2022-CS130FZ-January-Solutions

1

▼ **(a) What is a Database Management System (DBMS)? Describe any three functions of a DBMS. Provide any two advantages of using a DBMS.**

- *DBMS is a software system that enables users to define, create, maintain and control access to the database.*

Function:

- *Data definition (define the data types, structures, and constraints)*
- *Data manipulation (select, insert, update, delete date)*
- *Data control*

Advantages:

- *Data security: A DBMS provides various security measures to protect the data*
- *Data integrity: A DBMS ensures that the data is accurate and consistent, even if multiple users are accessing and manipulating the data at the same time.*

▼ **(b) Show the correct syntax for the use of a GROUP BY operator in SQL. Your answer should include the option of using the HAVING operator with the GROUP BY operator. Give an example in SQL that uses GROUP BY and HAVING operators. (You must give explanations of the example and a sample table.)**

- ***Syntax:***

```
SELECT column_list
FROM table_name
WHERE condition
GROUP BY column_name
HAVING condition
```

- ***Example:***

salesperson	region	month	sales
Bob	North	Jan	100
Bob	North	Feb	120
Bob	South	Jan	150
Alice	North	Jan	200
Alice	South	Feb	300
Alice	South	Mar	400
Alice	South	Apr	500

- *Suppose the sales more than 150 will be taken into account. To find the total sales for each salesperson in the North region, we use the following SQL query:*

```
SELECT salesperson, region, SUM(sales)
WHERE region = 'North'
GROUP BY salesperson, region
HAVING SUM(sales) > 150
```

▼ **(c) In terms of database security and vulnerabilities**

i. Explain what the term Injection Attack means.

ii. Briefly describe any three defenses that can be used against Injection Attacks.

- *It is a way that Attacker inject code into a database system to add new data or modify data stored in the database, consequently disrupting the system.*
- ***Defenses:***

- *Limit database permissions and segregate users*
- *Check syntax of input for validity*
- *Have length limits on input*

2

The following is the relational schema for a database of students' enrollment on modules:

`Student( StudentID , StudentGender, StudentFirstName, StudentLastName, StudentDOB, StudentEmail,StudentCourse);`

`Modules( ModuleID , ModuleTitle, ModuleCredits, ModuleSemester);`

`EnrolledOn( StudentID, ModuleID )`

The underlined attributes are the primary keys for each table. You can assume that all dates in the tables are stored in the format 'YYYY-MM-DD'.

▼ **(a) Write three appropriate CREATE statements for the Student, Modules, and EnrolledOn tables. You are free to choose the most appropriate data type for each attribute. You should ensure that Referential Integrity is enforced on this database.**

```
CREATE TABLE Student(
  StudentID TEXT PRIMARY KEY,
  StudentGender TEXT,
  StudentFirstName TEXT,
  StudentLastName TEXT,
  StudentDOB DATE,
  StudentEmail TEXT,
  StudentCourse TEXT
)
```

```
CREATE TABLE Modules(
  ModuleID TEXT PRIMARY KEY,
  ModuleTitle TEXT,
  ModuleCredits SERIAL,
  ModuleSemester TEXT
)
```

```
CREATE TABLE EnrolledOn(
  StudentID TEXT REFERENCES Student(StudentID) ON UPDATE CASCADE ON DELETE CASCADE,
  ModuleID TEXT REFERENCES Modules(ModuleID) ON UPDATE CASCADE ON DELETE CASCADE,
  PRIMARY KEY(StudentID, ModuleID)
)
```

▼ **(b) Write three INSERT statements for the Student, Modules, and EnrolledOn tables.**
For Student table: `(3001, Male, John, Smith, 2001-03-06, johns@google.com, BA);`
For Modules table: `(CS130, Database, 5, Semester 1);`
For EnrolledOn table: `(3001, CS130)`

```
INSERT INTO Student(StudentID, StudentGender, StudentFirstName, StudentLastName, StudentDOB, StudentEmail,StudentCourse) VALUES ('
INSERT INTO Modules(ModuleID, ModuleTitle, ModuleCredits, ModuleSemester) VALUES ('CS130', 'Database', 5, 'Semester 1');
INSERT INTO EnrolledOn(StudentID, ModuleID) VALUES ('3001', 'CS130');
```

