DATABASE FOR QUIZ PORTAL

by The Query Crew

Gaayathri Vaidhyanathan Pazhambalacode

Venkata Krishna Chaitanya Nuthalapati

Karishma Mahendra Wadhe

Romina Karimaei

INDEX

1. System Requirements

Purpose/Aim:

In this database, the quiz portal will be used for conducting quizzes and practical exams on different courses taught by their professors. The quiz portal would reduce the workload on the professors and the evaluation of the students would be fair and unbiased. The portal would also help in conducting quizzes in a timely manner. The students can check their marks and if they have any problem with the evaluation, they can reach out to the professor directly. The professor would have the record of each student and every quiz conducted by him, which will be maintained in this database. Here, the students can opt for nicknames, but their scores need to be linked to their profile. The quiz, questions, answers, scores, grades, and submissions status need to be recorded and used which helps with efficient maintenance of this database. Here, the professor can decide on which quiz is considered as homework, assignment, project, or mids and the form of submission can be decided by the professor conducting the quiz.

1.1 Entity Types:

a) User Entity:

The user entity is basically the students who take the quizzes by the professor based on the subjects they opt for. The attributes in this entity include campus id (primary key), student name (first name, last name), nickname (should be anonymous).

b) Subject Entity:

The subject entity has the subjects which are a part of the curriculum and taught by various professors. This entity contains the subject code (primary key) along with the subject name.

c) Professor Entity:

The professor entity describes the various professors who are involved in conducting the quizzes for the students with respect to the subjects they teach. This entity contains the professor's name, employee id (primary key).

d) Quiz Entity:

The quiz entity has the record of the different quizzes which are conducted by the professors subject-wise for their respective students. It contains a unique key as quiz name, quiz id (primary key) and the respective subject code for which that quiz is related to.

e) Questionnaire Entity:

The questionnaire entity is used to store the questions for the quizzes that are taken by the students according to their subjects. It contains question number (primary key), the actual questions that are asked in the quiz and the total marks allotted for each question.

f) **Performance Entity:**

The performance entity tracks the overall progress of the student with respect to the tasks completed by him/her. The performance of the student is tracked by their homework marks, assignment marks, mid-term marks, final exam marks, project marks which are all derived attributes. The performance is calculated based on the total percentage obtained according to the weightage of individual exam/activity.

g) **Response Entity:**

The response entity keeps a track of the answers for the questions that are asked in the quizzes along with the marks obtained by the students for their response on that question.

1.2. Relationship Types:

a) "Enrolls" Relationship:

This relationship is between the user entity and subject entity that allows the user to enroll for a subject to give the quiz on. A user can select a subject from the list of subjects for the quiz. It is a many-to-many relationship where a user can select many subjects and a subject can be enrolled by any number of students.

b) "Teaches" Relationship:

This relationship is between the subject entity and the professor entity. This is also a many-to-many relationship where a professor can teach multiple subjects and a subject can be taught by different professors.

c) "Conducts" Relationship:

This is between the professor entity and the Quiz entity. Professor conducts an exam/quiz to attempt. A professor can conduct multiple quizzes. Also, one quiz is usually designed by multiple professors.

d) "Contains" Relationship:

This is between the Quiz entity and questionnaire entity. One exam has multiple questions. A question belongs to a particular exam only.

e) "Responds" Relationship:

This relationship is between the User Entity and the Response Entity. It is mandatory for a user to respond to a question in the quiz. A user can attempt the quiz only once.

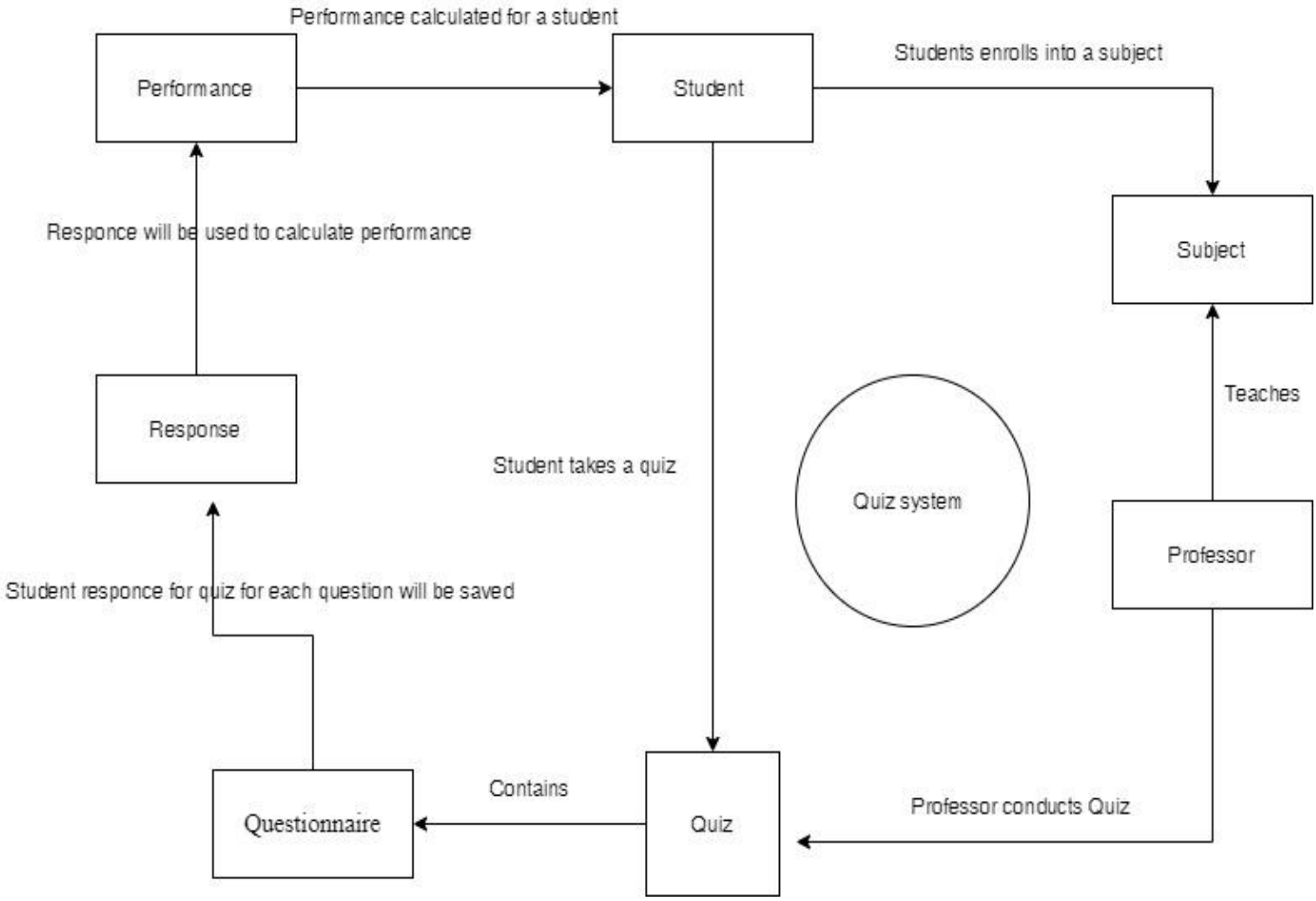
f) "Attempts" Relationship:

A relation that maps Questionnaire, and response entities. There needs to be at least one response to every question in the questionnaire. There will be multiple attempts (one from each user) to the questionnaire.

g) "Progress" Relationship:

A relation between the user and his performance. It has performances of every single user.

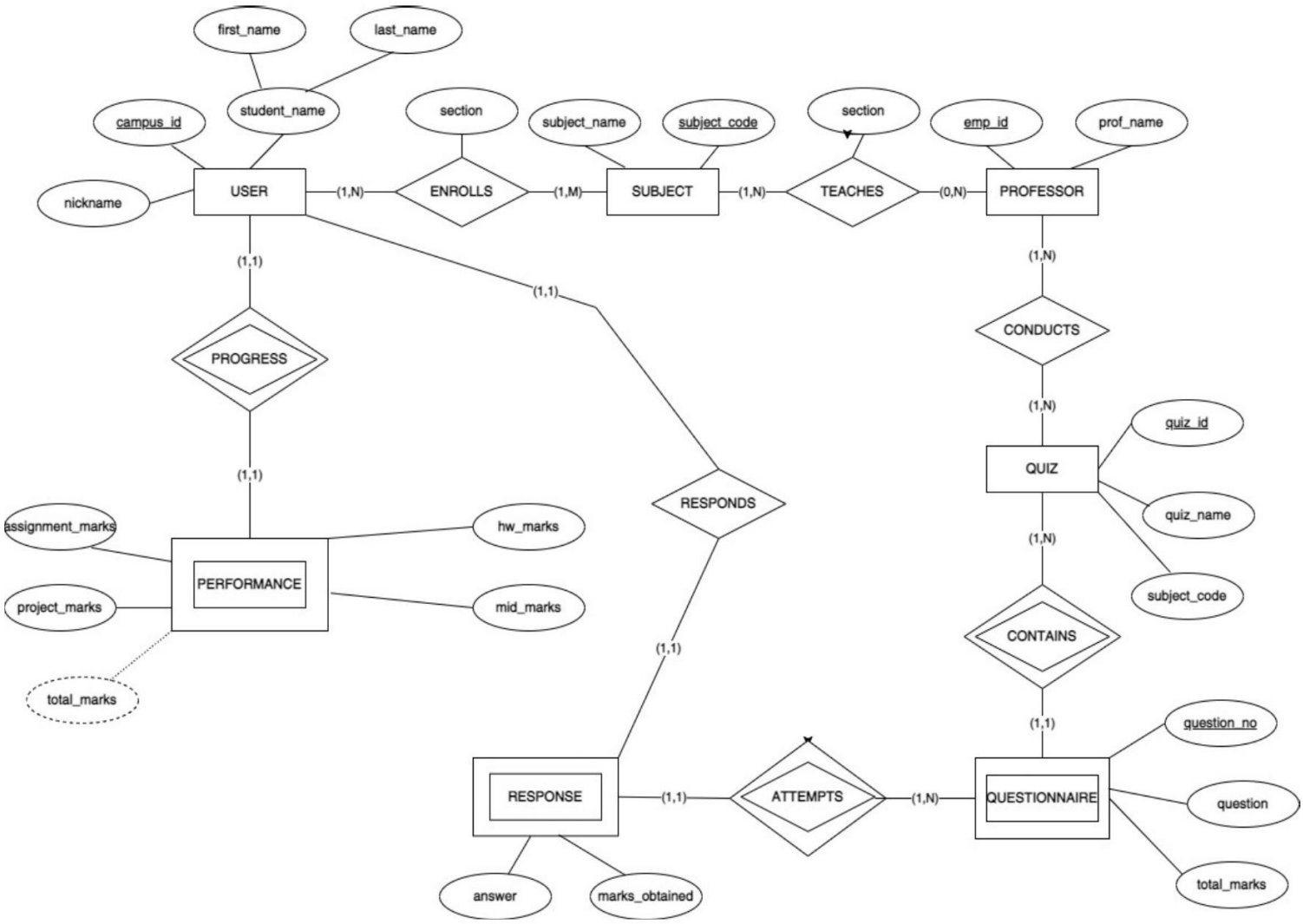
2. Contextual Data Flow Diagram



Contextual diagram shows the system under consideration as a single high level process and then shows the relationship that the system might have. This is basically a system that points out the flow of the information and external entities. It is made up of a context bubble. This diagram has so many benefits such as it will show the scope and boundaries of a system at a glance. In addition to that, no technical knowledge is required to understand the diagram. This diagram system is easy to draw and it has limited notation.

