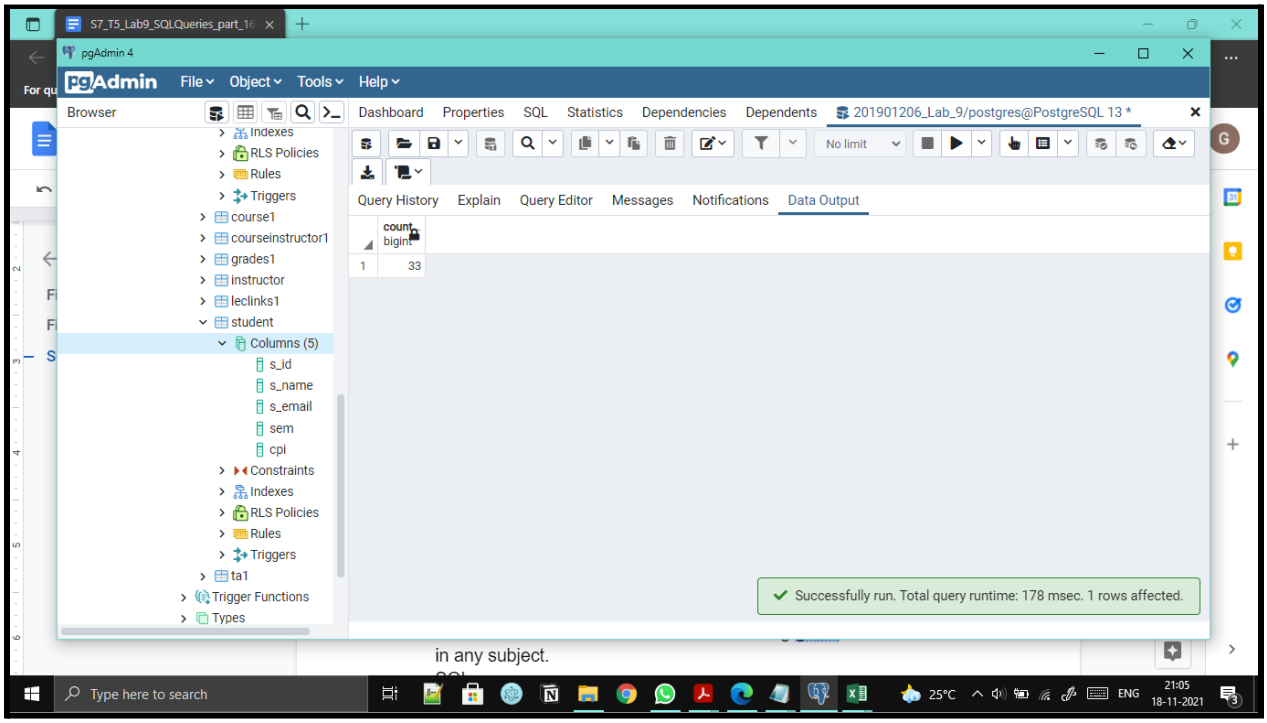
17. **Find the name/s of the student/s having the lowest cpi.**
SQL- select s_name from student where cpi=(select min(cpi) from student);



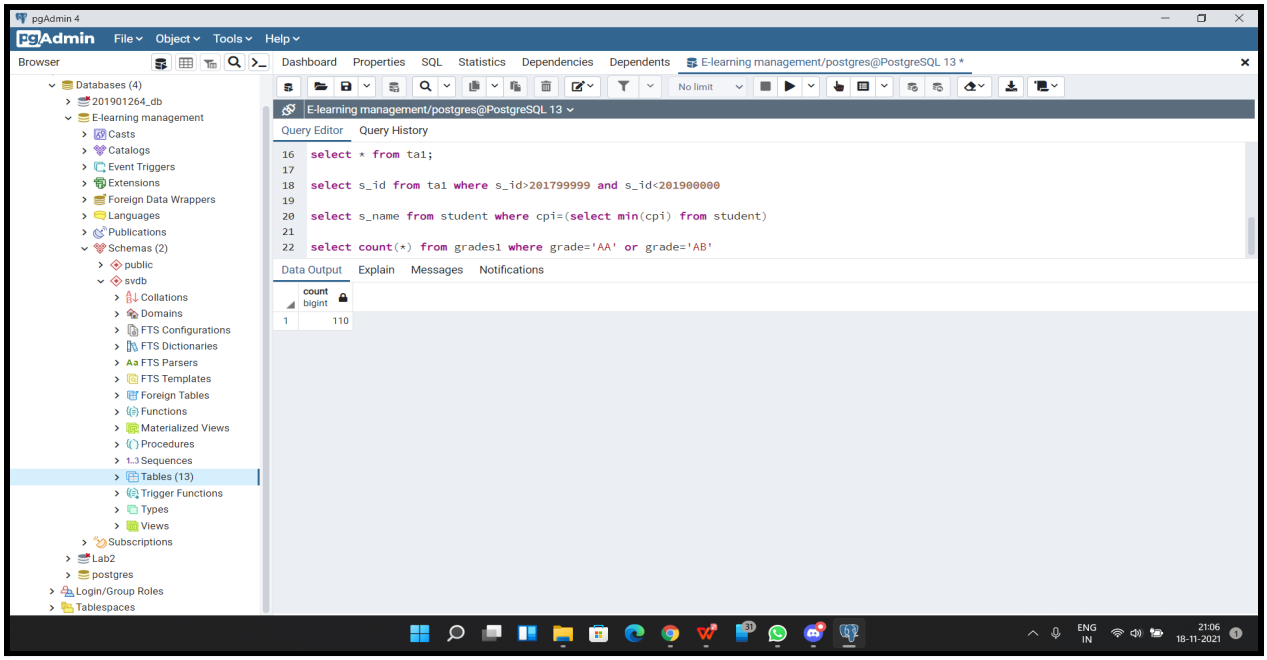
18. **Find the student ID of students who got BB grades in the IT course.**
SQL - select s_id from grades1 where grade='BB' and c_id like '%IT%';



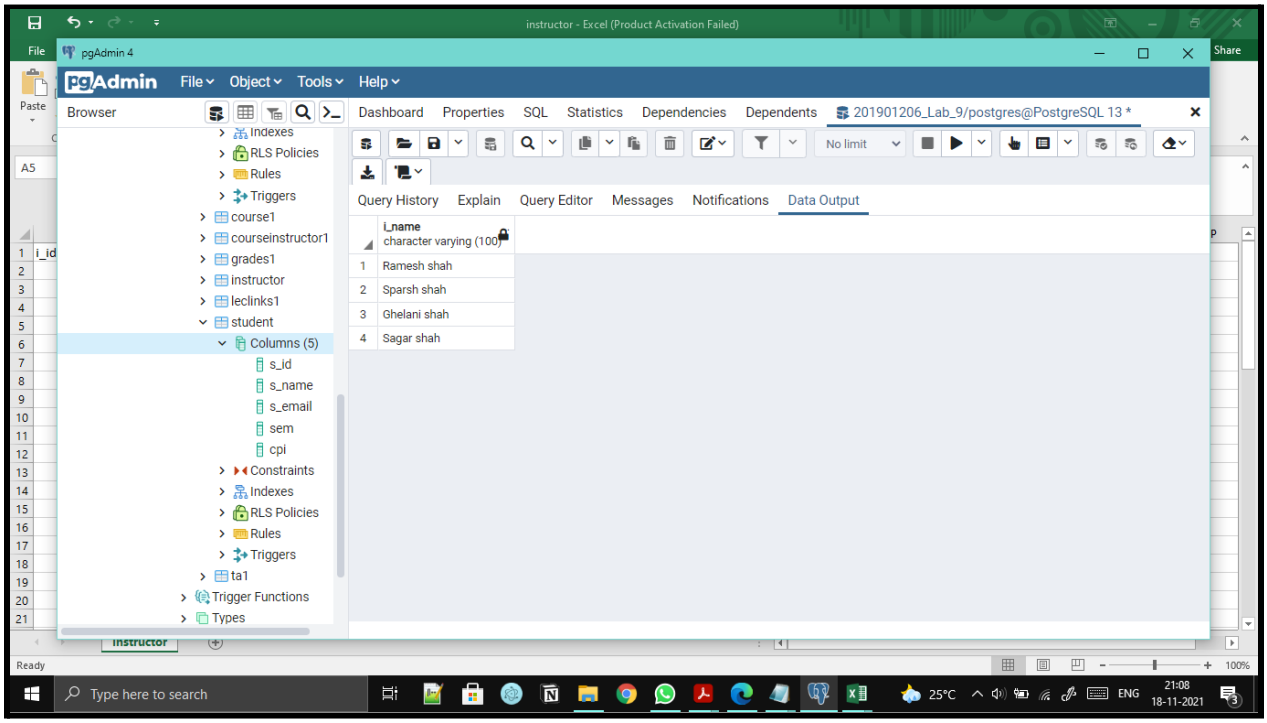
19. **Find the total number of students having CPI between 7 and 8.**
    SQL - select COUNT(*) from student where cpi<=8 and cpi>=7;



