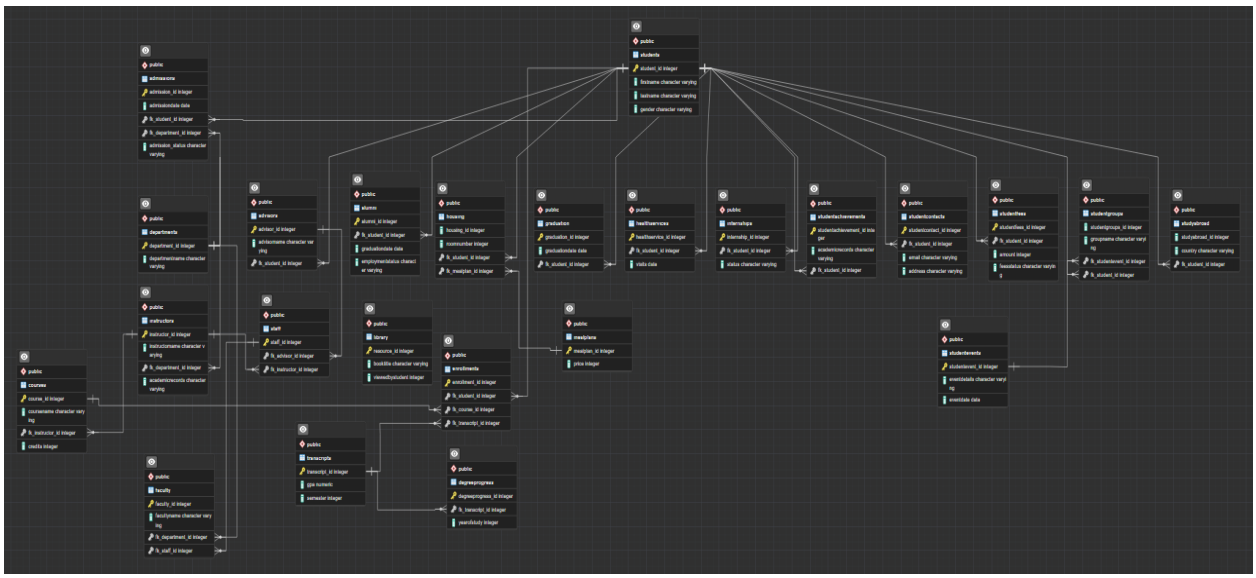

Question 1.

In this code we combine the students table(student_id) and the enrollments table(fk_student_id) by using a join clause. JOIN is a command clause that combines records from two or more tables in a database. It is a means of combining data in fields from two tables by using values common to each table.

```
--ex1
select student_id, firstName, lastName, enrollments.enrollment_id
from students
join enrollments on students.student_id = enrollments.fk_student_id
order by student_id, firstName, lastName, enrollments.enrollment_id;
```

The result returns list of all students and their enrollments in ascending order.

	student_id integer	firstname character varying	lastname character varying	enrollment_id integer
1	110	John	Doe	1
2	111	Jane	Smith	2
3	112	Alice	Johnson	3
4	113	Bob	Williams	4
5	114	Eva	Davis	5
6	115	Chris	Taylor	6
7	116	Sophie	Brown	7
8	117	Michael	Miller	8
9	118	Olivia	Jones	9
10	119	Daniel	Wilson	10
11	120	Emma	Thomas	11
12	121	Andrew	Wang	12
13	122	Mia	Lee	13
14	123	Madison	Adams	14
15	124	Lucas	Turner	15
16	125	Avery	Baker	16
17	126	Ethan	Evans	17
18	127	Sofia	Ward	18
19	128	Logan	Carter	19
20	129	Evelyn	Cooper	20
21	130	Daniel	Wilson	21
22	131	Aria	Harris	22
23	132	Henry	Bell	23
24	133	Amelia	Gray	24
25	134	Carter	Perez	25
26	135	Scarlett	Fisher	26
27	136	Sebastian	Morgan	27
28	137	Riley	Fletcher	28
29	138	Grace	Barnes	29
30	139	Leo	Butler	30

Question 2.