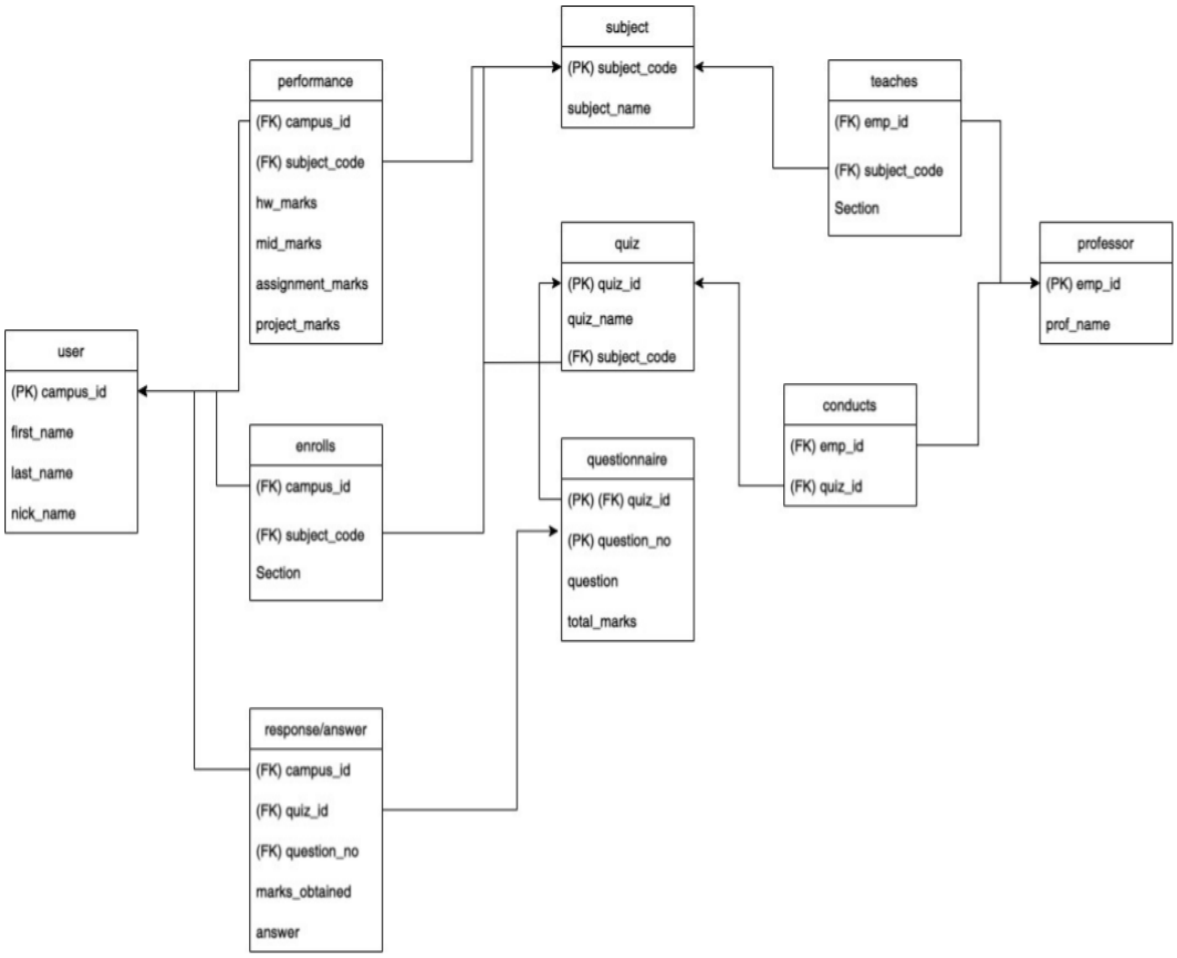
3. ER/EER Model in Min, Max notation

The ER Model in Min, Max notation is as follows:



Each system is using its own notation and system. Ann notational is showing entities as rectangular boxes and relationships as lines connecting boxes. Each of these use its own special symbol to represent a connection between entities.

4. Database Schema and Normalized DB Model



The above normalized and implementation-ready DB model is in Third Normal Form (3NF). Normalization is a technique for organizing data in a database. It is important that a database is normalized to minimize redundancy and to ensure only related data is stored and nothing else. This technique would prevent any issues from database modifications such as updates, insertion and deletion.

4.1 Reasons to choose 3rd Normal Form for this database

- **The 3rd normal form removes any data redundancy.**

  Tables in second normal form are especially vulnerable to some types of modification anomalies — in particular, those that come from transitive dependencies.

  A *transitive dependency* occurs when one attribute depends on a second attribute, which depends on a third attribute. Deletions in a table with such a dependency can cause unwanted information loss. A relation in third normal form is a relation in second normal form with no transitive dependencies.Look again at the SALES table, which you know is in first normal form. As long as you constrain entries to permit only one row for each Customer_ID, you have a single-attribute primary key, and the table is in second normal form.

  Selecting any other normal form below the 3rd normal form will not be the best method to prevent data redundancy and to show functional dependency between attributes.

  Normalization is one one of the best and the most important part of the database. It is very difficult sometimes to separate these two from each other. You can use ERD to provide the big picture or provide more room and window. A 3NF is a relation in the 3rd normal form and if there is none transitive dependency for non prime attributes as well.In the *first normal form*, only single values are permitted at the intersection of each row and column; hence, there are no repeating groups.To normalize a relation that contains a repeating group, remove the repeating group and form two new relations.The PK of the new relation is a combination of the PK of the original relation plus an attribute from the newly created relation for unique identification.

- **Reduces the amount of storage required which makes the database more efficient.**

Data storage points to the basic problem in information security analysis are scattered in major numbers and a lot of big data and logistics. There are many different approaches such as flat flies and traditional databases. Flat flies system records data and they are accessed directly by analysts. This is basically done as a simple parsing tool. Most log systems can create or delete a file. Flat system is usually simple to read and analyze.

- **Selecting any high level of normal form will increase complexity for data insertion, and selection queries.**

If the most simple and easiest in a query does not perform right or it's having issues, it is because it's working with so much other data that it conflicts with other stuff. Sometimes queries are demanding more work space due to how heavy the file could be. It is sometimes useful to see and think about how the numbers of rows are examined when analyzing queries because you can see and notice how efficient queries can be. If the queries have a high level of normal form will increase the complexity, we need to make sure the workspace has enough storage.

5. Recommendation on DBMS and rationale for your choice

Recommended Database: PostgreSQL

Deciding factors to choose PostgreSQL:

- **Database Performance:**

MYSQL is a bit faster than PostgreSQL but in the recent updates the difference is minimal. Postgres is relatively better at handling concurrency. Unless the performance and speed are major factors MySQL need not be preferred over Postgres.

- **Object-relational database:**

  As Postgres is an object-related database, we can use features like inheritance and function overloading. This object-relational database is utilized to bridge the gap between relational databases and the object-oriented modeling techniques used in programming languages like Java, C++ etc, which makes it much more powerful for creating databases.

- **Data-integrity:**

  Due to a better ability of concurrency control, Postgres can handle concurrency of data more efficiently. So in case of students taking exams all at once, Postgres can be a great choice to keep data integrity intact.

- **Extensibility:**

  Postgres is more extensible compared to MySQL and other databases available. We can use multiple data types such as GIS, network address types, JSONB which can be indexed, native UUID, timezone-aware timestamps. If required we can create our own database, operators, and index types. This can create ease for us to build a database for our requirements.

- **Open-source:**

  Postgres is open-sourced and highly community-driven. MYSQL is open-source but in a few cases, the license issues are common in real-world scenarios.

6. Data Dictionary

In ROLE: E-Entity Type
R N:M-Relationship (N:M) Type
A-Attribute

DATA DICTIONARY

NAME	SHORT DESCRIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
User	Student accessing quiz portal	E	N/A	N/A	N/A	N/A
campus_id	Unique ID for user/student	A	char(9)	Unique	No Null Values	N/A
first_name	First name of the student	A	varchar(25)	N/A	Null allowed	N/A
last_name	Last name of the student	A	varchar(25)	N/A	Null allowed	N/A
nick_name	Anonymous name to hide student i	A	varchar(40)	Unique	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Enrolls	Student "enrolling" to courses	R N:M	N/A	N/A	N/A	N/A
campus_id	Unique ID of student	A	char(9)	N/A	No Null Values	N/A
subject_code	Unique code/ID for subjects	A	char(8)	N/A	No Null Values	N/A
section_name	which section in that subject the st	A	char(3)	N/A	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Subject	Subject in which quiz is conducted	E	N/A	N/A	N/A	N/A
subject_code	Unique code/ID for subjects	A	char(8)	Unique	No Null Values	N/A
subject_name	Name of the subject	A	varchar(25)	N/A	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Professor	Teacher teaching the subject	E	N/A	N/A	N/A	N/A
emp_id	employee id of the professor	A	char(9)	Unique	No Null Values	N/A
prof_name	name of the professor	A	varchar(50)	N/A	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Teaches	Relationship telling subject taught	R N:M	N/A	N/A	N/A	N/A
emp_id	employee id of the professor	A	char(9)	N/A	No Null Values	N/A
subject_code	Unique code/ID for subjects	A	char(8)	N/A	No Null Values	N/A
section_name	which section in that subject the pr	A	char(3)	N/A	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Quiz	Entity for quiz conducted by the pro	E	N/A	N/A	N/A	N/A
quiz_id	Unique ID for quiz	A	char(4)	Unique	No Null Values	N/A
subject_code	Unique code/ID for subjects	A	char(8)	N/A	No Null Values	N/A
quiz_name	Name of the quiz conducted	A	varchar(25)	N/A	Null allowed	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Conducts	Professor "conducts" quiz relation	R N:M	N/A	N/A	N/A	N/A
emp_id	employee id of the professor	A	char(9)	N/A	No Null Values	N/A
quiz_id	Unique ID for quiz	A	char(4)	N/A	No Null Values	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Questionnaire	List of Questions in a quiz	E	N/A	Weak Entity of Quiz	N/A	N/A
quiz_id	Unique ID for quiz	A	char(4)	N/A	No Null Values	N/A
question_no	Question number of the question	A	int	N/A	No Null Values	N/A
question	The question to be asked	A	varchar(100)	N/A	Null allowed	N/A
total_marks	marks carried by a question	A	int	N/A	Null allowed	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Response	Answer for a question in quiz	E	N/A	Weak Entity of Questionnaire	N/A	N/A
campus_id	Unique ID for user/student	A	char(9)	N/A	No Null Values	N/A
quiz_id	Unique ID for quiz	A	char(4)	N/A	No Null Values	N/A
question_no	Question number of the question	A	int	N/A	No Null Values	N/A
answer	Answer to a question by student	A	varchar(500)	N/A	Null allowed	N/A
marks_obtained	marks obtained for a question by st	A	float	N/A	Null allowed	N/A