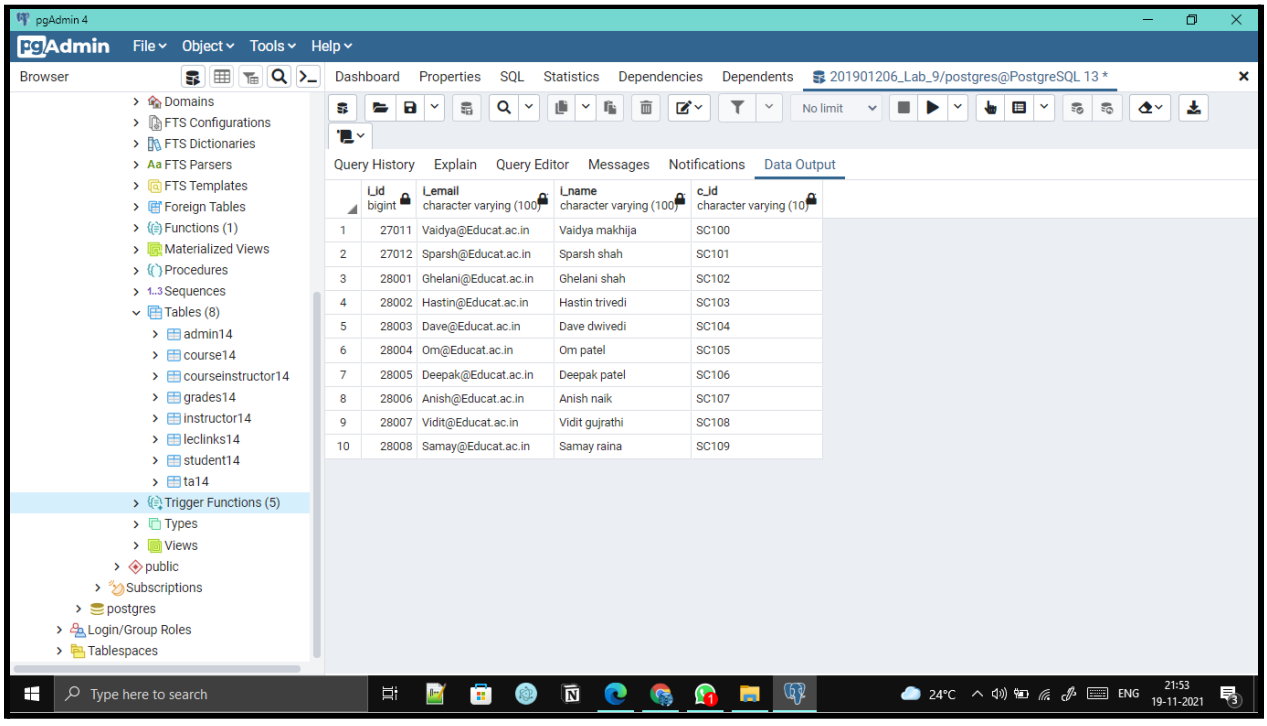
20. **Count the total number of students having grades either 'AA' or 'AB' in any subject.**
    SQL - select count(*) from grades1 where grade='AA' or grade='AB'



21. **Find the names of instructors who have surnames as ‘shah.’**
SQL - select i_name from instructor where i_name like '%shah%';

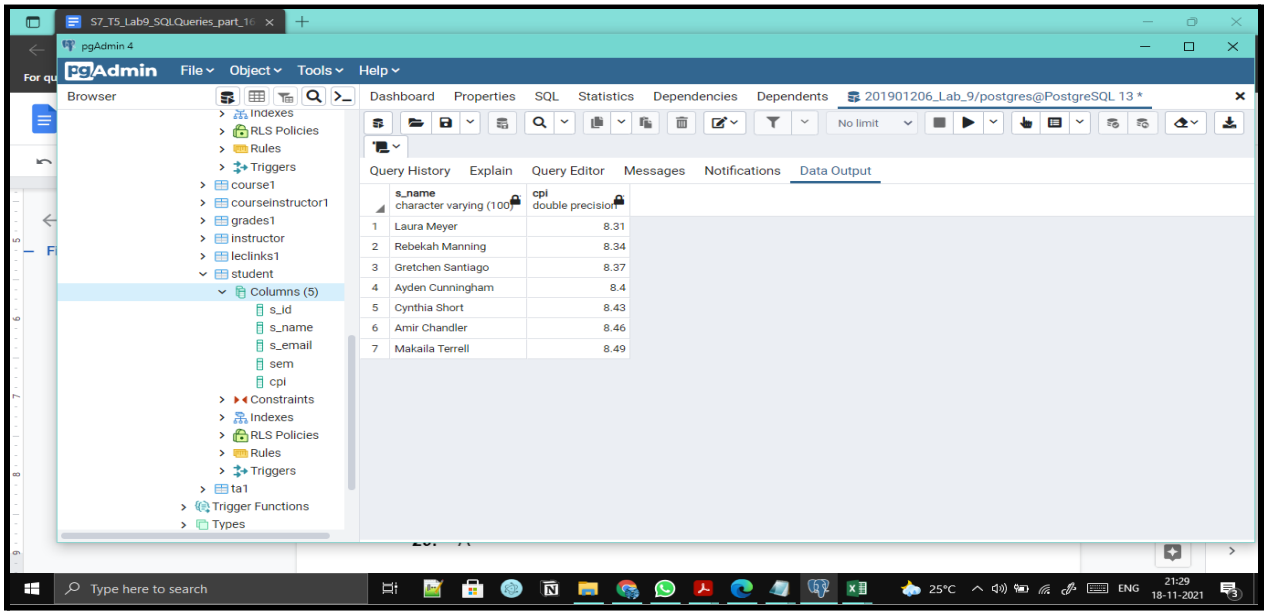


22. **Find names of instructors that teach in SC (mathematics) courses.**
SQL - select * from instructor natural join courseinstructor1 where instructor.i_id=courseinstructor1.i_id and c_id like '%SC%';



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

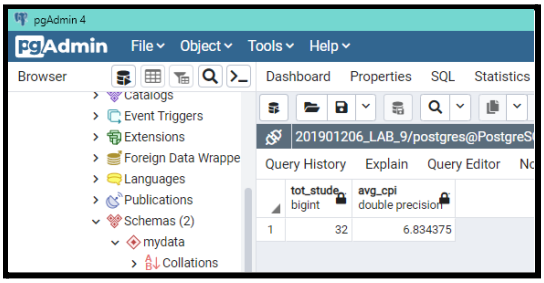
SQL- select s_name,cpi from student natural join ta1 where student.s_id=ta1.s_id and c_id like '%IT%';



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;

