Name: Basil Ahamed

PostgreSQL Assignment

Assignment Description

In this assignment, you will work with PostgreSQL, a powerful open-source relational database management system. Your task involves creating 03 tables based on the provided sample data and then writing and executing queries to perform various database operations such as creating, reading, updating, and deleting data. Additionally, you will explore concepts like LIMIT and OFFSET, JOIN operations, GROUP BY, aggregation and LIKE.

Instructions:

Database Setup:

- Create a fresh database titled "**university_db**" or any other appropriate name.

Image:

SQL Shell (psql)

```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (16.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# create database university_db;
CREATE DATABASE
postgres=#
```

Table Creation:

Create a "**students**" table with the following fields:

- student_id (Primary Key): Integer, unique identifier for students.
- student_name: String, representing the student's name.
- age: Integer, indicating the student's age.
- email: String, storing the student's email address.
- frontend_mark: Integer, indicating the student's frontend assignment marks.
- backend_mark: Integer, indicating the student's backend assignment marks.
- status: String, storing the student's result status.

SQL Shell (psql)

```
university_db(# age int not null,
university_db(# email varchar2(100) not null unique,
university_db(# frontend_mark int not null,
university_db(# backend_mark int not null,
university_db(# status varchar2(50) not null);
ERROR:  type "varchar2" does not exist
LINE 3: student_name varchar2(100) not null unique,
                     ^
university_db=# create table if not EXISTS students(
university_db(# student_id SERIAL  not null unique primary key,
university_db(# student_name varchar(100) not null unique,
university_db(# age int not null,
university_db(# email varchar(100) not null unique,
university_db(# frontend_mark int not null,
university_db(# backend_mark int not null,
university_db(# status varchar(50) not null)
university_db-# ;
CREATE TABLE
university_db=#
```

Create a "**courses**" table with the following fields:

- course_id (Primary Key): Integer, unique identifier for courses.
- course_name: String, indicating the course's name.
- credits: Integer, signifying the number of credits for the course.

Create an "**enrollment**" table with the following fields:

- enrollment_id (Primary Key): Integer, unique identifier for enrollments.
- student_id (Foreign Key): Integer, referencing student_id in "Students" table.
- course_id (Foreign Key): Integer, referencing course_id in "Courses" table.

SQL Shell (psql)

```
university_db=# drop table if exists enrollment;
NOTICE:  table "enrollment" does not exist, skipping
DROP TABLE
university_db=# create table enrollment(
university_db(# enrollment_id serial not null unique primary key,
university_db(# student_id  int,
university_db(# course_id  int,
university_db(# foreign key(student_id) REFERENCES  students(student_id),
university_db(# foreign key(course_id) REFERENCES courses(course_id));
CREATE TABLE
university_db=# _
```

Sample Datas

- Insert the following sample data into the "**students**" table:Insert the following sample data into the "**courses**" table:

```
university_db=# insert into students(student_name,age,email,frontend_mark,backend_mark)
university_db-# values('basil ahamed', 22, 'basilahamed46@gmail.com',100,100),
university_db-# ('mohamed basil', 21, 'mohamedbasil46@gmail.com',90,100),
university_db-# ('Mohamed Farvez', 23, 'mohamedfarvez@gmail.com',30,50),
university_db-# ('mohamed mishal', 28, 'mohamedmishal@gmail.com',30,50),
university_db-# ('Sandeep', 30, 'Sandeep@gmail.com',80,90),
university_db-# ('siva', 40, 'siva@gmail.com',100,90);
INSERT 0 6
university_db=# select *
university_db-# from students;
 student_id |  student_name  | age |          email           | frontend_mark | backend_mark | status
------------+----------------+-----+--------------------------+---------------+--------------+--------
          1 | basil ahamed   |  22 | basilahamed46@gmail.com  |           100 |          100 |
          2 | mohamed basil  |  21 | mohamedbasil46@gmail.com |            90 |          100 |
          3 | Mohamed Farvez |  23 | mohamedfarvez@gmail.com  |            30 |           50 |
          4 | mohamed mishal |  28 | mohamedmishal@gmail.com  |            30 |           50 |
          5 | Sandeep        |  30 | Sandeep@gmail.com        |            80 |           90 |
          6 | siva           |  40 | siva@gmail.com           |           100 |           90 |
(6 rows)
```

```
university_db=# select *
university_db-# from courses;
 course_id | course_name | credits
-----------+-------------+---------
         1 | Next.js     |       3
         2 | React.js    |       4
         3 | Database    |       3
         4 | Prisma      |       3
(4 rows)
```

```
university_db=# insert into enrollment(enrollment_id,student_id,course_id)values(1,1,1),(2,1,2),(3,2,1),(4,3,2);
INSERT 0 4
university_db=# select *
university_db-# from enrollment;
 enrollment_id | student_id | course_id
---------------+------------+-----------
             1 |          1 |         1
             2 |          1 |         2
             3 |          2 |         1
             4 |          3 |         2
(4 rows)
```