NAME	SHORT DESCIPTION	ROLE	TYPE	FORMAT	NULL ALLOWED	DEFAULT VALUE (if any)
Performance	Evaluation of students	R N:M	N/A	N/A	N/A	N/A
campus_id	Unique ID for user/student	A	char(9)	N/A	No Null Values	N/A
subject_code	Unique code/ID for subjects	A	char(8)	N/A	No Null Values	N/A
hw_marks	Marks obtained for HW by student	A	float	N/A	Null allowed	N/A
assignment_marks	Marks obtained for assignment by	A	float	N/A	Null allowed	N/A
mid_marks	Marks obtained for mid-term by stu	A	float	N/A	Null allowed	N/A
project_marks	Marks obtained for project by stude	A	float	N/A	Null allowed	N/A

7. SQL Queries with Constraints and some useful queries to get started

Inserting data into student table

7.1 Tables Creation:

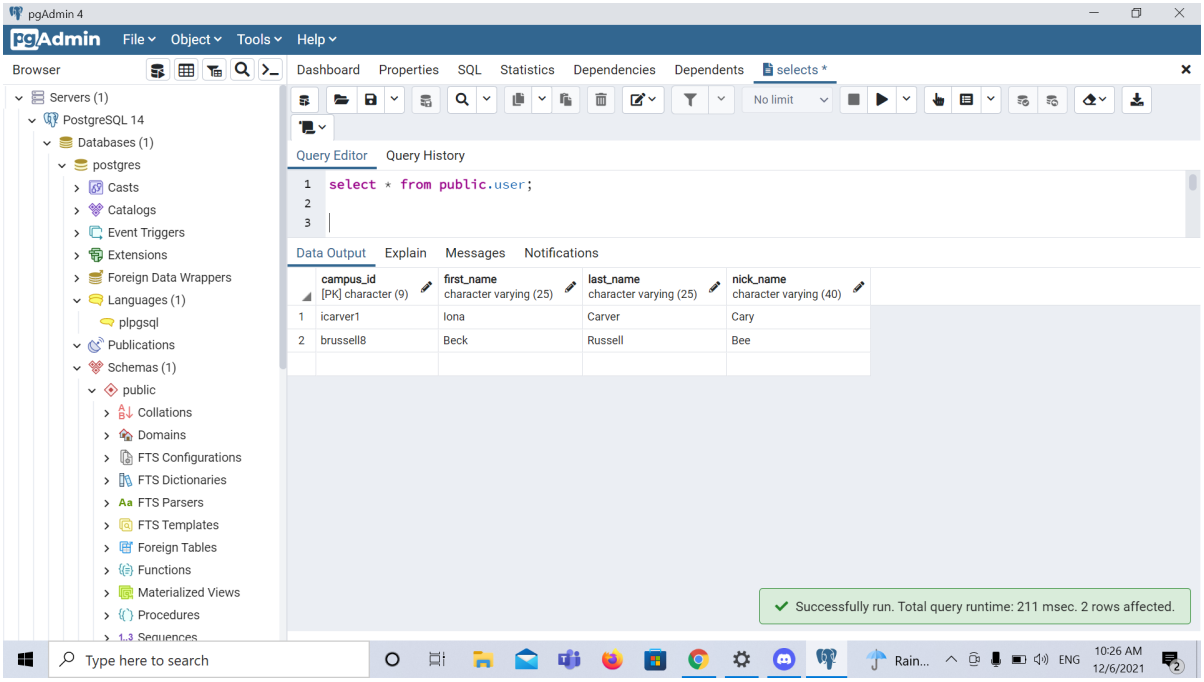
```
CREATE TABLE public.user(campus_id char(9) NOT NULL UNIQUE,
                         first_name varchar(25),
                         last_name varchar(25),
                         nick_name varchar(40) NOT NULL UNIQUE,
                         constraint pk_user PRIMARY KEY(campus_id));

CREATE TABLE subject(subject_code char(8) NOT NULL UNIQUE,
                     subject_name varchar(25) NOT NULL,
                     constraint pk_subject PRIMARY KEY(subject_code));


CREATE TABLE enrolls(campus_id char(9) NOT NULL,
                     subject_code char(8) NOT NULL,
                     section_name char(3) NOT NULL,
                     FOREIGN KEY (campus_id)
                      REFERENCES  public.user(campus_id),
          FOREIGN KEY (subject_code) REFERENCES subject(subject_code));

CREATE TABLE professor(emp_id char(9) NOT NULL UNIQUE,
                       prof_name varchar(50) NOT NULL,
                       constraint pk_professor PRIMARY KEY(emp_id));

CREATE TABLE teaches(emp_id char(9) NOT NULL,
                     subject_code char(8) NOT NULL,
                     section_name char(3) NOT NULL,
                     FOREIGN KEY (emp_id) REFERENCES professor(emp_id),
                     FOREIGN KEY (subject_code)
                      REFERENCES  subject(subject_code));

CREATE TABLE quiz(quiz_id char(4) NOT NULL UNIQUE,
                  subject_code char(8) NOT NULL,
                   quiz_name varchar(25),
                   constraint pk_quiz PRIMARY KEY(quiz_id),
            FOREIGN KEY (subject_code) REFERENCES subject(subject_code));
```

```
CREATE TABLE conducts(emp_id char(9) NOT NULL,
                      quiz_id char(4) NOT NULL,
                      FOREIGN KEY (emp_id) REFERENCES professor(emp_id),
                      FOREIGN KEY (quiz_id) REFERENCES quiz(quiz_id));

CREATE TABLE questionnaire(quiz_id char(4) NOT NULL,
                      question_no int NOT NULL,
                      question varchar(100),
                       total_marks int,
                      constraint pk_questionnaire PRIMARY KEY(quiz_id, question_no),
                       FOREIGN KEY (quiz_id) REFERENCES quiz(quiz_id));

CREATE TABLE response(campus_id char(9) NOT NULL,
                      quiz_id char(4) NOT NULL,
                      question_no int NOT NULL,
                      answer varchar(500),
                      marks_obtained float,
                      FOREIGN KEY (campus_id)
                       REFERENCES public.user(campus_id),
                      FOREIGN KEY (quiz_id, question_no)
                       REFERENCES  questionnaire(quiz_id, question_no));

CREATE TABLE performance(campus_id char(9) NOT NULL,
                      subject_code char(8) NOT NULL,
                      hw_marks float,
                      assignment_marks float,
                      mid_marks float,
                      project_marks float,
                      FOREIGN KEY (campus_id)
                       REFERENCES public.user(campus_id),
                      FOREIGN KEY (subject_code)
                       REFERENCES subject(subject_code));
```

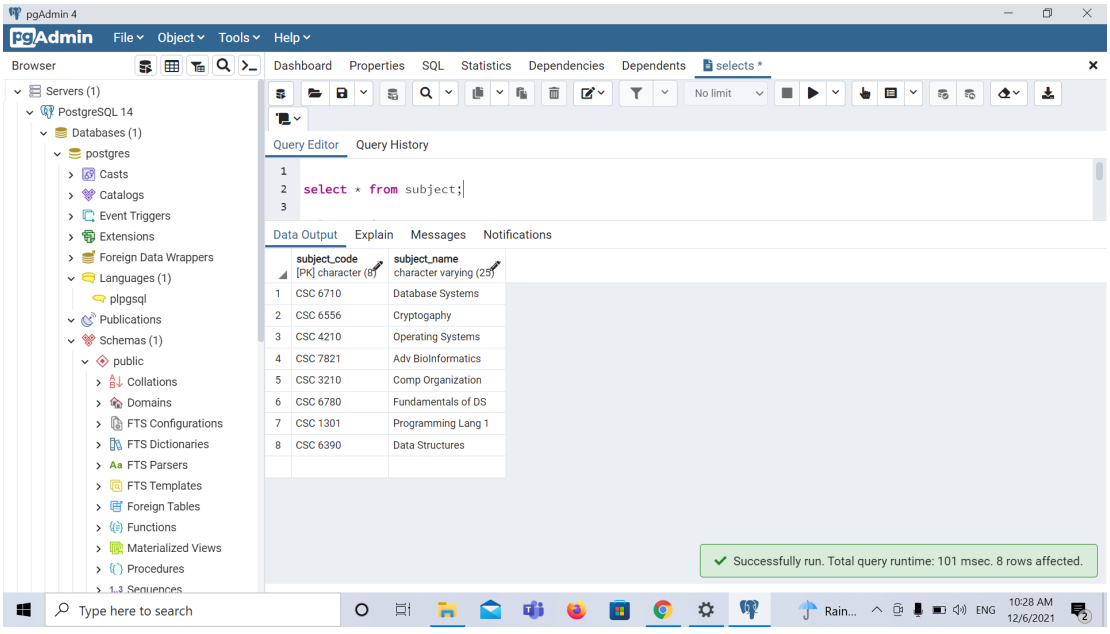
Inserting Student data

7.2 Inserting Values:

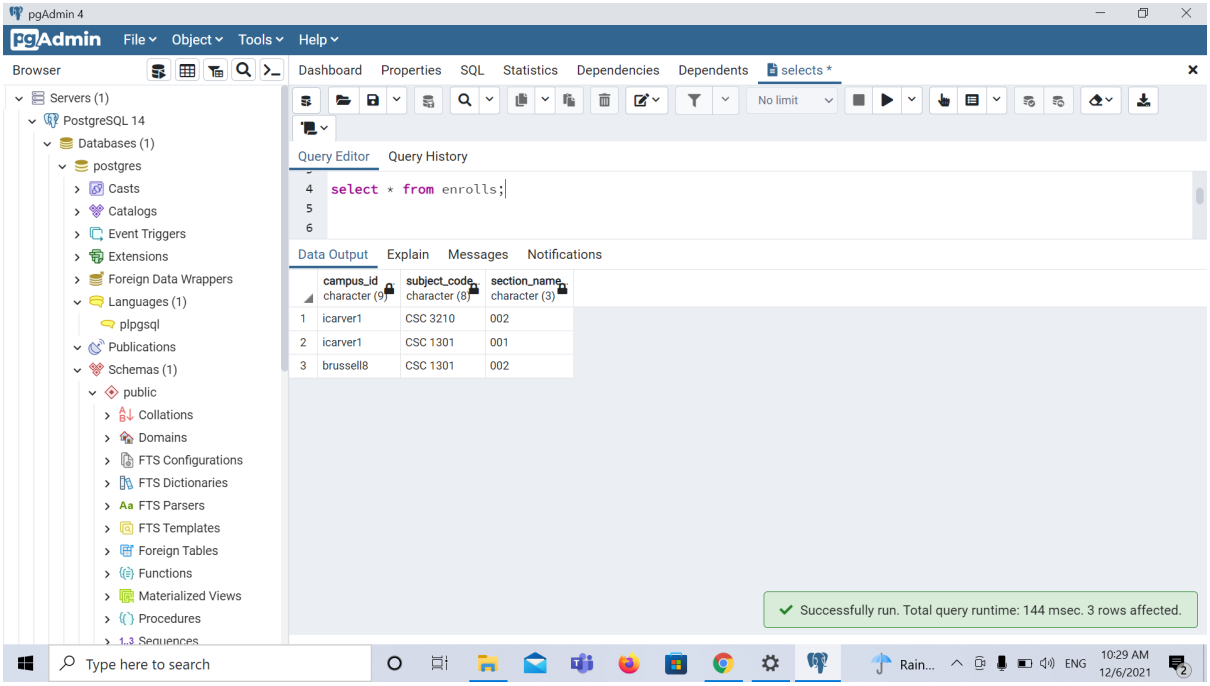
```
insert into public.user ("nick_name", "campus_id", "first_name" , "last_name")
values('Cary', 'icarver1', 'Iona', 'Carver'),
        ('Bee', 'brussell8', 'Beck', 'Russell');
```