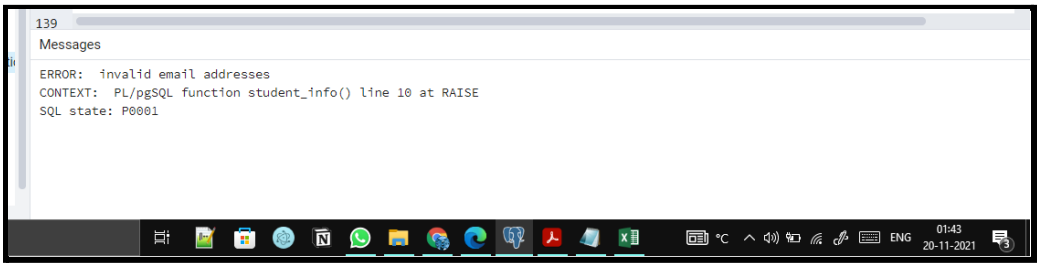
select * from semwise_cpi(2);



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.**

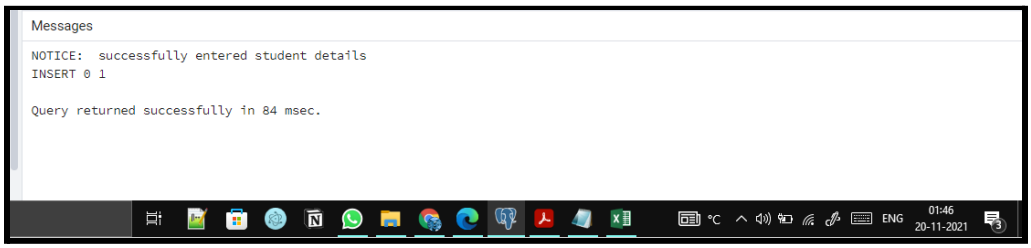
Insert statement :

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

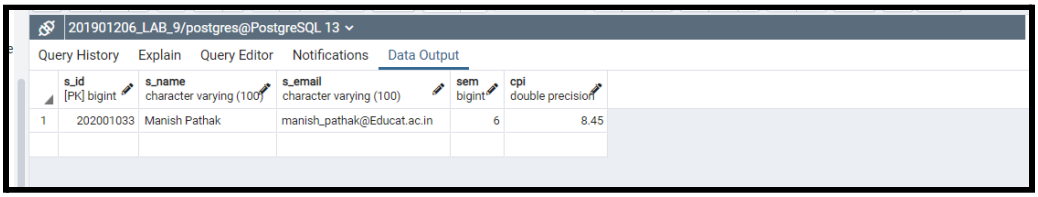


Corrected insert statement :

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

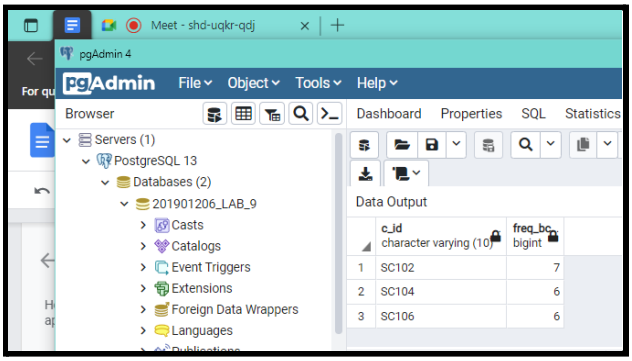


SQL - select * from Student14 where s_id = 202001033



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

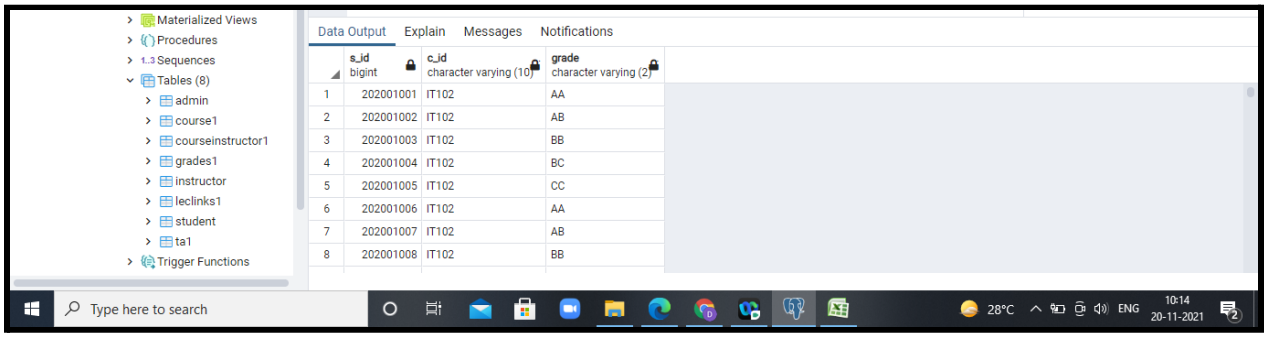
SQL - create view SC_freq as
select c_id,count(grade)
as freq_BC from Grades14 where c_id like '%SC%'
and grade = 'BC' group by c_id;
Select * From SC_freq;



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

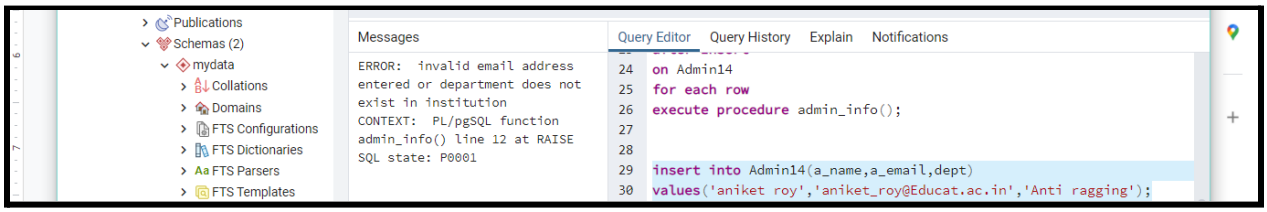
27. Print IT grades of all students.

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%'



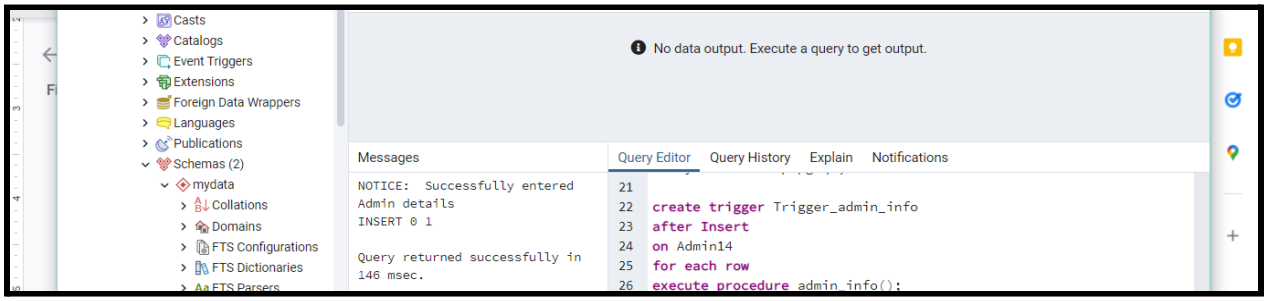
28. Find the output for insertion of the following values in admin.
Insert statement.