This code we select student_id, first name and last name columns from students table. Then combine students table(student_id) with enrollments table(fk_student_id) and enrollments table(fk_course_id) with courses table(couse_id) by using join clause.

```
--ex2
select students.student_id, firstName, lastName, courses.courseName
from students
join enrollments on students.student_id = enrollments.fk_student_id
join courses on enrollments.fk_course_id = courses.course_id;
```

The result returns a list of all students and the courses they are currently enrolled in, in ascending order.

	student_id integer	firstname character varying	lastname character varying	coursename character varying
1	110	John	Doe	Introduction to Computer Science
2	111	Jane	Smith	Calculus I
3	112	Alice	Johnson	English Composition
4	113	Bob	Williams	Physics I
5	114	Eva	Davis	History of Art
6	115	Chris	Taylor	Psychology 101
7	116	Sophie	Brown	Statistics for Business
8	117	Michael	Miller	Digital Marketing
9	118	Olivia	Jones	Organic Chemistry
10	119	Daniel	Wilson	World Literature
11	120	Emma	Thomas	Introduction to Engineering
12	121	Andrew	Wang	Linear Algebra
13	122	Mia	Lee	Environmental Science
14	123	Madison	Adams	Introduction to Sociology
15	124	Lucas	Turner	Computer Networks
16	125	Avery	Baker	Financial Accounting
17	126	Ethan	Evans	Spanish I
18	127	Sofia	Ward	Artificial Intelligence
19	128	Logan	Carter	Macroeconomics
20	129	Evelyn	Cooper	Microbiology
21	130	Daniel	Wilson	Philosophy of Ethics
22	131	Aria	Harris	Data Structures
23	132	Henry	Bell	Marketing Management
24	133	Amelia	Gray	French I
25	134	Carter	Perez	Introduction to Astronomy
26	135	Scarlett	Fisher	Human Resource Management
27	136	Sebastian	Morgan	Marketing
28	137	Riley	Fletcher	Ethics in Business
29	138	Grace	Barnes	Spanish II
30	139	Leo	Butler	Computer Graphics

Question 3.

This code select from student table student_id, firstName, lastName and by using the WHERE clause specifies criteria that field values must meet for the records that contain the values to be included in the query results. NOT IN operator is used to filter the result if the values that are mentioned as part of the IN operator is not satisfied.

```sql
--ex3
SELECT student_id, firstName, lastName
FROM students
WHERE student_id NOT IN (SELECT DISTINCT fk_student_id FROM advisors);
```

The code return the students who do not have assigned advisors.

	student_id [PK] integer	firstname character varying	lastname character varying
1	140	Layla	Cole

Question 4.

In this code we select student_id, firstName and lastName from table students. By using JOIN operations to combine data from multiple tables, students with enrollments, enrollments with transcript, trabscript with studentAchivement and also sets a limit so that the result does not exceed a certain amount.

```sql
--ex4
select students.student_id, students.firstName, students.lastName,
transcripts.gpa, studentAchievements.academicRecords
from students
join enrollments on students.student_id = enrollments.fk_student_id
join transcripts on enrollments.fk_transcript_id = transcripts.transcript_id
join studentAchievements on students.student_id = studentAchievements.fk_student_id
order by transcripts.gpa desc
limit 5;
```

The provided code retrieves specific information about students, including their student ID, first name, last name, GPA, and academic records.

	student_id integer	firstname character varying	lastname character varying	gpa numeric	academicrecords character varying
1	121	Andrew	Wang	3.9	Economics Research Fellowship Recipient
2	115	Chris	Taylor	3.9	Psychology Research Assistant
3	112	Alice	Johnson	3.8	Literary Magazine Publication
4	114	Eva	Davis	3.7	Art History Essay Contest Winner
5	118	Olivia	Jones	3.7	Chemistry Lab Assistant

Question 5.