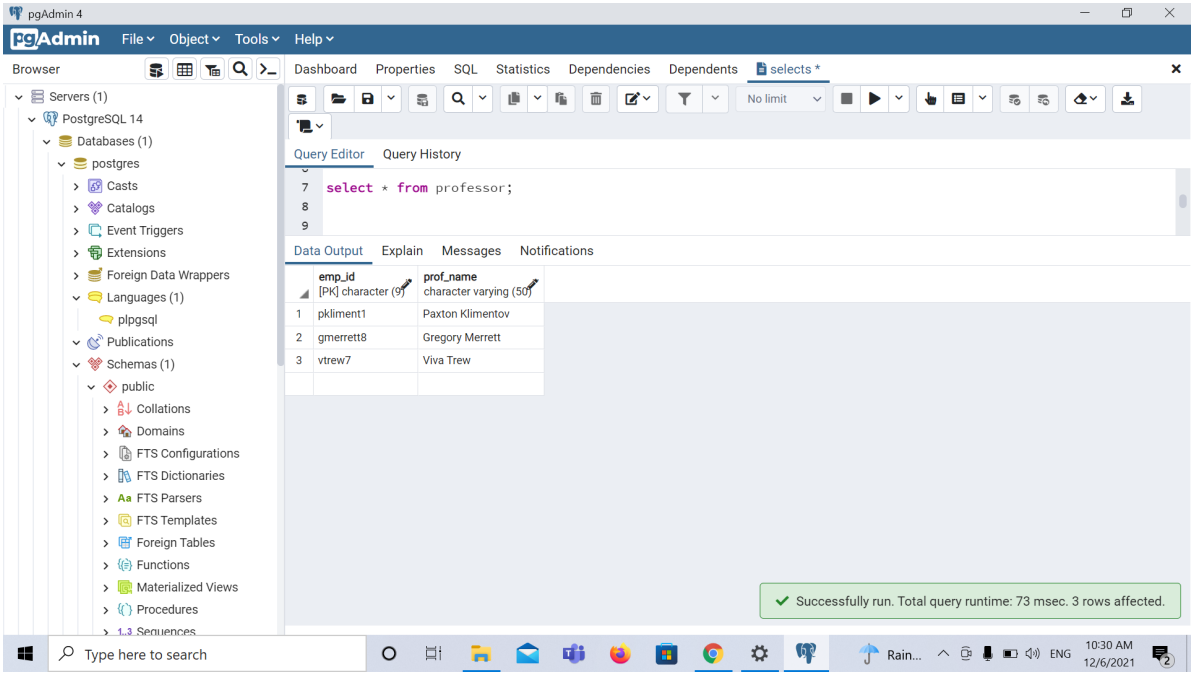

```
insert into subject ("subject_code", "subject_name")
values ('CSC 6710','Database Systems'),
        ('CSC 6556', 'Cryptography'),
        ('CSC 4210', 'Operating Systems'),
      ('CSC 7821', 'Adv BioInformatics'),
      ('CSC 3210', 'Comp Organization'),
      ('CSC 6780', 'Fundamentals of DS'),
      ('CSC 1301', 'Programming Lang 1'),
      ('CSC 6390', 'Data Structures');
```



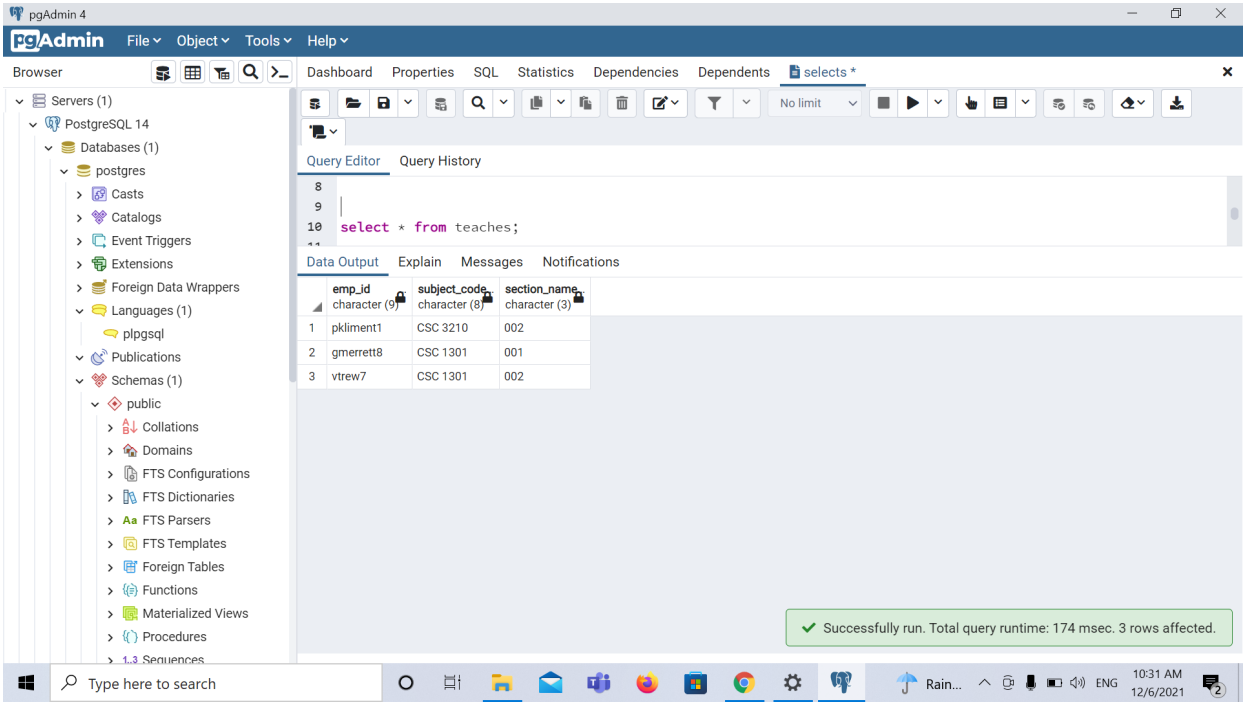
```
insert into enrolls ("campus_id", "subject_code", "section_name")
values('icarver1', 'CSC 3210', '002'),
      ('icarver1', 'CSC 1301', '001'),
      ('brussell8', 'CSC 1301', '002');
```



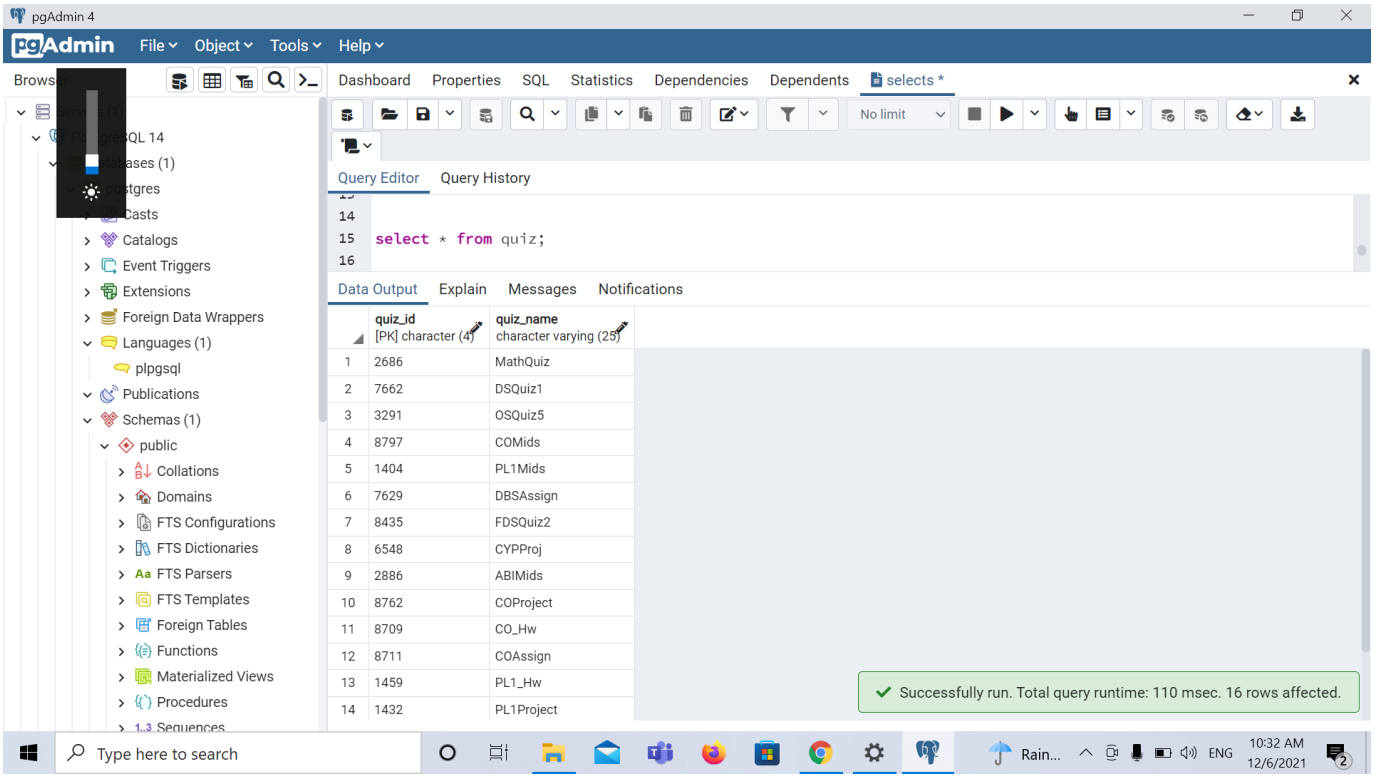

```
insert into professor ("emp_id", "prof_name")
values('pkliment1','Paxton Klimentov'),
      ('gmerrett8', 'Gregory Merrett'),
      ('vtrew7','Viva Trew');
```