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



Corrected insert statement :

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

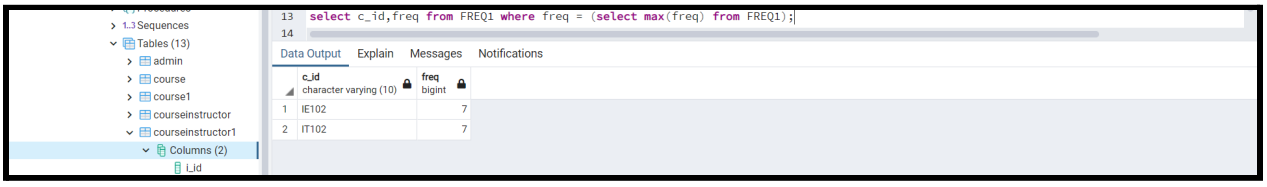


SQL - select * from Admin14 where a_email='aniket_pathak@Educat.ac.in';

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;

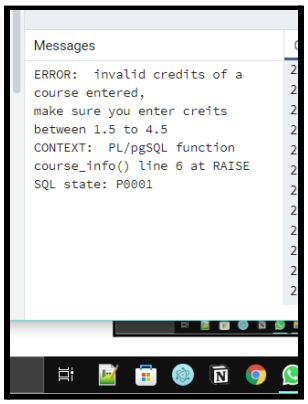
select c_id,freq from FREQ1 where freq = (select max(freq) from FREQ1);



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 -

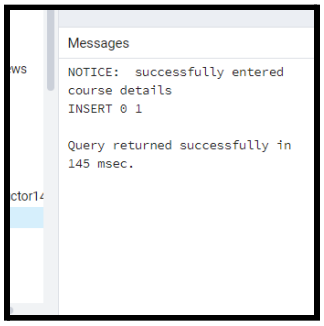
Insert statement :

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

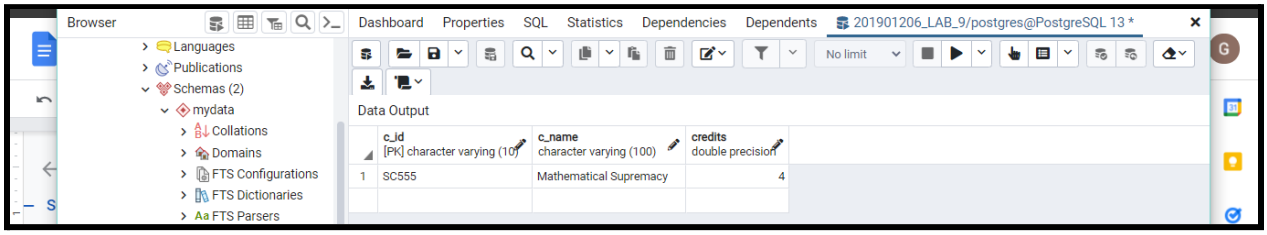


Corrected insert statement :

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



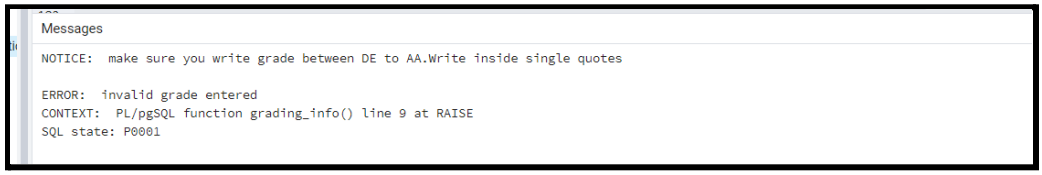
SQL - select * from Course14 where c_id='SC555';



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.**

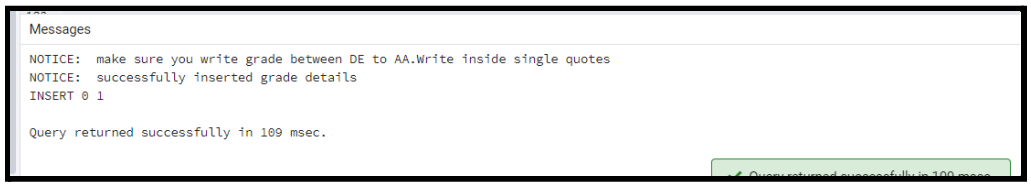
Insert statement :

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

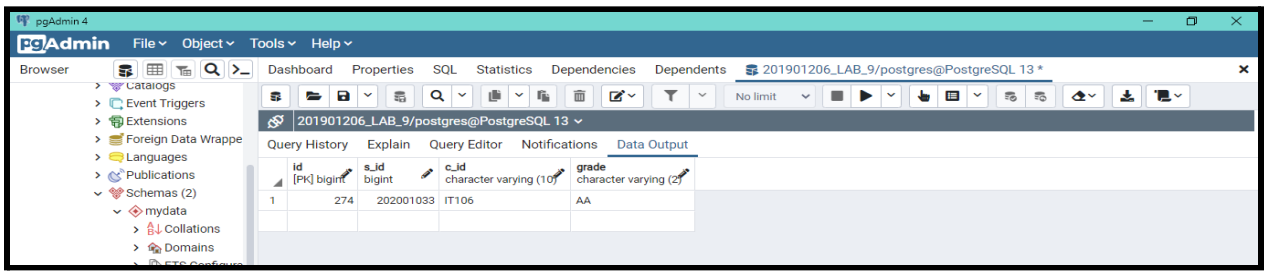


Corrected insert statement :

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



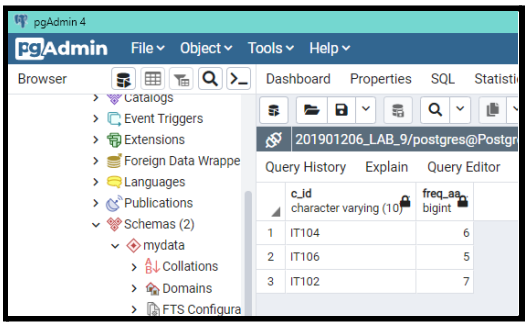
SQL - select * from Grades14 where s_id = 202001033



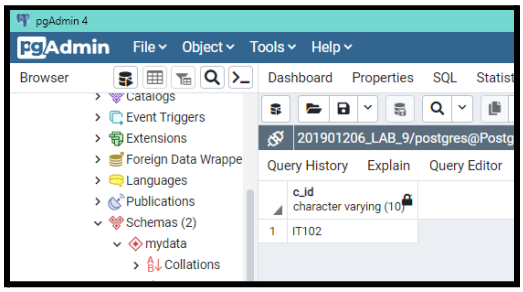
32. **Find the course ID in IT in which the student got maximum AA grade.**

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;

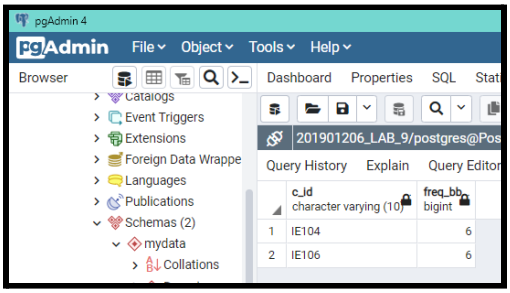
Select * From FREQ;



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

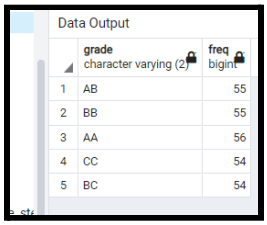


33. **Find out frequency of BB grade in IE104 and IE106 courses.**
SQL - create view FREQ as select c_id,count(grade)
as freq_BB from Grades14 where (c_id='IE104' and grade='BB') or (c_id='IE106' and grade = 'BB') group by c_id

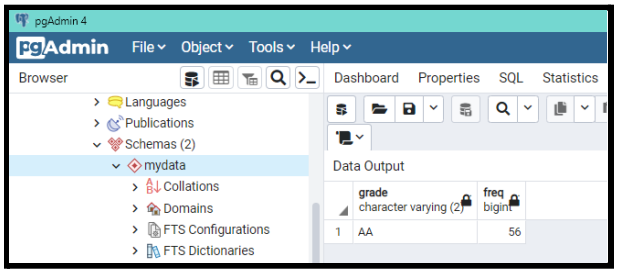


34. **Find the grade that appears most times in the grade table.**
SQL - create view Grade_Freq as
select grade,count(grade) as freq from Grades14 group by grade;
Select * from Grade_Freq;