This code select departmentName as major, and round our average students gpa to 2 zeroes after point to make it seems better and smaller and save this data as avgGPA. Combine all tables by using join clause.

```sql
--ex5
select departments.departmentName as major,
round(avg(transcripts.gpa),2) as avgGPA
from students
join enrollments on students.student_id = enrollments.fk_student_id
join transcripts on enrollments.fk_transcript_id = transcripts.transcript_id
join courses on enrollments.fk_course_id = courses.course_id
join instructors on courses.fk_instructor_id = instructors.instructor_id
join departments on instructors.fk_department_id = departments.department_id
group by major
order by avgGPA desc;
```

This code return two columns as major and avgGPA that contain data from departments and transcript and list all of them in descending order.

	major character varying	avggpa numeric
1	IDigital Marketing	3.80
2	English Language Teaching	3.80
3	Marketing	3.80
4	Business Administration	3.80
5	Economics	3.75
6	Mathematics Education	3.70
7	Environmental Science	3.70
8	Philosophy	3.68
9	Political Science	3.65
10	AProject Management	3.60
11	Computer Engineering	3.55
12	Computer Science	3.55
13	Engineering	3.50
14	Data Science	3.50
15	Multimedia Sciences	3.40
16	Business information systems	3.40
17	Iformation Systems	3.35
18	Management	3.30
19	Software Engineering	3.20
20	International Law	3.20

Question 6.

This code count number of courses that offered each department by using count() function. The COUNT() function returns the number of records returned by a select query. After that the tables join to each other, departments with instructors, instructors with courses. Then I use limit function to figure out only the highest data. I also group the query by department_id and departmentName.

```sql
--ex6
select departments.department_id, departments.departmentName,
count(courses.course_id) as numberOfCourses
from departments
join instructors on departments.department_id = instructors.fk_department_id
join courses on instructors.instructor_id = courses.fk_instructor_id
group by departments.department_id, departments.departmentName
order by numberOfCourses desc
limit 5;
```

This code return the highest number of courses from each departments and and figure out only several rows(limit 5).

	department_id [PK] integer	departmentname character varying	numberofcourses bigint
1	13	Philosophy	4
2	10	Economics	2
3	19	English Language Teaching	2
4	2	Iformation Systems	2
5	8	Political Science	2

Question 8.

This code select studentGroup_id and groupName from studentGroups, then join two tables, students with studentGroup. Then we calculate how many students have in each club by using count() function and save this data as members.

```sql
--ex8
select studentGroups.studentGroups_id, studentGroups.groupName,
count(students.student_id) as members
from studentGroups
join students on studentgroups.fk_student_id = students.student_id
group by studentGroups.studentGroups_id, studentGroups.groupName
order by members desc;
```

Then this code retrieve the number of members in each club and order by the highest number of members.

	studentgroups_id integer	groupname character varying	members bigint
1	611	Political Debaters	10
2	601	Programming Club	9
3	625	Pazzl Club	4
4	626	Mountain Girls	3
5	624	Music Lovers	3
6	636	Mountain Girls	1

Question 7.

This code select student_id, firstName and lastName and avisor_id from students table. Then combine students(student_id) table with advisors table(fk_student_id) using join function.

```
--ex7
select students.student_id, students.firstName,students.lastName, advisors.advisor_id
from students
join advisors on students.student_id = advisors.fk_student_id
```

Then it return all student_id and studentName with their advisors in ascending order.

	student_id integer	firstname character varying	lastname character varying	advisor_id integer
1	110	John	Doe	501
2	111	Jane	Smith	502
3	112	Alice	Johnson	503
4	113	Bob	Williams	504
5	114	Eva	Davis	505
6	115	Chris	Taylor	506
7	116	Sophie	Brown	507
8	117	Michael	Miller	508
9	118	Olivia	Jones	509
10	119	Daniel	Wilson	510
11	120	Emma	Thomas	511
12	121	Andrew	Wang	512
13	122	Mia	Lee	513
14	123	Madison	Adams	514
15	124	Lucas	Turner	515
16	125	Avery	Baker	516
17	126	Ethan	Evans	517
18	127	Sofia	Ward	518
19	128	Logan	Carter	519
20	129	Evelyn	Cooper	520
21	130	Daniel	Wilson	521
22	131	Aria	Harris	522
23	132	Henry	Bell	523
24	133	Amelia	Gray	524
25	134	Carter	Perez	525
26	135	Scarlett	Fisher	526
27	136	Sebastian	Morgan	527
28	137	Riley	Fletcher	528
29	138	Grace	Barnes	529
30	139	Leo	Butler	530

Question 9.