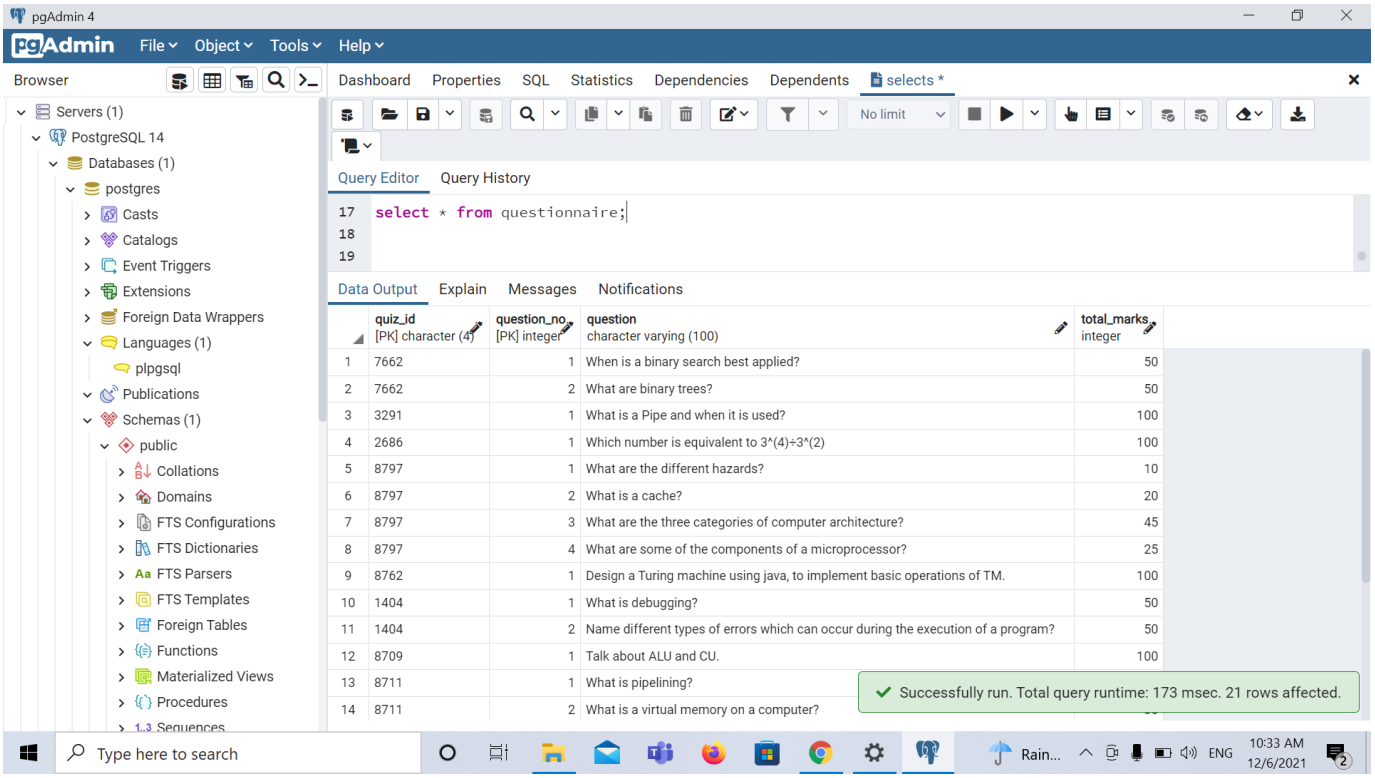
```
insert into teaches("emp_id", "subject_code", "section_name")
values('pkliment1', 'CSC 3210', '002'),
        ('gmerrett8', 'CSC 1301', '001'),
        ('vtrew7','CSC 1301', '002');
```



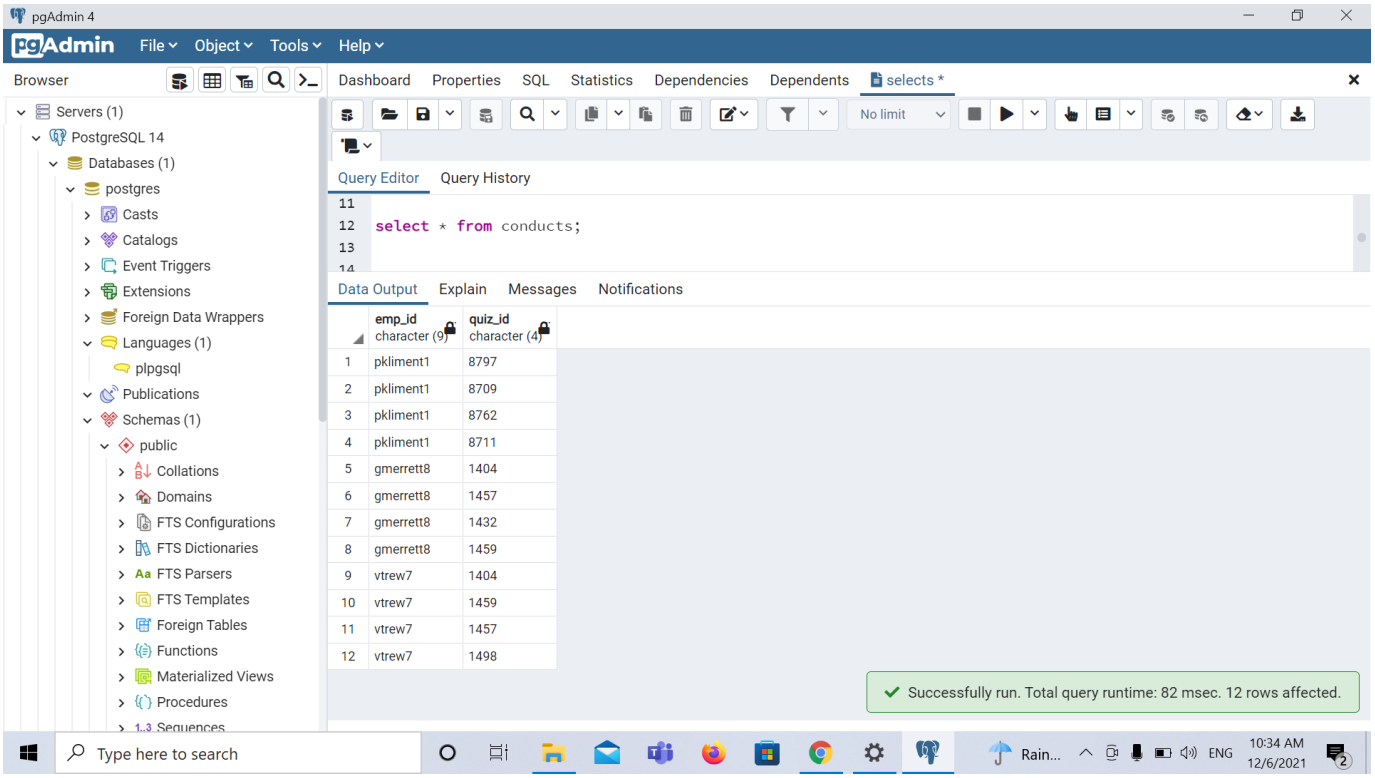

```
insert into quiz("quiz_id", "quiz_name", "subject_code")
values('7662', 'DSQuiz1', 'CSC 6390'),
        ('3291', 'OSQuiz5', 'CSC 4210'),
        ('8797', 'COMids', 'CSC 3210'),
        ('1404', 'PL1Mids', 'CSC 1301'),
        ('7629', 'DBSAssign', 'CSC 6710'),
        ('8435', 'FDSQuiz2', 'CSC 6780'),
        ('6548', 'CYPProj', 'CSC 6556'),
        ('2886', 'ABIMids', 'CSC 7821'),
        ('8762', 'COProject', 'CSC 3210'),
        ('8709', 'CO_Hw', 'CSC 3210'),
        ('8711', 'COAssign', 'CSC 3210'),
        ('1459', 'PL1_Hw', 'CSC 1301'),
        ('1432', 'PL1Project', 'CSC 1301'),
        ('1457', 'PL1Assign', 'CSC 1301'),
        ('1498', 'PL1_Projectwork', 'CSC 1301');
```




```
insert into questionnaire("quiz_id", "question_no", "question", "total_marks")
values('7662', 1, 'When is a binary search best applied?', 50),
        ('7662', 2, 'What are binary trees?', 50),
        ('3291', 1, 'What is a Pipe and when it is used?', 100),
        ('8797', 1, 'What are the different hazards?', 10),
        ('8797', 2, 'What is a cache?', 20),
        ('8797', 3, 'What are the three categories of computer architecture?', 45),
        ('8797', 4, 'What are some of the components of a microprocessor?', 25),
        ('8762', 1, 'Design a Turing machine using java, to implement basic operations of TM.', 100),
        ('1404', 1, 'What is debugging?', 50),
        ('1404', 2, 'Name different types of errors which can occur during the execution of a
program?',  50),
        ('8709', 1, 'Talk about ALU and CU.', 100),
        ('8711', 1, 'What is pipelining?', 35),
        ('8711', 2, 'What is a virtual memory on a computer?', 35),
        ('8711', 3, 'What are the different types of fields that are part of instruction?', 30),
        ('1459', 1, 'Briefly explain the various types of loops.', 100),
        ('1432', 1, 'Graph-coloring register allocation', 100),
        ('1457', 1, 'What are constants? Explain their types.', 20),
        ('1457', 2, 'Please explain program documentation. Why is it important?', 50),
        ('1457', 3, 'Please explain the operators.', 30),
        ('1498', 1, 'Build a Working Chess Game.', 100);
```

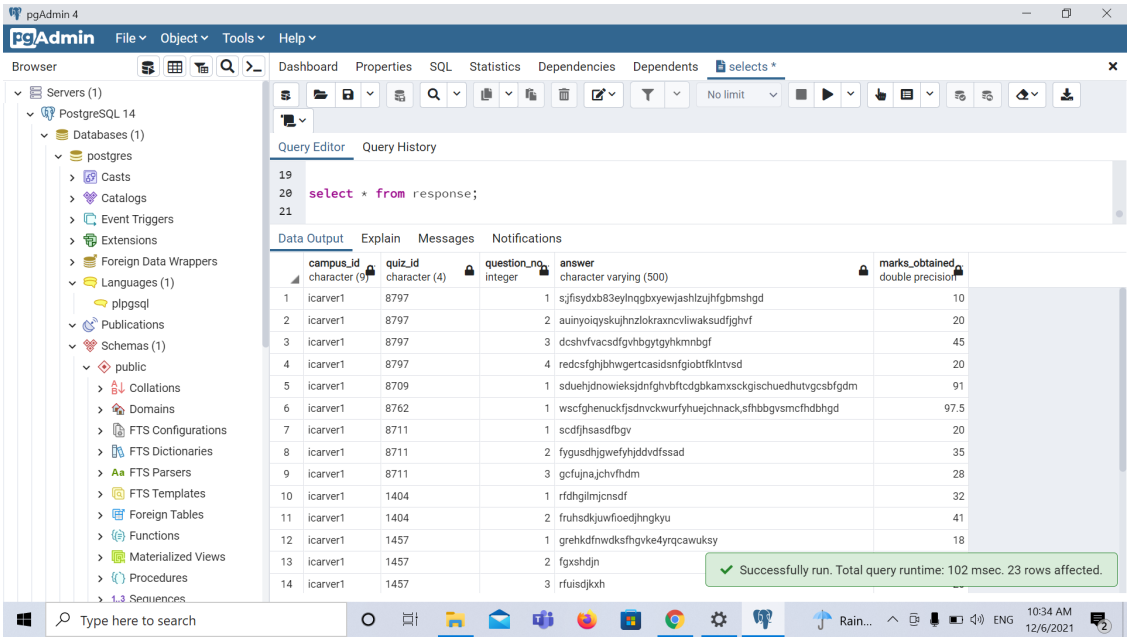



```
 insert into conducts ("emp_id", "quiz_id")
values('pkliment1', '8797'),
        ('pkliment1', '8709'),
        ('pkliment1', '8762'),
        ('pkliment1', '8711'),
        ('gmerrett8', '1404'),
        ('gmerrett8', '1457'),
        ('gmerrett8', '1432'),
        ('gmerrett8', '1459'),
        ('vtrew7','1404'),
        ('vtrew7', '1459'),
        ('vtrew7', '1457'),
        ('vtrew7', '1498');
```