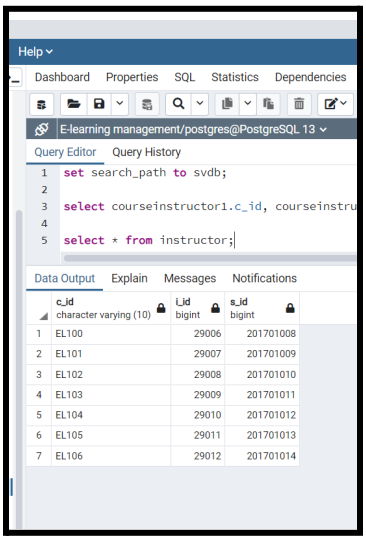
select grade,freq from Grade_Freq



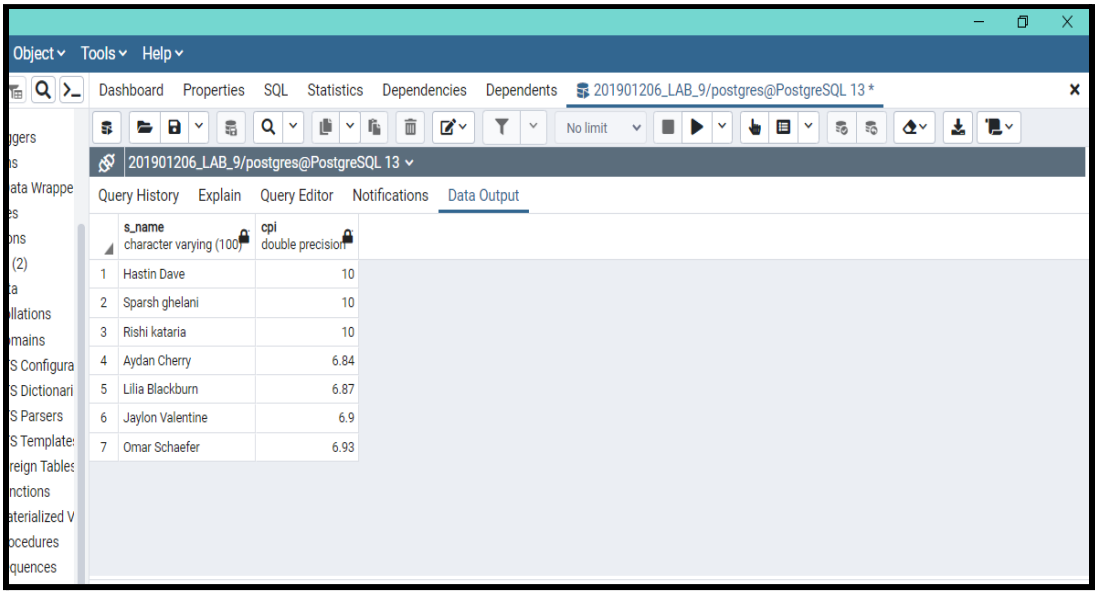
where freq=(select max(freq) from Grade_Freq);



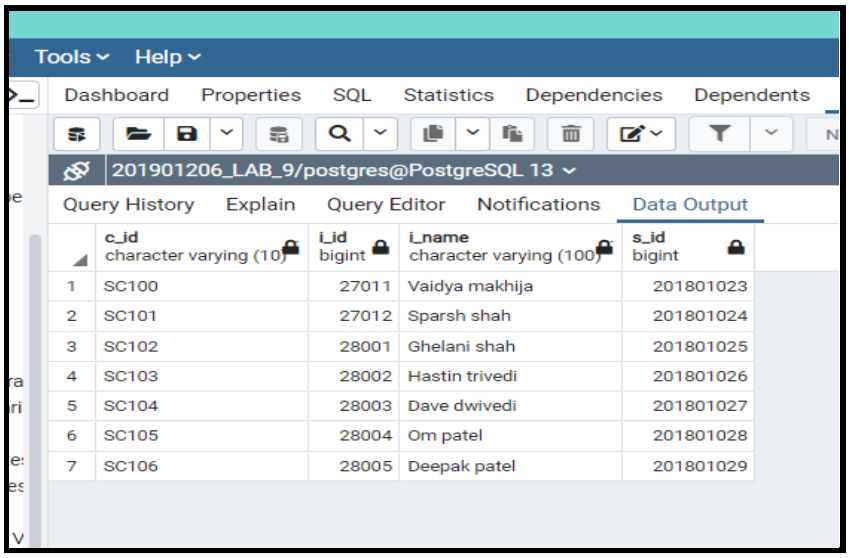
35. **Find the instructors who teach EL courses and their Teaching Assistants.**
SQL - select courseinstructor1.c_id, courseinstructor1.i_id, ta1.s_id from courseinstructor1 natural join ta1 where courseinstructor1.c_id=ta1.c_id and c_id like '%EL%';



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))

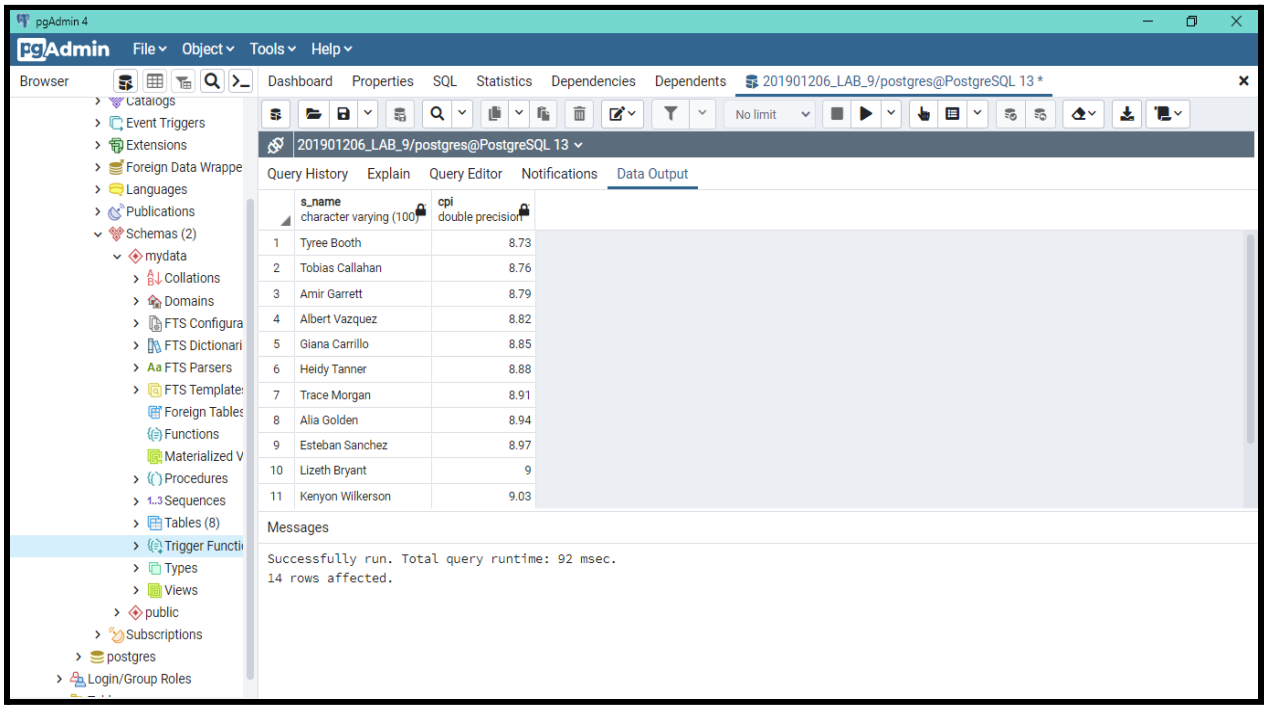


37. **Find the instructors who teach the SC course and the ID of TA's who are assisting the course.**
SQL - select c_id,i_id,i_name,s_id from Instructor14
natural join Courseinstructor14 natural join Ta14
where c_id like '%SC%' and Instructor14.i_id = Courseinstructor14.i_id;



	c_id character varying (10)	i_id bigint	i_name character varying (100)	s_id bigint
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));