This code calculate the number of students in each room by using count() statement. In one room have 4 places for students, to find the occupancy percentage I use this formula Occupancy rate=(number of occupied rooms / total number of rooms) * 100. Then to combine two tables to each other, I use join statement.

```
--ex9
select (count(housing.fk_student_id) * 100 / (count(distinct housing.roomNumber) * 4)) as occupancyPercent
from housing
join students on housing.fk_student_id = students.student_id;
```

The code return the percentage of occupied numbers of rooms.

	occupancypercent bigint
1	83

Question 10.

This code calculate average cost of meals for different groups of students by selecting from degreeProgress where I have 1st, 2nd, 3rd year of study students. Then combine all tables, degreeProgress with transcript, transcript with enrollments, enrollments with students, students with housing, and last housing with mealPlan, where I have the price for each meal. I also round the average cost of meal to make it simply to understand.

```
--ex10
select degreeProgress.yearOfStudy,
round(avg(mealPlans.price),2) as averageCostOfMeal
from degreeProgress
join transcripts on degreeProgress.fk_transcript_id = transcripts.transcript_id
join enrollments on transcripts.transcript_id = enrollments.fk_transcript_id
join students on enrollments.fk_student_id = students.student_id
join housing on students.student_id = housing.fk_student_id
join mealPlans on housing.fk_mealPlan_id = mealPlans.mealPlan_id
group by degreeProgress.yearOfStudy
order by averageCostOfMeal desc;
```

At the end we see two columns that contain year of study of students and the average price for each year group.

	yearofstudy integer	averagecostofmeal numeric
1	3	2366.67
2	2	2210.00
3	1	2054.55

Question 11.

The following SQL code calculate all credit from each faculty by using sum() statement and by multiplying it to price we will take a course cost roe each faculty.

Also we join each table from enrollments to faculty, then necessarily group tthem by faculty(id, name).

```
--ex11
select faculty.faculty_id, faculty.facultyName,
sum(courses.credits) * count(*) as courseCost
from enrollments
join courses on enrollments.fk_course_id = courses.course_id
join instructors on courses.fk_instructor_id = instructors.instructor_id
join departments on instructors.fk_department_id = departments.department_id
join faculty on departments.department_id = faculty.fk_department_id
group by faculty.faculty_id, faculty.facultyName;
```

This code return the total tuition revenue generated by each academic department.

	faculty_id [PK] integer	facultyname character varying	coursecost bigint
1	1804	Faculty of Social Sciences	362970
2	1816	Faculty of Psychology	362970
3	1815	Faculty of Communication Studies	362970
4	1808	Faculty of Education	725940
5	1810	Faculty of Information Technology	846930
6	1809	Faculty of Fine Arts	362970
7	1802	Faculty of Science	846930
8	1803	Faculty of Engineering	483960
9	1813	Faculty of Economics	1572870
10	1814	Faculty of Agriculture	483960
11	1807	Faculty of Law	483960
12	1801	Faculty of Arts and Humanities	846930
13	1812	Faculty of Health Sciences	362970
14	1805	Faculty of Business Administration	362970
15	1819	Faculty of Science and Technology	725940
16	1818	Faculty of Environmental Studies	483960
17	1817	Faculty of Computer Science	846930

Question 12.

This code select all resources as available resources and then calculate the sum of the number viewed by student as checked out resource. Also if we have a null values we don't check and calculate them.

```
--ex12
select count(*) as availableResouces,
sum(viewedByStudent) as checkedOutResources
from library
where viewedByStudent is not null;
```

The result is show us the two columns with all resources and how many times students check this recourses at all.

	availableresouces bigint	checkedoutresources bigint
1	15	37

Question 13.

Question 14.