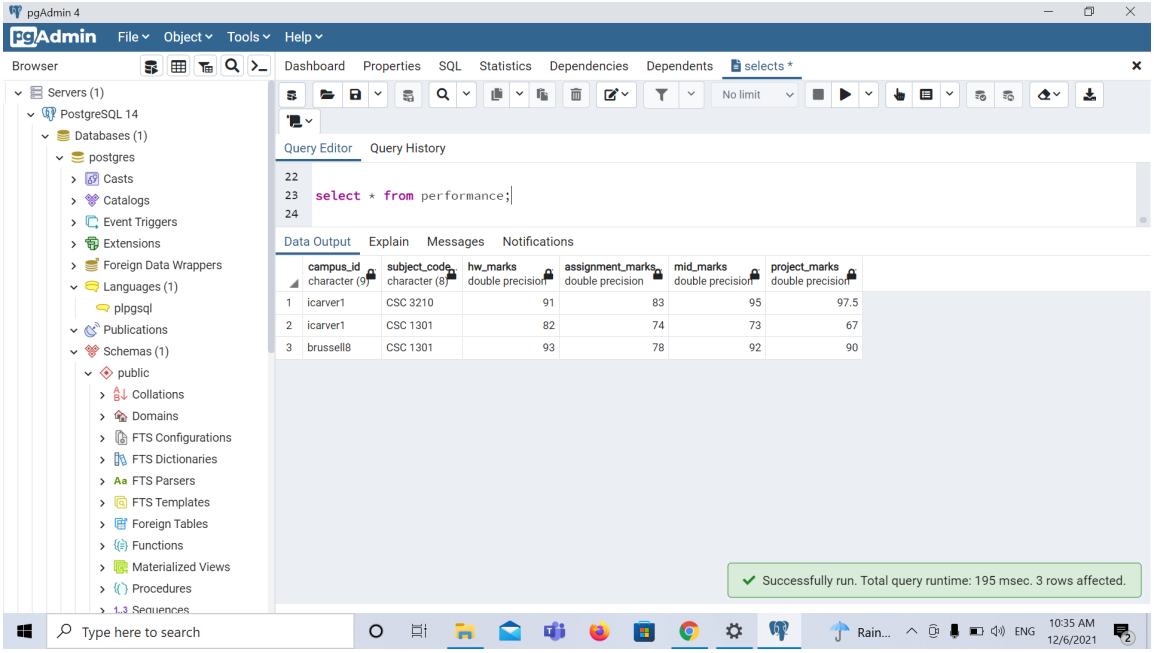



```
insert into response ("campus_id", "quiz_id", "question_no", "answer", "marks_obtained")
values('icarver1', '8797', 1, 's;jfisydxb83eylnqgbxyewjashlzujhfgbmshgd', 10),
        ('icarver1', '8797', 2, 'auinyoiqyskujhnzlokraxncvliwaksudfjghvf', 20),
        ('icarver1', '8797', 3, 'dcshvfvacsdfgvhbgytgyhkmnbgf', 45),
        ('icarver1', '8797', 4, 'redcsfghjbhwgertcasidsnfgiobtfklntvsd', 20),
        ('icarver1', '8709', 1, 'sduehjdnowieksjdnfghvbftcdgbkamxsckgischuedhutvgcsbfgdm', 91),
        ('icarver1', '8762', 1, 'wscfghenuckfjsdnvckwurfyhuejchnack,sfhbbgvsmcfhdbhgd', 97.5),
        ('icarver1', '8711', 1, 'scdfjhsasdfbgv', 20),
        ('icarver1', '8711', 2, 'fygusdhjgwefyhjddvdfssad', 35),
        ('icarver1', '8711', 3, 'gcfujna,jchvfhdm', 28),
        ('icarver1', '1404', 1, 'rfdhgilmjcnsdf', 32),
        ('icarver1', '1404', 2, 'fruhsdkjuwfioedjhngkyu', 41),
        ('icarver1', '1457', 1 , 'grehkdfnwdksfhgvke4yrqcawuksy', 18),
       ('icarver1', '1457', 2, 'fgxshdjn', 36),
       ('icarver1', '1457', 3, 'rfuisdjkxh', 20),
       ('icarver1', '1432', 1, 'eghfiukjxfiuwexhn,wfesdkx', 67),
       ('icarver1', '1459', 1, 'gyekudfchjnetidukvgservyu4iry5ieskrhd', 82),
       ('brussell8', '1404', 1, 'yruhdsjdgcw3qigukgedhvfchweuqkawjoi5l', 42.5),
       ('brussell8', '1404', 2, 'yuerhdws,ales,jkfchgnukerjhdnfirks', 49.5),
       ('brussell8', '1457', 1, 'ifukdjhfdgvdrmxs', 20),
       ('brussell8', '1457', 2, 'vhikn,ese,nvhiod', 39),
       ('brussell8', '1457', 3, 'hdsakofiurjhfxszaa', 19),
       ('brussell8', '1459', 1, 'gsdhuk92sqo2ijenhg7iww3ywufhasfgd3w', 93),
```

('brussell8', '1498', 1, 'werydhuwijkfegduqjwesdnfbd3qiak', 90);



insert into performance ("campus_id", "subject_code", "hw_marks", "assignment_marks", "mid_marks", "project_marks")
values('icarver1', 'CSC 3210', 91, 83, 95, 97.5),
('icarver1', 'CSC 1301', 82, 74, 73, 67),
('brussell8', 'CSC 1301', 93, 78, 92, 90);



The above queries are used to implement the database and to insert some initial data for our Quiz Portal database.

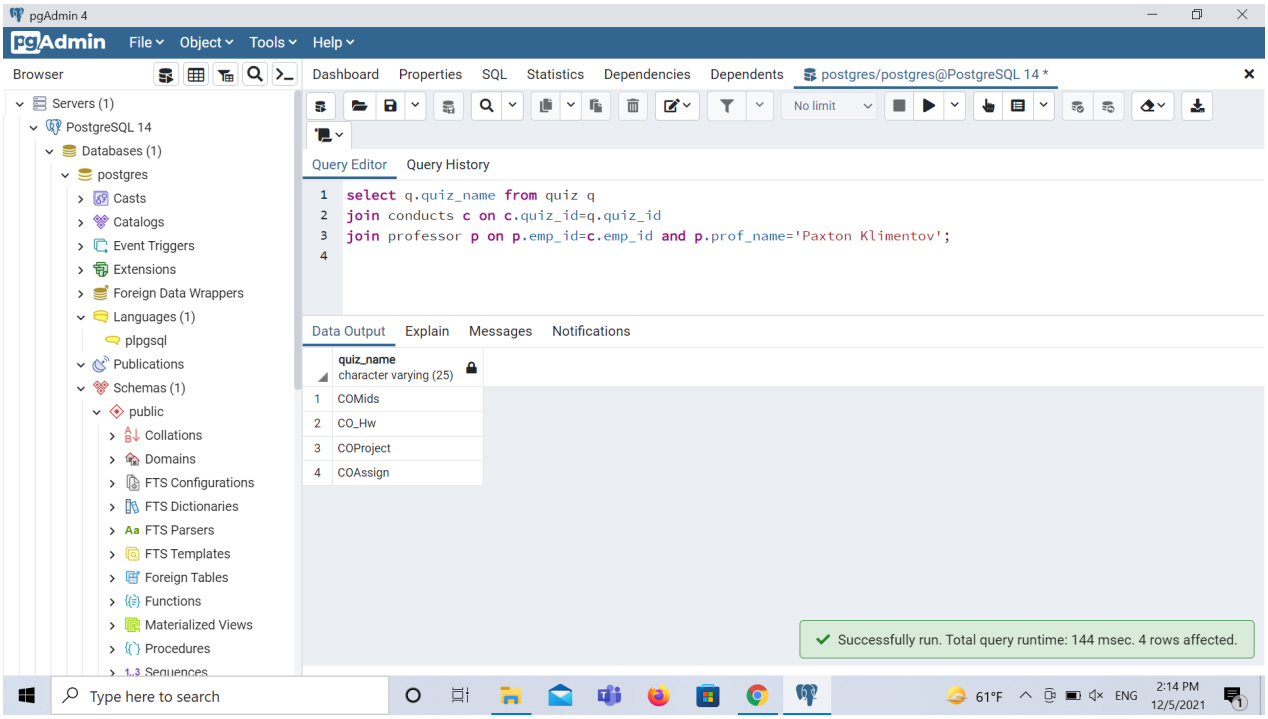
Following are some useful queries that can help users to retrieve relevant data:

7.3 Queries:

First display all the tables then implement the following queries:

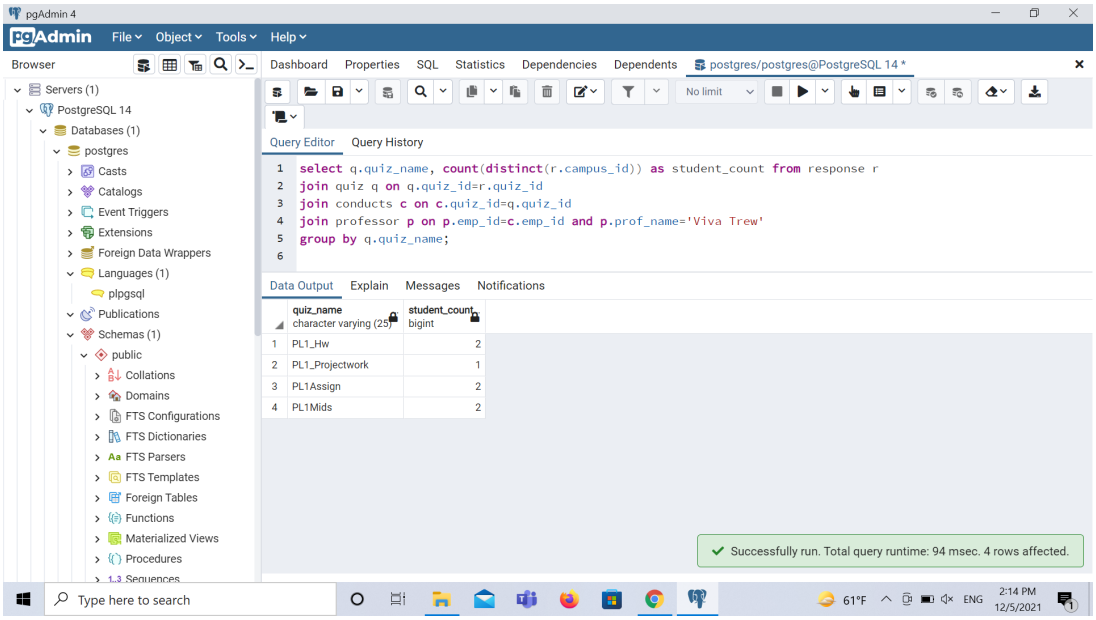
1. Display the quizzes conducted by professor Paxton Klimentov.