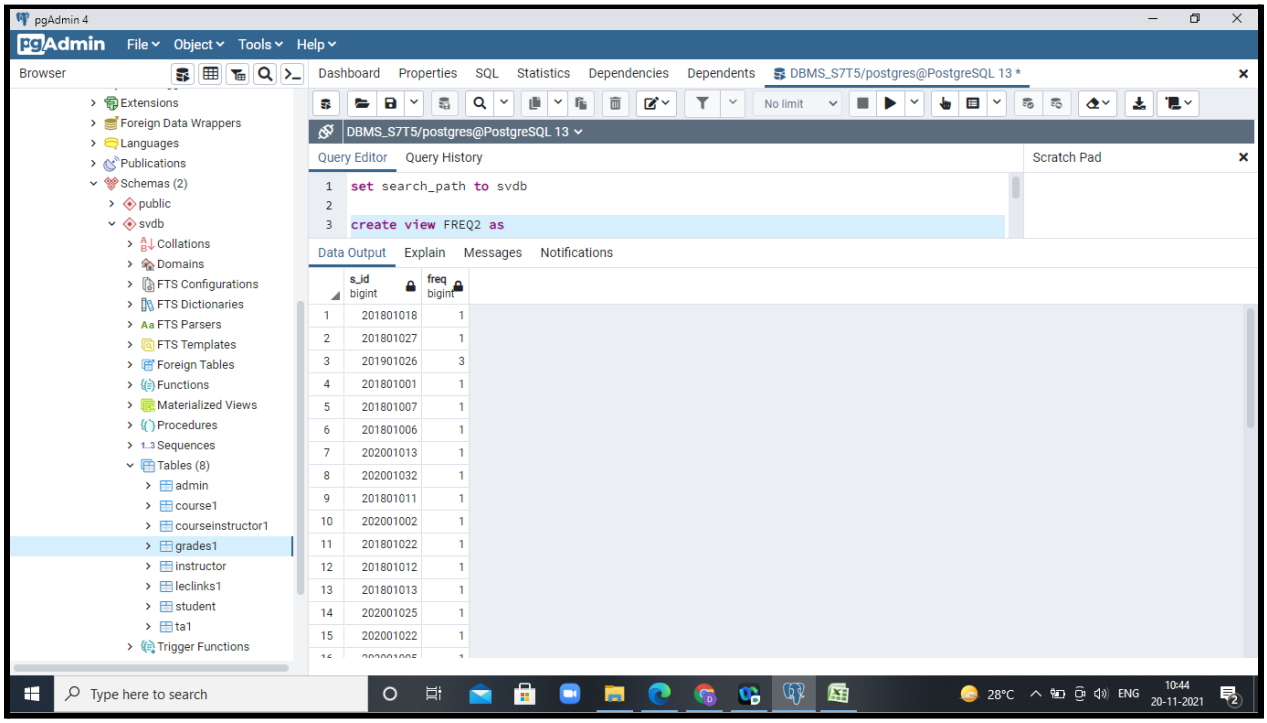


	s_name character varying (100)	cpi double precision
1	Tyree Booth	8.73
2	Tobias Callahan	8.76
3	Amir Garrett	8.79
4	Albert Vazquez	8.82
5	Giana Carrillo	8.85
6	Heidy Tanner	8.88
7	Trace Morgan	8.91
8	Alia Golden	8.94
9	Esteban Sanchez	8.97
10	Lizeth Bryant	9
11	Kenyon Wilkerson	9.03

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

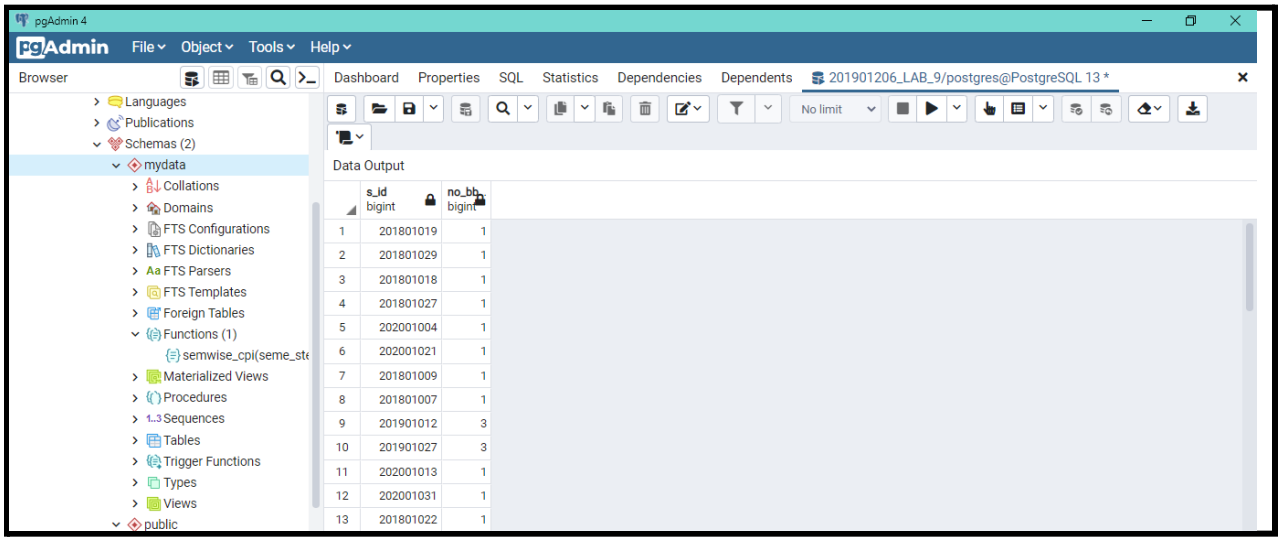
SQL - create view FREQ2 as
select s_id,count(grade)
as freq from grades1 where grade = 'AB' group by s_id;

select * From FREQ2;



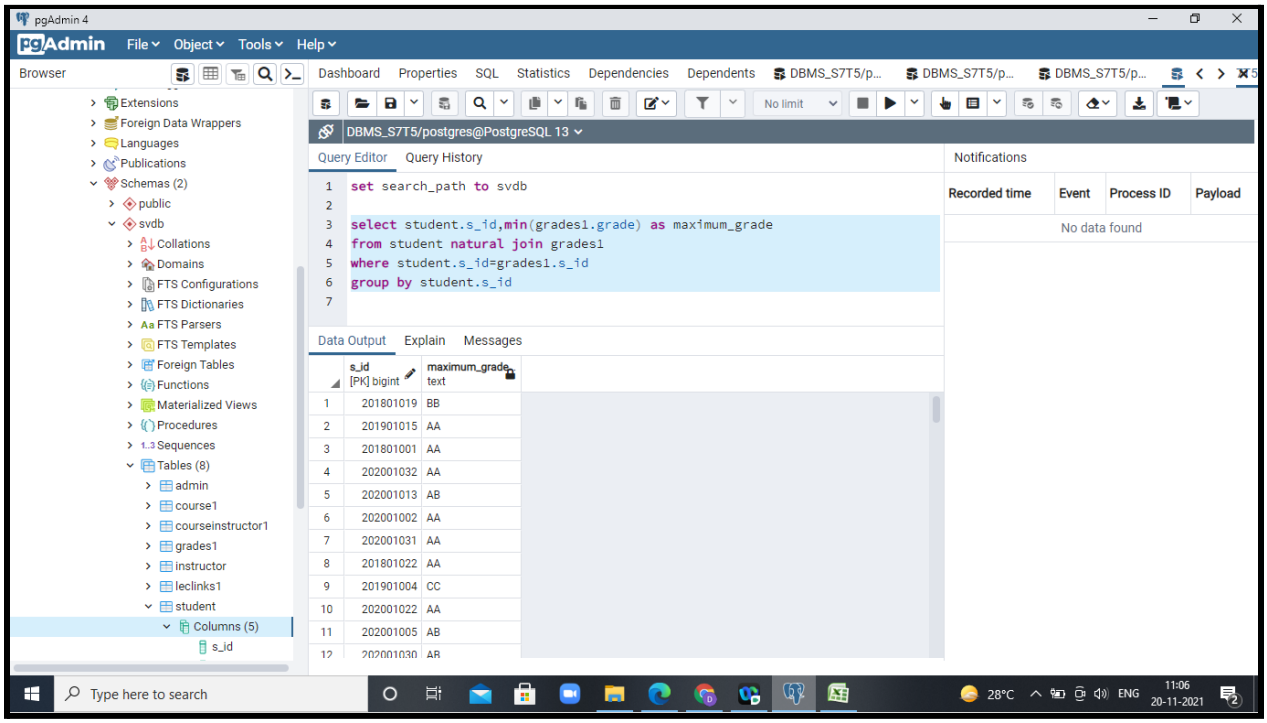
40. Display the number of 'BB' grades received by every student in the batch.