In this code we are finding the selected list of students academic records for each department. To finf it, I combine all tables, students with studentAchivments, students with enrollment, enrollments with courses, courses with instuctors and last instructors with departments.

```
--ex14
select studentAchievements.studentAchievement_id, studentAchievements.academicRecords, departments.departmentName
from studentAchievements
join students on studentAchievements.fk_student_id = students.student_id
join enrollments on students.student_id = enrollments.fk_student_id
join courses on enrollments.fk_course_id = courses.course_id
join instructors on courses.fk_instructor_id = instructors.instructor_id
join departments on instructors.fk_department_id = departments.department_id
group by departments.department_id, studentAchievements.studentAchievement_id,studentAchievements.academicRecords
order by  departments.departmentName;
```

We can see that result is in ascending order by Studentachivement_id and each academic record is grouped by departments.

	studentachievement_id integer	academicrecords character varying	departmentname character varying
1	1214	Biology Student of the Year	AProject Management
2	1207	Business Innovation Challenge Winner	Business information systems
3	1201	Deans List - Fall 2022	Computer Science
4	1204	Physics Research Symposium Presenter	Computer Science
5	1212	Economics Research Fellowship Recipient	Economics
6	1211	Engineering Design Competition Winner	Engineering
7	1203	Literary Magazine Publication	English Language Teaching
8	1209	Chemistry Lab Assistant	Environmental Science
9	1202	Mathematics Excellence Award	Iformation Systems
10	1213	Anthropology Fieldwork Scholar	Management
11	1215	Philosophy Essay Competition Finalist	Philosophy
12	1208	Sociology Outstanding Contribution Award	Philosophy
13	1206	Psychology Research Assistant	Philosophy
14	1210	Political Science Debate Champion	Political Science
15	1205	Art History Essay Contest Winner	Political Science

Question 15.

In this SQL code we need to find the percentage of participated students in internships. First I select all students that we have as total Students, then by using count() statement I find number of students that participated. Here I also have the column “status” with two data, “participated” and “not participated”. By using this strings and “case when” statement we can search students that we need for this query. Then to convert this number to percentage I use this formula = (the number of participated students * 100.0 / all students);

```
--ex15
select count(internships.fk_student_id) as totalStudents,
       count(distinct case when status = 'participated' then internships.fk_student_id end)
       as studentsWithInternship,
      (count(distinct case when status = 'participated' then internships.fk_student_id end)
       * round(100.0 / count(distinct internships.fk_student_id),2) ) as percentageWithInternships
from internships;
```

The result of this query is 3 columns, 1st is total students, 2nd number of students with internship and last percentage of the students with internships.

	totalstudents bigint	studentswithinternship bigint	percentagewithinternships numeric
1	30	18	59.94

Question 16.

Here I select column country where table name is StudyAbroad(fk_student_id) and combine it to students table(student_id) with join. Then I count each student as numberOfStudents. At the end group that data by country.

```
--ex16
select StudyAbroad.country,
count(distinct StudyAbroad.fk_student_id) as NOStudents
from StudyAbroad
join students on StudyAbroad.fk_student_id = students.student_id
group by StudyAbroad.country;
```

This code return the countries where students studied abroad and the number of students in each country.

	country character varying	nostudents bigint
1	America	3
2	Australia	4
3	Brazil	2
4	Canada	4
5	Columbia	1
6	India	2
7	Italy	2
8	Japan	2
9	Netherlands	1
10	Ozbekstan	2
11	Russia	1
12	South Korea	2
13	Sweden	2
14	Switzerland	1
15	Ukraine	4
16	United Kingdom	3

Question 17.

The code select studentEvent_id and studentDetails from studentEvents table. To find the events that will be in the future I use CURRENT_DATE function. CURRENT_DATE function returns the current date (the system date on the machine running PostgreSQL) as a value in the 'YYYY-MM-DD' format. By using ">=" in the code, it will takes only the greater or equal dates than current.

```
--ex17
select studentEvents.studentEvent_id, studentEvents.eventDetails
from studentEvents
where studentEvents.eventDate >= current_date
order by studentEvents.eventDate desc;
```

The following code return the list of upcoming events and their details in descending order.

	studentevent_id [PK] integer	eventdetails character varying
1	730	Computer Engineering Expo
2	729	Marketing Analytics Symposium
3	728	Clinical Psychology Panel Discussion
4	727	Art Conservation Workshop
5	726	Astrophysics Lecture Series
6	725	History Documentary Screening
7	724	Communication Skills Workshop
8	723	International Relations Conference
9	722	Organic Chemistry Seminar
10	721	Social Work Symposium
11	720	Marketing Trends Forum
12	719	Environmental Science Workshop
13	718	Music Concert
14	717	Fine Arts Exhibition

Question 18.