A:
```
select q.quiz_name from quiz q
join conducts c on c.quiz_id=q.quiz_id
join professor p on p.emp_id=c.emp_id and p.prof_name='Paxton Klimentov';
```



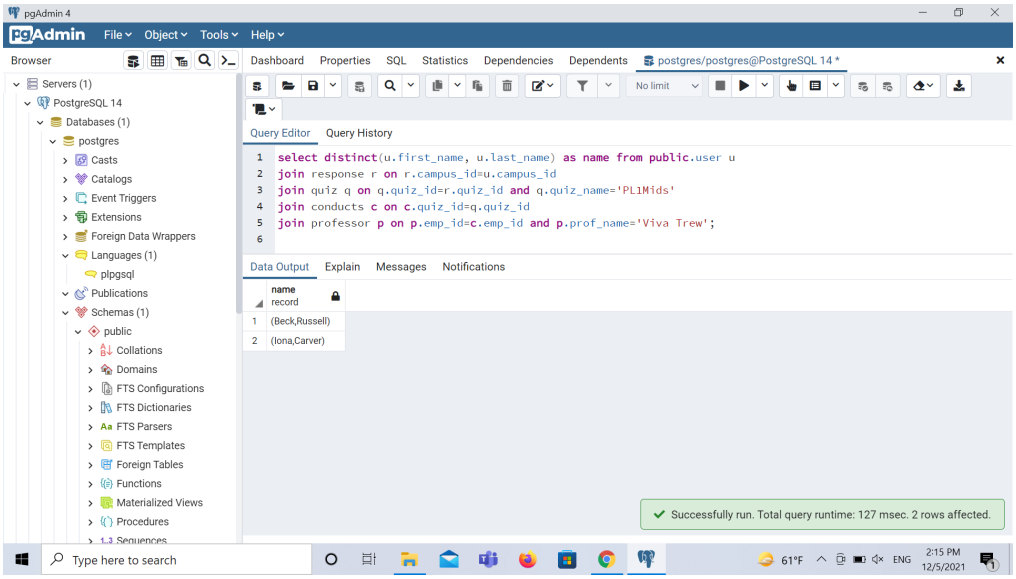
2. Display the number of students who attempted various quizzes conducted by professor Viva Trew.

A:
```
select q.quiz_name, count(distinct(r.campus_id)) as student_count from response r
join quiz q on q.quiz_id=r.quiz_id
join conducts c on c.quiz_id=q.quiz_id
join professor p on p.emp_id=c.emp_id and p.prof_name='Viva Trew'
group by q.quiz_name;
```



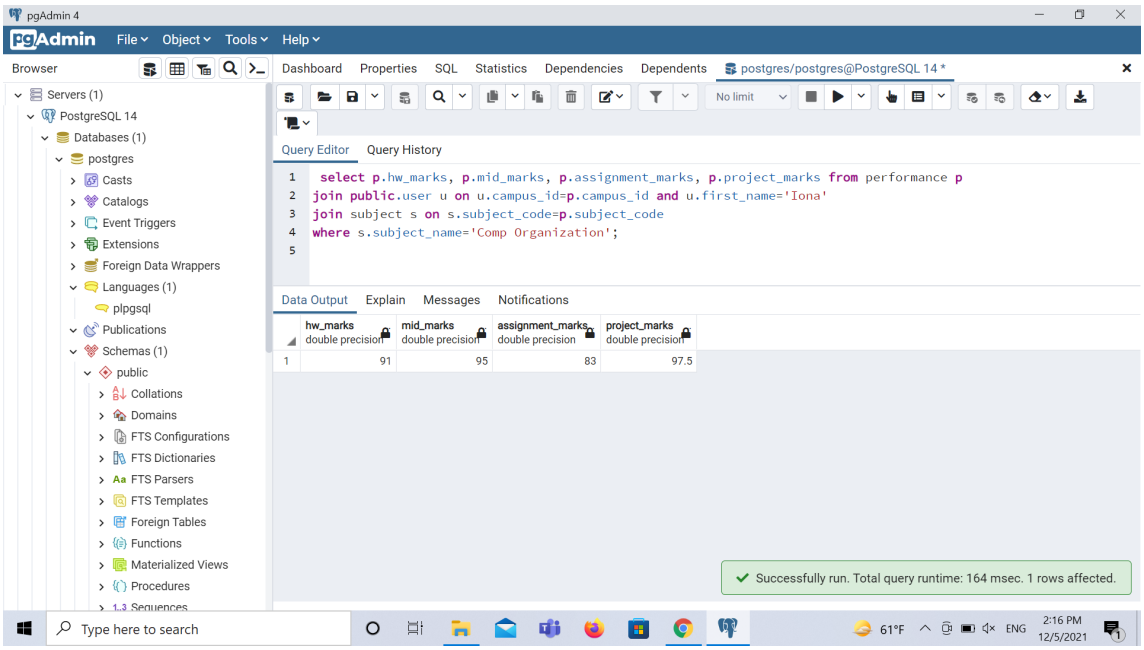
3. List the names of students who attempted a particular quiz ‘PL1Mids’ conducted by professor Viva Trew.

A:
```
select distinct(u.first_name, u.last_name) as name from public.user u
join response r on r.campus_id=u.campus_id
join quiz q on q.quiz_id=r.quiz_id and q.quiz_name='PL1Mids'
join conducts c on c.quiz_id=q.quiz_id
join professor p on p.emp_id=c.emp_id and p.prof_name='Viva Trew';
```



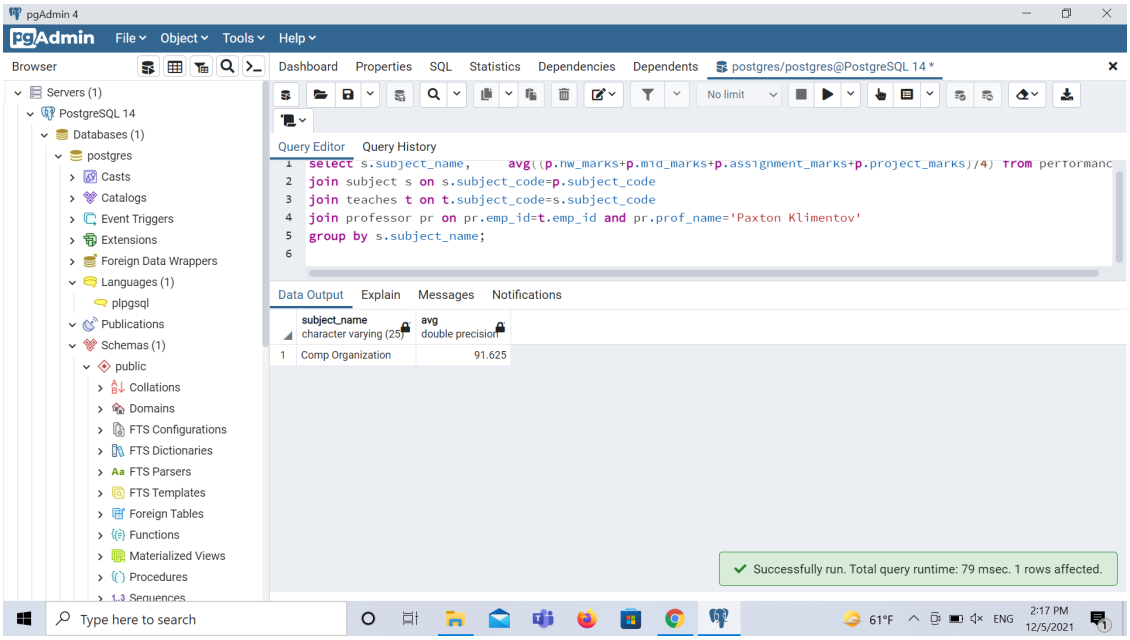
4. Display the overall performance of a student with the first name ‘Iona’ in the subject ‘Comp Organization.

A: select p.hw_marks, p.mid_marks, p.assignment_marks, p.project_marks from performance p
join public.user u on u.campus_id=p.campus_id and u.first_name='Iona'
join subject s on s.subject_code=p.subject_code
where s.subject_name='Comp Organization';



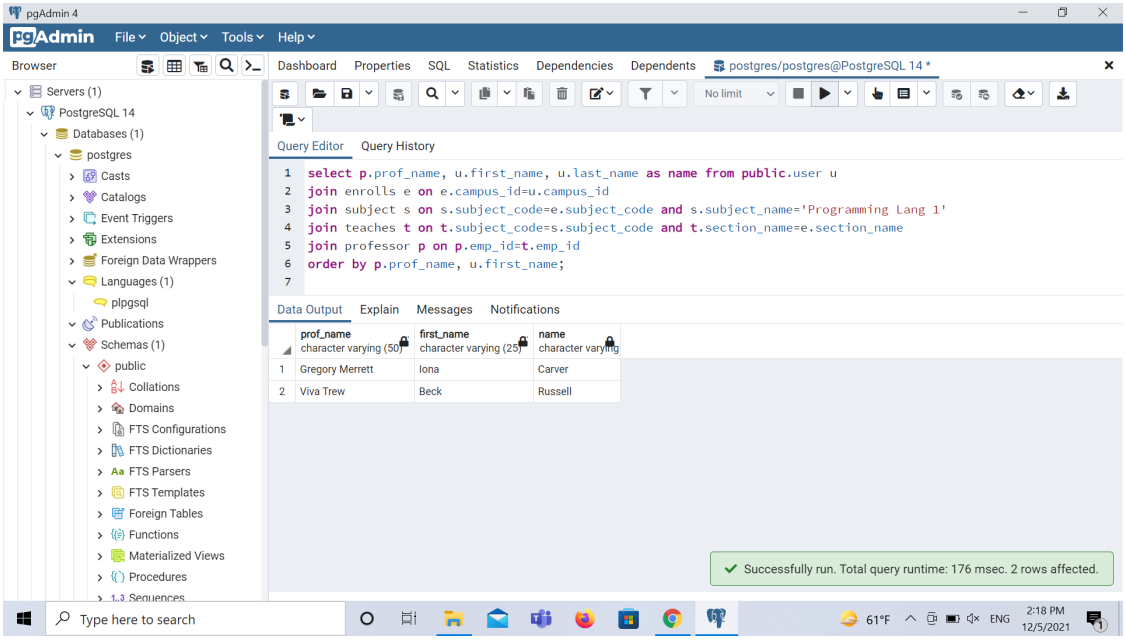
5. Display the class average performance and subject name of all the subjects taught by professor Paxton Klimentov.