SQL - select s_id,count(grade) as no_BB from Grades14 where grade='BB' group by s_id;



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



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.FileInputStream;
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 dekete 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();
                        }

                }

}
```

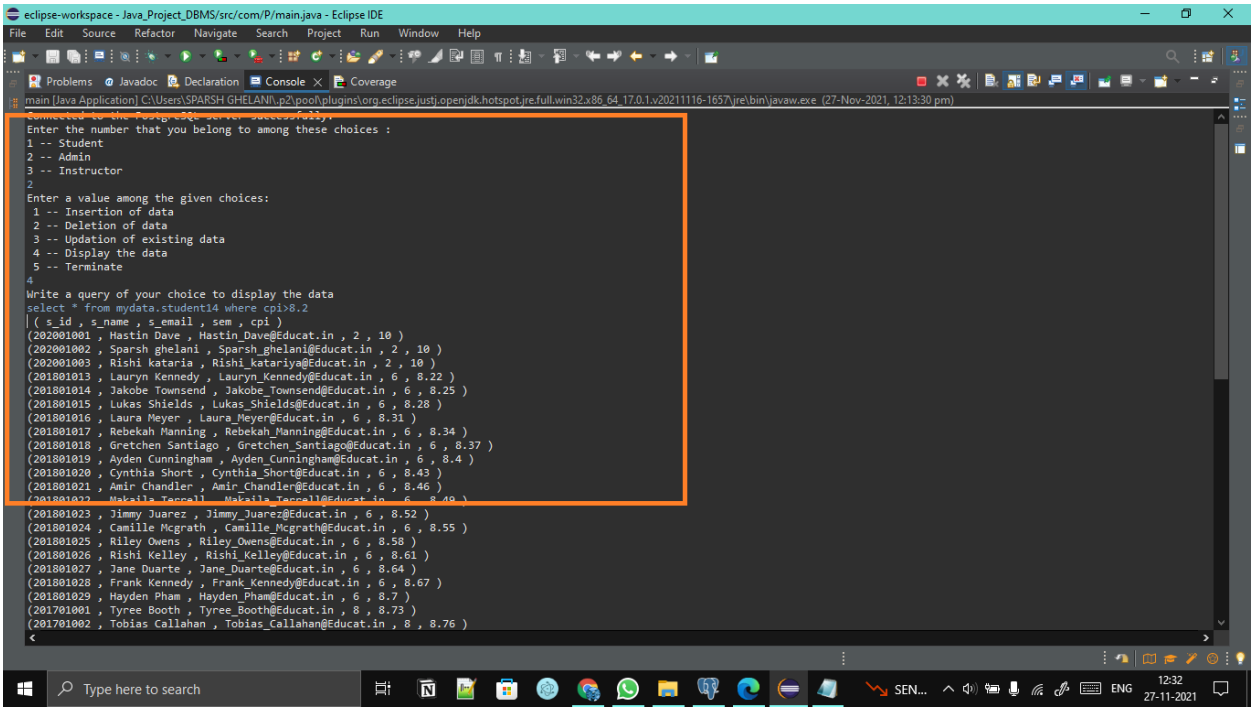
```
/* Display of results obtained according to query written */
        public void display()
{
                try {
                        String url = "jdbc:postgresql://localhost/201901206_LAB_9";
                        String user = "postgres";
                        String password = "admin";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