This code select departmentName from departments and calculate the number of alumni by using the count() function as employedAlumniCount, it will count employed alumni number for each department. Then combine all tables, departments with admissions, admissions with students, students with alumni. Then we are search by using “where” statement status of alumni, if it is ‘Employed’ we count them, if it is not we are skipping this alumni. At thr rnd of the code we group it by department name(major).

```
--ex18
select departments.departmentName as major,
count(alumni.alumni_id) as employedAlumniCount
from departments
join admissions on departments.department_id = admissions.fk_department_id
join students on admissions.fk_student_id = students.student_id
join alumni on students.student_id = alumni.fk_student_id
where alumni.employmentStatus = 'Employed'
group by major
order by employedAlumniCount;
```

The result of this code show us which faculties produce the most employable graduates in ascending order.

	major character varying	employedalumnicount bigint
1	Political Science	1
2	Psychology	1
3	Business information systems	1
4	Data Science	1
5	Economics	1
6	Engineering	1
7	English Language Teaching	1
8	Iformation Systems	1
9	International Law	1
10	Management	1
11	Marketing	1
12	Mathematics Education	1
13	Multimedia Sciences	1
14	Philosophy	1
15	Software Engineering	2
16	Computer Science	2
17	Business Administration	2

Question 20.

The provided SQL code retrieves enrollment statistics from the admissions table, specifically extracting the enrollment year from the admissionDate column. It then counts the distinct number of students enrolled in each year, grouping the results by the enrollment year and ordering them in ascending order. Counting distinct students ensures that each student is only counted once, regardless of the number of admissions they may have. This provides an accurate count of unique students enrolled in each year.

```sql
--ex20
select extract(year from admissions.admissionDate) as enrollmentYear,
count(distinct admissions.fk_student_id) as totalEnrollments
from admissions
group by enrollmentYear
order by enrollmentYear;
```

The results of the code is in ascending order analyze the historical enrollment data to identify trends in student enrollment over the past few years.

	enrollmentyear numeric	totalenrollments bigint
1	2019	8
2	2020	12
3	2021	10

Question 19.

This code select instructorName and their acedemicRecords from instructors table. Also use where statement to identify if academicRecords is “not null”, we will take the data. Grouping the result by instructorName, academicRecords.

```sql
--ex19
select instructors.instructorName, instructors.academicrecords
from instructors
where instructors.academicrecords is not null
group by instructors.instructorName, instructors.academicrecords;
```

This code return all faculty members who have expertise in specific research areas, based on their academic records.

	instructorname character varying	academicrecords character varying
1	Sebastian Morgan	Ph.D. in Clinical Psychology
2	Olivia Jones	Ph.D. in Chemistry
3	Emma Thomas	Ph.D. in Engineering
4	Leo Butler	Ph.D. in Data Science
5	Madison Adams	Ph.D. in Biology
6	Aria Harris	Ph.D. in Computer Engineering
7	Avery Baker	Ph.D. in Communications
8	Alice Johnson	Ph.D. in English Literature
9	Jane Doe	Ph.D. in Mathematics
10	Amelia Gray	Ph.D. in English Language Teaching
11	Henry Bell	Ph.D. in Mathematics Education
12	Ethan Evans	Ph.D. in History
13	Chris Taylor	Ph.D. in Psychology
14	Mia Lee	Ph.D. in Anthropology
15	Michael Miller	Ph.D. in Sociology
16	Carter Perez	Ph.D. in Astrophysics
17	Riley Fletcher	Ph.D. in Marketing
18	Daniel Wilson	Ph.D. in Political Science
19	Sofia Ward	Ph.D. in Environmental Science
20	John Smith	Ph.D. in Computer Science
21	Layla Cole	Ph.D. in International Relations
22	Bob Williams	Ph.D. in Physics
23	Evelyn Cooper	Ph.D. in Music
24	Grace Barnes	Ph.D. in Social Work
25	Andrew Wang	Ph.D. in Economics
26	Eva Davis	Ph.D. in Art History
27	Sophie Brown	Ph.D. in Business Administration
28	Lucas Turner	Ph.D. in Philosophy
29	Scarlett Fisher	Ph.D. in International Law
30	Logan Carter	Ph.D. in Fine Arts

Question 21.