A: select s.subject_name,
avg((p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4) from performance p
join subject s on s.subject_code=p.subject_code
join teaches t on t.subject_code=s.subject_code
join professor pr on pr.emp_id=t.emp_id and pr.prof_name='Paxton Klimentov'
group by s.subject_name;



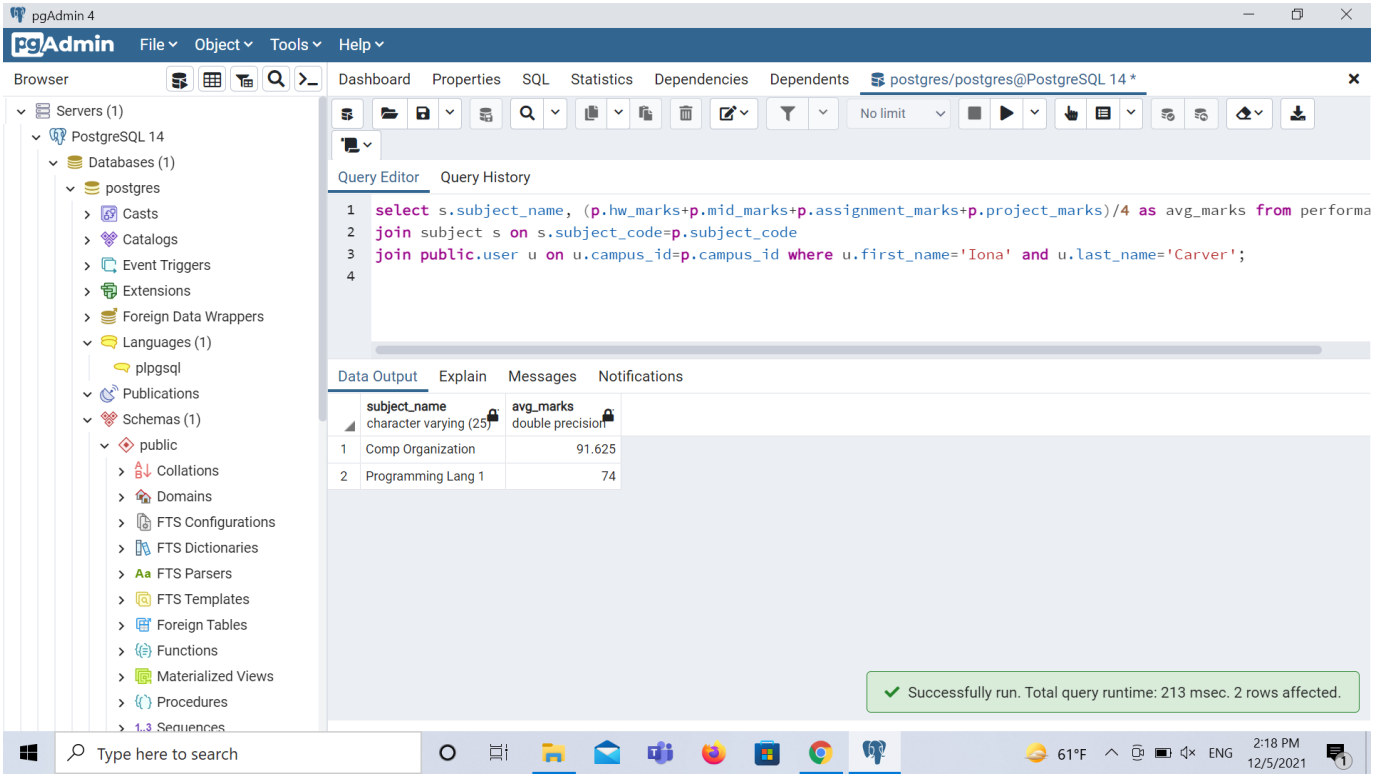
6. List the names of students enrolled in subject X, group by the professor teaching them, order by student names in ascending order.

A:
```
select p.prof_name, u.first_name, u.last_name as name from public.user u
join enrolls e on e.campus_id=u.campus_id
join subject s on s.subject_code=e.subject_code and s.subject_name='Programming Lang 1'
join teaches t on t.subject_code=s.subject_code and t.section_name=e.section_name
join professor p on p.emp_id=t.emp_id
order by p.prof_name, u.first_name;
```



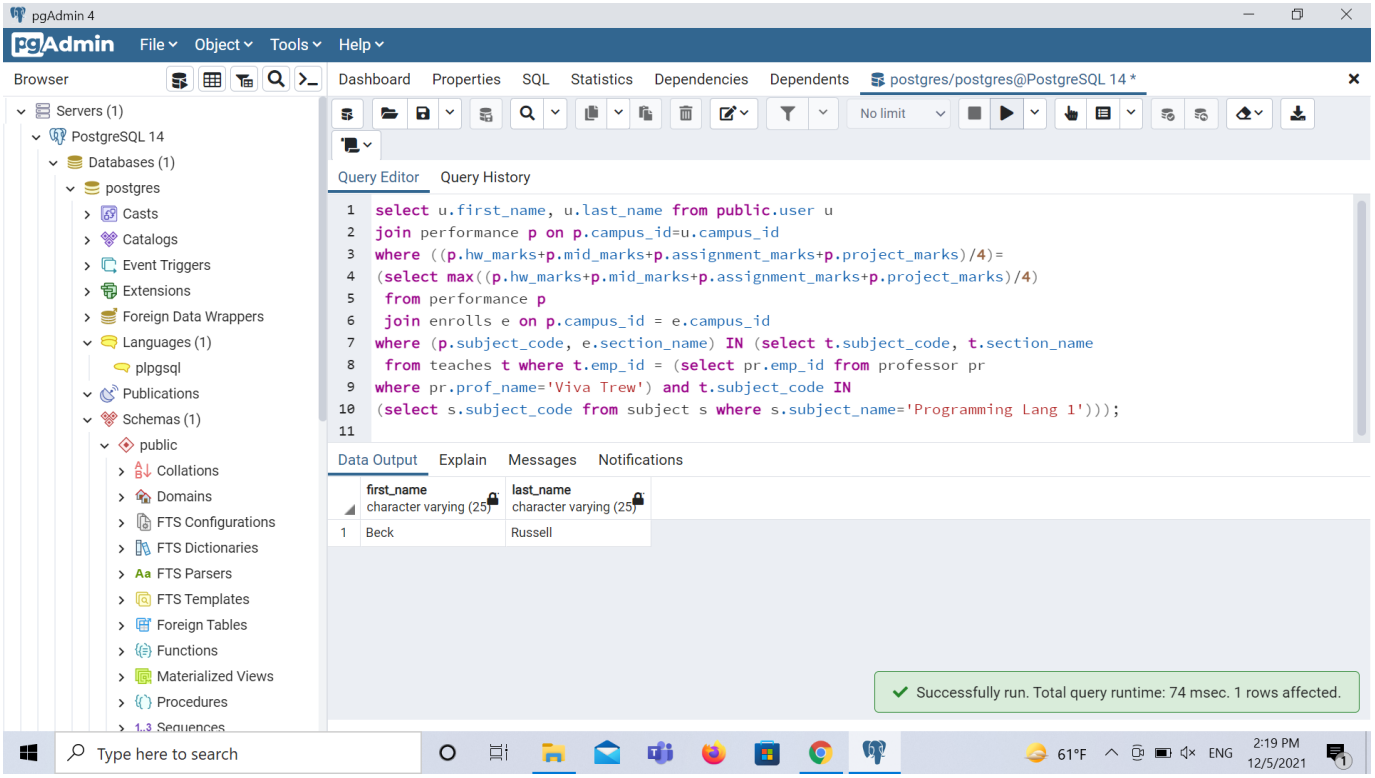
7. List the avg marks obtained by a student named ‘Iona Carver’ in each course enrolled by the student. (Display course name along with avg marks in that course)

A: select s.subject_name, (p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4 as avg_marks from performance p
join subject s on s.subject_code=p.subject_code
join public.user u on u.campus_id=p.campus_id where u.first_name='Iona' and u.last_name='Carver';



8. Display the name of students scoring max marks in subject Programming Lang 1 taught by professor Viva Trew.

A: select u.first_name, u.last_name from public.user u
join performance p on p.campus_id=u.campus_id
where ((p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4)=
(select max((p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4)
from performance p
join enrolls e on p.campus_id = e.campus_id
where (p.subject_code, e.section_name) IN (select t.subject_code, t.section_name
from teaches t where t.emp_id = (select pr.emp_id from professor pr
where pr.prof_name='Viva Trew') and t.subject_code IN
(select s.subject_code from subject s where s.subject_name='Programming Lang 1')));


pgAdmin 4
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@PostgreSQL 14 *
Servers (1)
PostgreSQL 14
Databases (1)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages (1)
plpgsql
Publications
Schemas (1)
public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Query Editor
Query History
select u.first_name, u.last_name from public.user u
join performance p on p.campus_id=u.campus_id
where ((p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4)=
(select max((p.hw_marks+p.mid_marks+p.assignment_marks+p.project_marks)/4)
from performance p
join enrolls e on p.campus_id = e.campus_id
where (p.subject_code, e.section_name) IN (select t.subject_code, t.section_name
from teaches t where t.emp_id = (select pr.emp_id from professor pr
where pr.prof_name='Viva Trew') and t.subject_code IN
(select s.subject_code from subject s where s.subject_name='Programming Lang 1')));
Data Output
Explain
Messages
Notifications
first_name
character varying (25)
last_name
character varying (25)
1
Beck
Russell
Successfully run. Total query runtime: 74 msec. 1 rows affected.
Type here to search
61°F
ENG
2:19 PM
12/5/2021

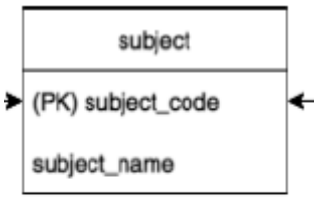
8. Time Logs

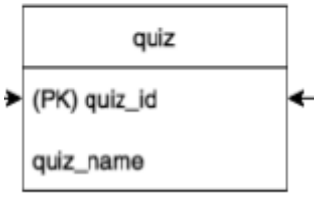
9. Appendix:

TA Comments:

Please find below for feedback comments:

1. How do you differentiate quizzes belonging to each subject or to each professor ? I don't see any relation between quiz entity and subject entity.





My comments: You can have subject_id's as foreign key (from the Subject table) . That will differentiate quizzes belonging to each subject.

As per the comment made by the TA, we have added the subject_code as a foreign key to the quiz entity to form the correlation between the subject in which the quiz is conducted.

2. Your performance entity also includes marks related to assignment and projects. How do you differentiate which quizzes are you going to consider for assignments or which quizzes are you going to consider for Midterm ? Or If you make an assumption that quizzes are considered only for homework and assignments, and project marks are directly uploaded by the professor. Then

Your ER diagram needs to have a relation between professor and performance allowing the professor to upload marks directly. If I were you , I would create another attribute in the quiz table, that would define the type of quiz which accepts values such as [Homework, Assignments, MidTerm, Final, Projects[If Required]].

As per our use case, it is upto the professor to consider any quiz as homework, assignment, project, or mids. There is no need to explicitly mention or have a separate attribute denoting it.

3. Since your performance entity also requires project marks, It would be good if you create another entity for projects and track all the project activities related to campus id in this table.

Project also pretty much comes under quiz, so it will not require a separate entity as that will lead to redundancy of tables.

4. Implementation-ready DB Model (+ which normalization level have you chosen, and why?)normalization level have you chosen, and why?)
    - I don’t see any reference to what kind of normalization is used in the database structure. It would be great if you can add some lines regarding normalization of your entities.

It has been explained above in the explanation of 3rd normal form and why we chose it.