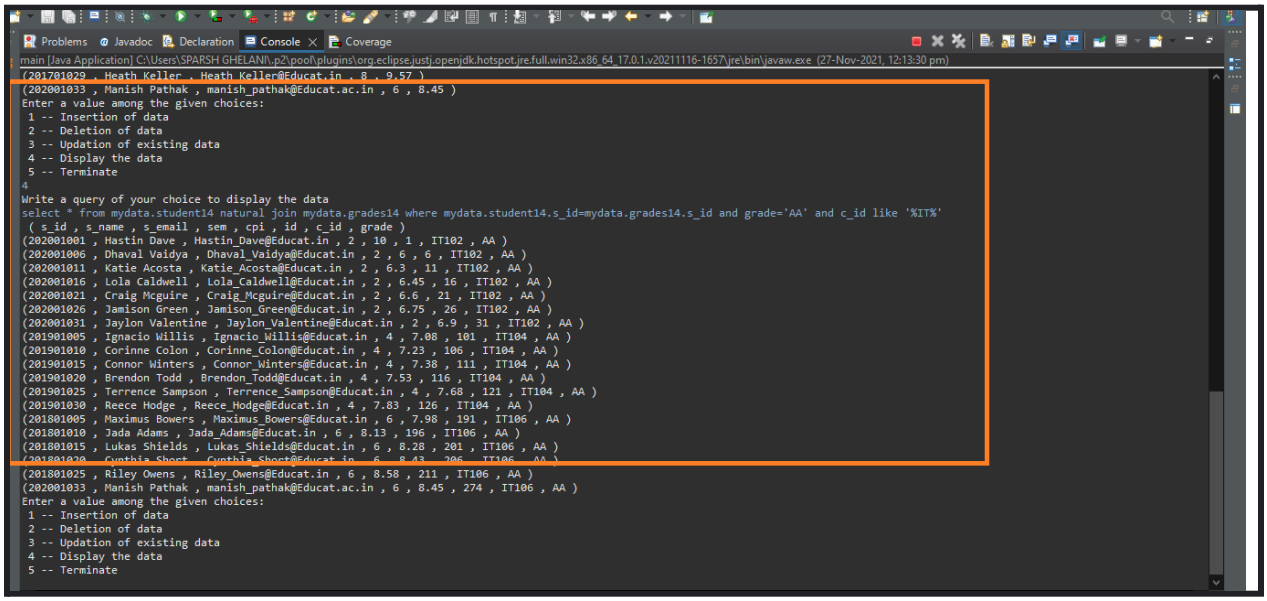
```
                    System.out.println(" 6 -- instructor ");
                    System.out.println(" 7 -- admin");
                    System.out.println(" 8 -- ta ");

                    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

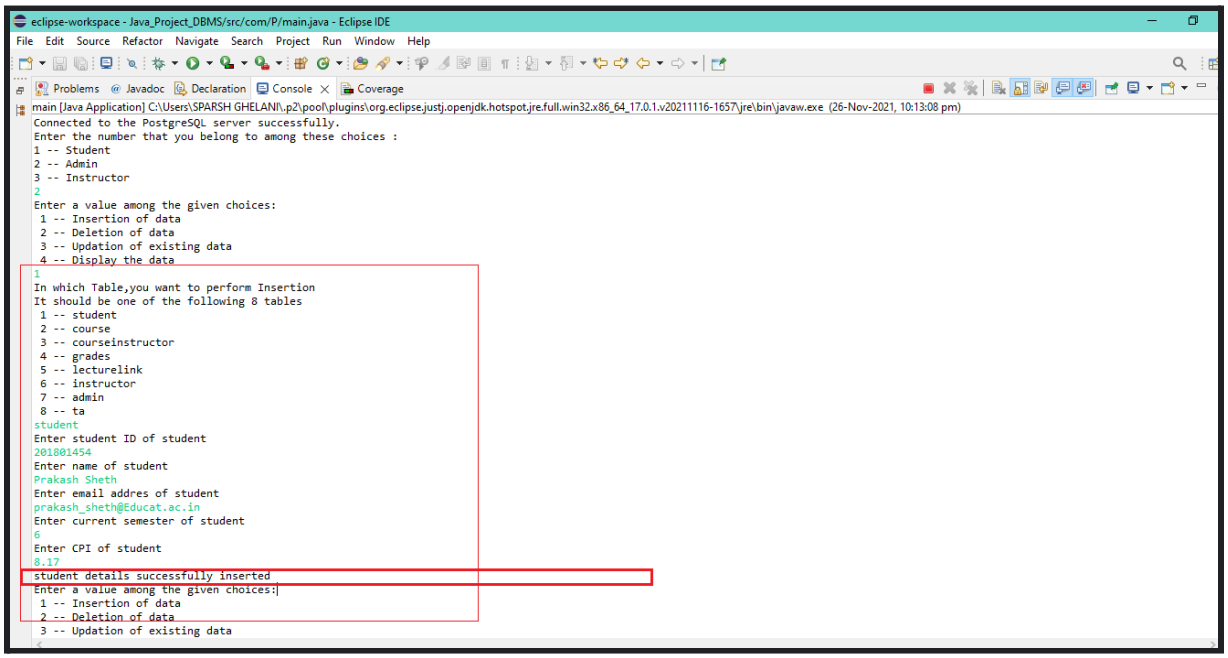
1. **Print details of students scoring cpi>8.2.**

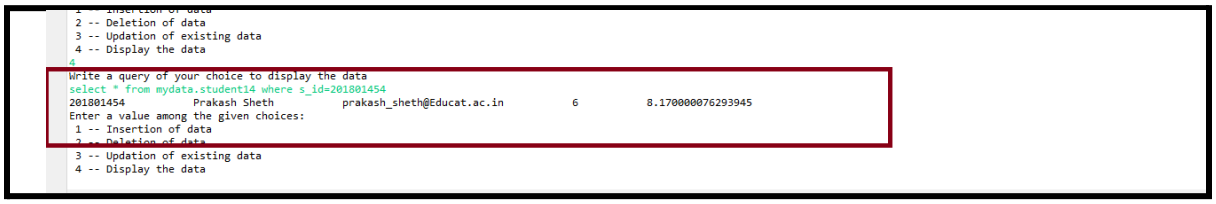


2. **Find Student ID of students who got AA grade in any IT course.**

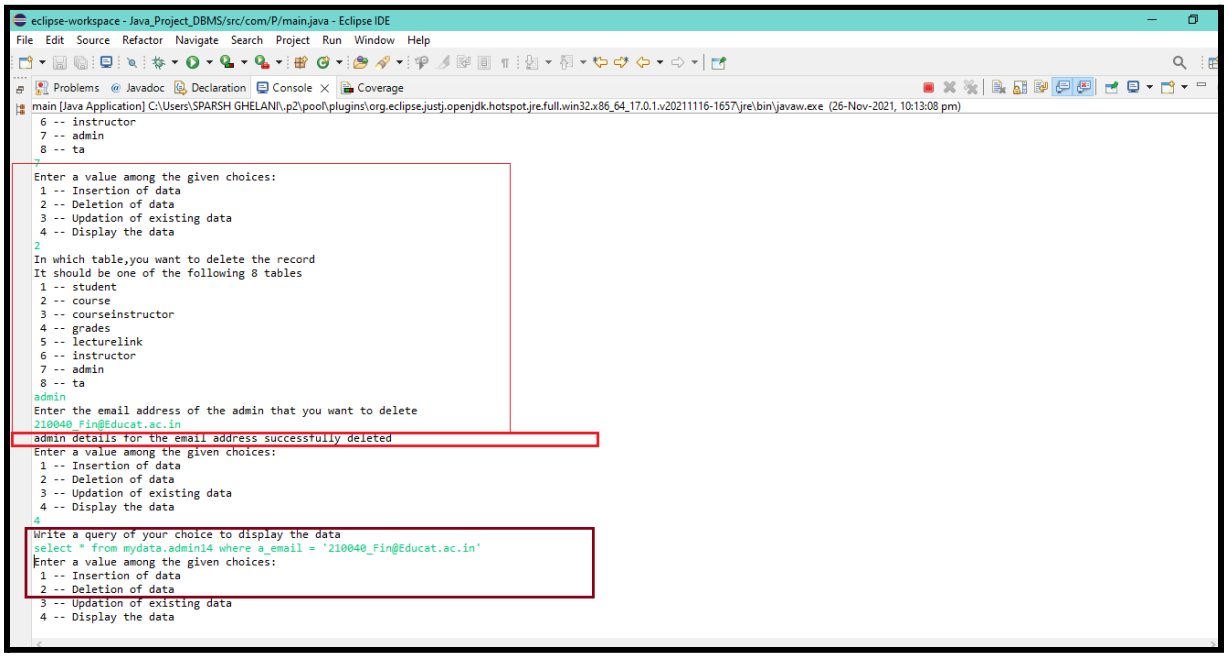


3. Insertion of values in the table.



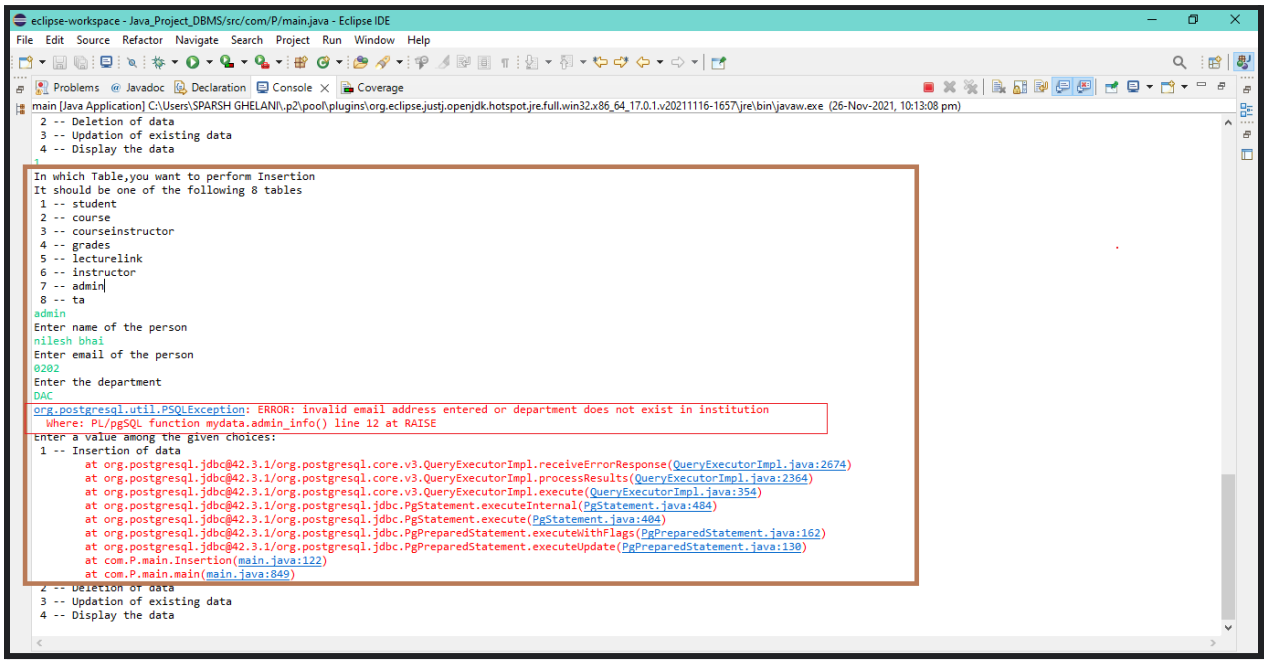


4. Deletion of values from table.



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

Inserting values into admin



Inserting values into grade table