In this code we are selected the student_id ,first and last names from students table and gpa from transcripts table. Then join the tables, students with enrollments, enrollments with transcripts.

To find the students that enrolling in advanced courses and pass all prerequisites must have more than 2,5 gpa score in their transcript. Then group them by the student_id ,first and last names and by gpa score.

```
--ex21
select students.student_id, students.firstName, lastName,
transcripts.gpa as passed
from students
join enrollments on students.student_id = enrollments.fk_student_id
join transcripts on enrollments.fk_transcript_id = transcripts.transcript_id
where transcripts.gpa >= 2.5
group by students.student_id, students.firstName, lastName, transcripts.gpa
order by transcripts.gpa desc;
```

The following query code return those students who are able to select next courses and pass previews courses in descending order.

	student_id integer	firstname character varying	lastname character varying	passed numeric
1	135	Scarlett	Fisher	3.9
2	132	Henry	Bell	3.8
3	139	Leo	Butler	3.8
4	110	John	Doe	3.5
5	133	Amelia	Gray	3.5
6	134	Carter	Perez	3.2
7	115	Chris	Taylor	2.9
8	121	Andrew	Wang	2.9
9	112	Alice	Johnson	2.8
10	125	Avery	Baker	2.8
11	114	Eva	Davis	2.7
12	118	Olivia	Jones	2.7
13	124	Lucas	Turner	2.7
14	136	Sebastian	Morgan	2.7
15	113	Bob	Williams	2.6
16	119	Daniel	Wilson	2.6
17	123	Madison	Adams	2.6
18	137	Riley	Fletcher	2.6
19	120	Emma	Thomas	2.5
20	138	Grace	Barnes	2.5

Question 22.

Provided code select student_id, studentName(first, last) from students table, then calculate the total fees by summing amount of the studentsFees. Join students table with studentFees and look if the studentFees status is ‘Pendeng’.The results are filtered to include only students with pending fees, and the grouping is based on the student's ID and name. The HAVING clause filters, ensuring that only students with pending fees are included in the final output. The grouped results after the GROUP BY operation. In this context, it ensures that only those students with pending fees (having a total pending fee amount greater than zero) are included in the final result set

```
--ex22
select students.student_id, students.firstName, students.lastName,
sum(studentFees.amount) as totalFees
from students
join studentFees on students.student_id = studentFees.fk_student_id
where studentFees.feesStatus = 'Pending'
group by students.student_id, students.firstName, students.lastName
having sum(studentFees.amount) > 0;
```

The result contain list of students with outstanding fees, including the total amount owed.

	student_id [PK] integer	firstname character varying	lastname character varying	totalfees bigint
1	111	Jane	Smith	1600
2	114	Eva	Davis	1900
3	117	Michael	Miller	2200
4	120	Emma	Thomas	2500
5	123	Madison	Adams	2800
6	126	Ethan	Evans	3100
7	129	Evelyn	Cooper	3400
8	131	Aria	Harris	3600
9	134	Carter	Perez	3900
10	137	Riley	Fletcher	4200

Question 23.

Question 24.

This code provides a students statistic distribution of gender, select the students gender column from students table, then count all students as studCount. Also I use round condition to simplify the percentage results. To find percentage for each gender I use this formula = (number of (male or female)students * 100.0 / number of all students); At the end group it all by gender.

```
--ex24
select students.gender,
count(*) as studCount,
round((count(*) * 100.0) / (select count(*)
                            from students), 2) as percentage
from students
group by gender;
```

The result of the code is contain 2 columns, genders, number of each gender and their percentage.

	gender character varying	studcount bigint	percentage numeric
1	Female	17	54.84
2	Male	14	45.16

Question 25.

Question 26.