Execute SQL queries to fulfill the ensuing tasks:

Query 1:

Insert a new student record with the following details:

- Name: YourName
- Age: YourAge
- Email: YourEmail
- Frontend-Mark: YourMark
- Backend-Mark: YourMark
- Status: NULL

```
LINE 2: values( ahamed basil , 22,  ahamedbasil46@gmail.com 50,50);
                                                            ^
university_db=# insert into students(student_name,age,email,frontend_mark,backend_mark)
university_db-# values('ahamed basil', 22, 'ahamedbasil46@gmail.com',50,50);
INSERT 0 1
university_db=#
university_db=# select *
university_db-# from students;
 student_id |  student_name  | age |          email           | frontend_mark | backend_mark | status
------------+----------------+-----+--------------------------+---------------+--------------+--------
          1 | basil ahamed   |  22 | basilahamed46@gmail.com  |           100 |          100 |
          2 | mohamed basil  |  21 | mohamedbasil46@gmail.com |            90 |          100 |
          3 | Mohamed Farvez |  23 | mohamedfarvez@gmail.com  |            30 |           50 |
          4 | mohamed mishal |  28 | mohamedmishal@gmail.com  |            30 |           50 |
          5 | Sandeep        |  30 | Sandeep@gmail.com        |            80 |           90 |
          6 | siva           |  40 | siva@gmail.com           |           100 |           90 |
          7 | ahamed basil   |  22 | ahamedbasil46@gmail.com  |            50 |           50 |
(7 rows)
```

Query 2:

Retrieve the names of all students who are enrolled in the course titled 'Next.js'.

```
university_db=# select student_name
university_db-# from students
university_db-# where student_id in (select student_id
university_db(# from enrollment
university_db(# where course_id = (select course_id
university_db(# from courses
university_db(# where course_name = 'Next.js'));
 student_name
---------------
 basil ahamed
 mohamed basil
(2 rows)
```

Query 3:

Update the status of the student with the highest total (frontend_mark + backend_mark) mark to 'Awarded'

```
university_db=# update students
university_db-# set status = 'Awarded'
university_db-# where student_id = (select student_id
university_db(# from students
university_db(# order by (frontend_mark+backend_mark) desc
university_db(# limit 1);
UPDATE 1
university_db=# select *
university_db-# from students;
 student_id |  student_name  | age |          email           | frontend_mark | backend_mark | status
------------+----------------+-----+--------------------------+---------------+--------------+---------
          2 | mohamed basil  |  21 | mohamedbasil46@gmail.com |            90 |          100 |
          3 | Mohamed Farvez |  23 | mohamedfarvez@gmail.com  |            30 |           50 |
          4 | mohamed mishal |  28 | mohamedmishal@gmail.com  |            30 |           50 |
          5 | Sandeep        |  30 | Sandeep@gmail.com        |            80 |           90 |
          6 | siva           |  40 | siva@gmail.com           |           100 |           90 |
          7 | ahamed basil   |  22 | ahamedbasil46@gmail.com  |            50 |           50 |
          1 | basil ahamed   |  22 | basilahamed46@gmail.com  |           100 |          100 | Awarded
(7 rows)
```