▼ **(c) Using the above database, formulate the following queries in SQL**

▼ **i. Select all the students that were born in 2001.**

```
SELECT * FROM Student WHERE date_part('YEAR',StudentDOB)='2001';
```

▼ **ii. Write an appropriate JOIN query to list every student (name, gender, email) who is enrolled on the module with ID 'CS123' for any semester.**

```
SELECT StudentFirstName,StudentLastName,StudentEmail,StudentGender FROM Student AS t1
JOIN EnrolledOn AS t2 ON t1.StudentID = t2.StudentID
JOIN Modules AS t3 ON t3.ModuleID = t2.ModuleID
WHERE ModuleTitle = 'CS123'
```

▼ **iii. Write an appropriate SQL query to list all students whose first names ended with 'ry' (case insensitive).**

```
SELECT * FROM Student
WHERE StudentFirstName ~ '.*ry$'
```

▼ **iv. Write a query to display every enrollment of all the students that are enrolled on modules which have module credits of between 10 and 15 credits inclusive and where the student is not an undergraduate. Undergraduates have StudentCourse with BSc or BA. All other student courses are Postgraduat.**

```
SELECT * FROM EnrolledOn AS t1
JOIN Student AS t2 ON t1.StudentID = t2.StudentID
JOIN Modules AS t3 ON t1.ModuleID = t3.ModuleID
WHERE ModuleCredits BETWEEN 10 AND 15
AND StudentCourse NOT IN ('BSC','BA')
```

▼ **v. Write a query which will delete the module with module ID CS2800 from the database. You are asked to write a number of select queries to find the total number of rows affected. It is important that you remember that there are CASCADING DELETES in operation to support REFERENTIAL INTEGRITY in the database.**

```
-- the number of row in EnrolledOn before deletion
SELECT * FROM EnrolledOn WHERE ModuleID = 'CS2800'

-- delete the relevant data
DELETE ModuleID FROM Module WHERE ModuleID = 'CS2800'

-- the number of row in EnrolledOn after deletion
SELECT * FROM EnrolledOn WHERE ModuleID = 'CS2800'
```

▼ **(a) Define normalization and give two reasons why it can be useful to normalize tables.**

- *Normalization is the process of organizing a database in a way that reduces repetition of data and ensures that data is stored in a logical manner.*

Reason

- *Increase flexibility to manipulate data*
- *Avoid frequent restructuring of tables and reduce disk space*

▼ **(b) For the following un-normalized Table, reorganize it into a table/tables in First Normal Form.**

EmpID	Name	Dept Code	Dept Name	Proj 1	Time Proj1	Proj 2	Time Proj2	Proj 3	Time Proj3
EN1-26	Sean Breen	TW	Technical Writing	30-T3	25%	30-TC	40%	31-T3	30%
EN1-33	Amy Guyu	TW	Technical Writing	30-T3	50%	30-TC	35%	31-T3	60%
EN1-36	Liz Roslyn	AC	Accounting	35-TC	90%				

EmpID	FirstName	LastName	DeptCode	DeptName	ProjectName	ProjectTime

▼ **(c) For the following Table in 1st Normal Form, reorganize it into a table/tables in 2nd and 3rd Normal Forms.**

Rep ID	Rep First Name	Rep Last Name	Client ID	Client	Time With Client
TS-89	Gilroy	Gladstone	978	US Corp	14 hrs
TS-89	Gilroy	Gladstone	665	Taggarts	26 hrs
TS-89	Gilroy	Gladstone	782	Kilroy Inc.	9 hrs
RK-56	Mary	Mayhem	221	Italiana	67 hrs
RK-56	Mary	Mayhem	982	Linkers	2 hrs

3nd Normal Form:

RepID	RepFirstName	RepLastName

ClientID	Client

RepID	ClientID	TimeWithClient