This code slecet advisorName and advisor_id from advisors table and average of gpa from transcript table. By using join condition combine all tables, advisor with students, students with enrollments, enrollments with transcripts. Grouping the query columns by advisor_id and advisorName.

```
--ex26
select advisors.advisor_id, advisors.advisorName,
round(avg(transcripts.gpa),2) as avgGPA
from advisors
join students on advisors.fk_student_id = students.student_id
join enrollments on students.student_id = enrollments.fk_student_id
join transcripts on enrollments.fk_transcript_id = transcripts.transcript_id
group by advisors.advisor_id, advisors.advisorName
order by avgGpa desc;
```

Return the academic performance of students based on their faculty advisors in descending order.

	advisor_id [PK] integer	advisorname character varying	avggpa numeric
1	512	Prof. Lee	3.90
2	519	Dr. Cooper	3.90
3	526	Prof. Fletcher	3.90
4	506	Prof. Brown	3.90
5	516	Prof. Evans	3.80
6	530	Prof. Smith	3.80
7	503	Dr. Davis	3.80
8	523	Dr. Perez	3.80
9	505	Dr. Taylor	3.70
10	527	Dr. Barnes	3.70
11	509	Dr. Wilson	3.70
12	515	Dr. Baker	3.70
13	522	Prof. Gray	3.70
14	510	Prof. Thomas	3.60
15	504	Prof. Williams	3.60
16	514	Prof. Turner	3.60
17	528	Prof. Butler	3.60
18	521	Dr. Bell	3.60
19	529	Dr. Cole	3.50
20	501	Dr. Johnson	3.50
21	524	Prof. Fisher	3.50
22	518	Prof. Carter	3.50
23	511	Dr. Wang	3.50
24	507	Dr. Miller	3.40
25	520	Prof. Harris	3.40
26	513	Dr. Adams	3.30
27	517	Dr. Ward	3.20
28	525	Dr. Morgan	3.20
29	502	Prof. Smith	3.20
30	508	Prof. Jones	3.20

Question 27.

For this query we need to identify studentGroups that have members from different majors. Select studentGroup_id and studentGroupName, at first count all members then number of majors from departments. Join studentGroups with students, students with enrollments, enrollments with courses, courses with instructors and instructors with departments.

Grouping it all by studentGroup_id and groupName.

```
--ex27
select studentgroups.studentgroups_id, studentGroups.groupName,
count(distinct enrollments.fk_student_id) as NumOFMembers,
count(distinct departments.department_id) as NumOFMajors
from studentGroups
join students on studentGroups.fk_student_id = students.student_id
join enrollments on students.student_id = enrollments.fk_student_id
join courses on enrollments.fk_course_id = courses.course_id
join instructors on courses.fk_instructor_id = instructors.instructor_id
join departments on instructors.fk_department_id = departments.department_id
group by studentgroups.studentgroups_id, studentGroups.groupName
order by NumOFMajors desc;
```

Here we have group_id and names witch contain number of members and their numbers of major in each studentGroups. Result is in descending order.

	studentgroups_id integer	groupname character varying	numofmembers bigint	numofmajors bigint
1	611	Political Debaters	9	8
2	601	Programming Club	8	6
3	625	Pazzl Club	4	4
4	624	Music Lovers	3	3
5	626	Mountain Girls	3	3
6	636	Mountain Girls	1	1

Question 28.

Question 29.

This code select departmentName as major and calculate the average year to graduate from each major. By using year from age(date,date) statement we find the difference between two dates. The extract function in SQL is used to isolate a specific part of a date. Join admissions with graduation, graduation with departments, grouping the data by departmentName.

```
--ex29
select departments.departmentName as major,
round(avg(extract(year from age(graduationDate, admissionDate))),2) as avgYearToGraduate
from admissions
join graduation on admissions.fk_student_id = graduation.fk_student_id
join departments on admissions.fk_department_id = departments.department_id
group by departments.departmentName;
```

The code result give us the majors and their average year to graduate from each department.

	major character varying	avgyeartograduate numeric
1	International Law	3.00
2	Business information systems	3.00
3	Marketing	2.00
4	AProject Management	3.00
5	Business Administration	3.00
6	Computer Science	2.50
7	Environmental Science	3.00
8	Computer Engineering	3.00
9	Finance	2.00
10	Philosophy	3.00
11	IDigital Marketing	2.00
12	Economics	2.00
13	Political Science	3.00
14	English Language Teaching	3.00
15	Engineering	2.00
16	Management	3.00
17	Software Engineering	3.00
18	Iformation Systems	2.50
19	Data Science	2.00
20	Multimedia Sciences	3.00
21	Mathematics Education	3.00
22	Psychology	3.00

Question 30.