Query 4:

Delete all courses that have no students enrolled.

```
university_db=# DELETE FROM courses WHERE course_id NOT IN (SELECT DISTINCT course_id FROM enrollment);
DELETE 2
university_db=# select *
university_db-# from courses;
 course_id | course_name | credits
-----------+-------------+---------
         1 | Next.js     |       3
         2 | React.js    |       4
(2 rows)
```

Query 5:

Retrieve the names of students using a limit of 2, starting from the 3rd student.

```
university_db=#
university_db=# select student_name
university_db-# from students
university_db-# offset 3
university_db-# limit 2;
 student_name
--------------
 Sandeep
 siva
(2 rows)
```

Query 6:

Retrieve the course names and the number of students enrolled in each course.

```
university_db=# select c.course_name , count(enrollment_id)
university_db-# from courses c inner join
university_db-# enrollment e on c.course_id = e.course_id
university_db-# group by c.course_name;
 course_name | count
-------------+-------
 Next.js     |     2
 React.js    |     2
(2 rows)
```

Query 7:

Calculate and display the average age of all students.

```
university_db=# select avg(age)
university_db-# from students
university_db-# ;
         avg
--------------------
 26.5714285714285714
(1 row)
```

Query 8:

Retrieve the names of students whose email addresses contain 'example.com'.

```
             ^
university_db=# select student_name
university_db-# from students
university_db-# where email like '%gmail.com';
  student_name
----------------
 mohamed basil
 Mohamed Farvez
 mohamed mishal
 Sandeep
 siva
 ahamed basil
 basil ahamed
(7 rows)
```

Final_output_description

Prepare the SQL code for table creation, sample data insertion, and the seven queries in a text document or your preferred format. Include comments explaining each query's purpose and functionality. **Save your document as "PostgreSQL_Assignment.sql" or any other appropriate name.**

Based on the above table data explain the concept along with the example for below items

1. **Explain the primary key and foreign key concepts in PostgreSQL.**

primary key is used to find the uniquely from the table and foreign key is used to established the connection between the two tables

2. **What is the difference between the VARCHAR and CHAR data types?**

Varchar	**Char**
varchar datatype contain all characters	char datatype also contain all characters
varchare variable length allocation	Char datatype use fixed length allocation

3. **Explain the purpose of the WHERE clause in a SELECT statement.**

Where:

where clause is used to filter the records and it was execute row by row

select:

select clause is used to display the records according to the conduction

4. **What are the LIMIT and OFFSET clauses used for?**

Linit:

limit is used to give the limited rows according to number you enter

Offset:

offset is used for which row you want to start the record data

5. **How can you perform data modification using UPDATE statements?**

Syntax:

update table_name
set colomn_name = “values”
(or)
update table_name
set coloum_name = “values”
where conduction

6. **What is the significance of the JOIN operation, and how does it work in PostgreSQL?**

in postgresql we have joins

- inner join
- left outer join
- right outer join
- full outer join
- cross join
- self join

7. **Explain the GROUP BY clause and its role in aggregation operations.**

group by clause is used to group the records according to the column name and aggregation function for the example we can use the avg() function to aggregation function

8. **How can you calculate aggregate functions like COUNT, SUM, and AVG in PostgreSQL?**

syntax:

select count(*), avg(age), sum(age)
from students;

9. **What is the purpose of an index in PostgreSQL, and how does it optimize query performance?**

An index in PostgreSQL is a database object that improves the speed of data retrieval operations on a table at the cost of additional storage space and decreased performance on data modification operations.

10. **Explain the concept of a PostgreSQL view and how it differs from a table.**

A **view** in PostgreSQL is a saved SQL query that you can treat as if it were a table. It is a virtual table derived from one or more tables or other views. Unlike a table, a view does not store data physically.