QUESTION BANK

Faculty of Applied Science
Department of: Information Technology
Course: Database Systems II

Q. Choose the correct answer.

1. Which one is NOT a spatial data type in MySQL?

A. POINT	**B. CHAR**
C. GEOMETRY	D. LINESTRING

2. ………………..... is a DCL (Data Control Language) statement in SQL.

A. CREATE	B. SELECT
C. GRANT	D. INSERT

3. What is the output of this SQL statement? → **select concat**('IT' , 'Department');

A. IT Department	B. IT , Department
C. ITDepartment	D. None of them

Q. Fill in the blanks with the correct word(s).

A. ……User-Defined Functions…….. are one of the most useful features in MySQL, allowing users to extend MySQL functionality by creating custom functions.

B. The …char_length()… function returns the length of a given string in characters (number of characters).

C. The …default…. constraint is used to provide a default value to a column.

Q. Write the required SQL query/command for each part.

Student Table

stuID	stuName	deptName	credits
1	Saeed	IT	55
2	Kawa	IT	36
3	Lana	English	60

A. Write an SQL command to create **student** table. Consider below points during creation of table.

i. **stuID** is primary key.

ii. Default value for **deptName** column is ***'IT'***.

```
create table student
(stuID int,
stuName varchar(70),
deptName varchar(50) default 'IT',
credits int,
primary key (stuID));
```

B. Write an SQL command to enter data inside the student table as shown above.

```
insert into student values (1, 'Saeed', 'IT', 55),
                           (2, 'Kawa', 'IT', 36),
                           (3, 'Lana', 'English', 60);
```

C. Write an SQL command to drop **stuName** column from **student** table.

```
alter table student
drop column stuName;
```

Q. Write the required SQL query/command for each part.

Student Table

stuID	stuName	deptName	credits
1	Saeed	IT	55
2	Kawa	IT	36
3	Lana	English	60

A. Write an SQL command to create **student** table. Consider below points during creation of table.

i. **stuID** is primary key and has auto_increment constraint.

ii. Default value for **deptName** column is ***'General Education'***.

iii. **stuName** column has **not null** constraint.

```
create table student
(stuID int auto_increment,
stuName varchar(70) not null,
deptName varchar(50) default 'General Education',
credits int,
primary key (stuID));
```

B. Write an SQL statement to add/drop the following constraints to/from the table after table's creation.

i. Add **UNIQUE** constraint to **stuName**.

ii. Remove **DEFAULT** constraint from **deptName** column.

```
alter table student
add unique(stuName);


alter table student
alter deptName drop default;
```

Q. By having **Item** table, Write the required SQL command to create a function named **TotalQuantity** to calculate and return the total quantity of all items in the **item** table. (Use a local variable in the function.)

```
DELIMITER $$
CREATE  FUNCTION  TotalQuantity ( )
RETURNS  INT  DETERMINISTIC
BEGIN
          DECLARE  total  INT;
          SELECT  sum(StockQuantity)  INTO  total  FROM  Item;
          RETURN  total;
END $$
DELIMITER ;
```

Item

ItemID	ItemName	Price	StockQuantity
1	Calculator	20	31
2	Punching Machine	15	25
3	Scissors	5	100

Q. By having **Item** table, Write the required SQL command to create a function named **getPrice** to take **ID** an item and return its **price**.

Item

ItemID	ItemName	Price	StockQuantity
1	Calculator	20	31
2	Punching Machine	15	25
3	Scissors	5	100

```
DELIMITER $$
CREATE  FUNCTION  getPrice ( ID  int)
RETURNS  INT  DETERMINISTIC
BEGIN
        DECLARE  PriceVar  INT;
        SELECT  price  INTO  PriceVar  FROM  Item WHERE  ItemID = ID;
        RETURN  priceVar;
END $$
DELIMITER ;
```

Q. By having the following two tables, Write the required SQL command for each part.

Product

PID	PName	Price
1	Laptop	2000
2	External HDD	200
3	Keyboard	40

Orders

orderID	productID	Quantity
1	3	25
2	3	100
3	2	10

A. Write an SQL command to create a view named **lowPrice** to find the **ID** and **name** of products priced less than $300.

```
CREATE VIEW lowPrice AS
        SELECT PID, PName
        FROM Product
        WHERE Price < 300;
```

B. Write an SQL command to create a view named **highQuantity** to find the **orderID** and **name** of products that their ordered quantity is greater than 50.

```
CREATE VIEW highQuantity AS
        SELECT orderID, PName
        FROM Product, Order
        WHERE Product.PID = Order.productID AND Quantity > 50;
```

Q. Write the required SQL query/command for each part.

Product Table

PID	PName	Price
1	Laptop	2000
2	External HDD	200
3	Keyboard	40

A. Write an SQL command to create **product** table. Consider below points during creation of table.

i. **PID** is primary key.

ii. **Price** column cannot take **null** value.

```
create table product
(PID int,
PName varchar(100),
Price int not null,
primary key (PID));
```

B. Write an SQL query to show the following output according to the given conditions:

Conditions

Price ≥ 900	→	PriceLevel : '**Expensive**'
200 ≤ Price < 900	→	PriceLevel : '**Reasonable**'
0 ≤ Price < 200	→	PriceLevel : '**Cheap**'
Price < 0	→	PriceLevel : '**Wrong Price**'

Output

PName	Price	PriceLevel
Laptop	2000	**Expensive**
External HDD	200	**Reasonable**
Keyboard	40	**Cheap** [11]

```
SELECT Pname, Price,
       CASE
             WHEN Price >= 900 THEN 'Expensive'
             WHEN Price >=200 AND Price < 900 THEN 'Reasonable'
             WHEN Price >= 0 AND Price < 200 THEN 'Cheap'
             ELSE 'Wrong Price'
       END AS PriceLevel
FROM Product;
```

Q. By having the **Book** table, write required SQL codes for each part.

<u>**ISBN**</u>	**bookName**	**Price**
113	Intro to Python	45
114	Machine Learning	30
115	Computer Networks	75

A. Create a procedure that takes ISBN of a book and updates the price of only that book by adding 10 dollars.

B. Call the procedure by passing ISBN = 114 to the procedure.

```
DELIMITER //
CREATE PROCEDURE updatePrice(IN ISBN int )
BEGIN
            UPDATE Book SET Price = Price + 10 WHERE Book.ISBN = ISBN;
END//
DELIMITER ;
```

```
CALL updatePrice(114);
```

```
SELECT * FROM Book;
```

Q. Suppose there is a database named **University** and there is a table named **Student** in the database. Accordingly, write the required SQL codes for each part:

A. Create a user named ‘HoD’, with a password.
B. Give the ‘HoD’ user all permissions on **Student** table in the database.
C. Then, take back DELETE and INSERT privileges on Student table from ‘HoD’ user.

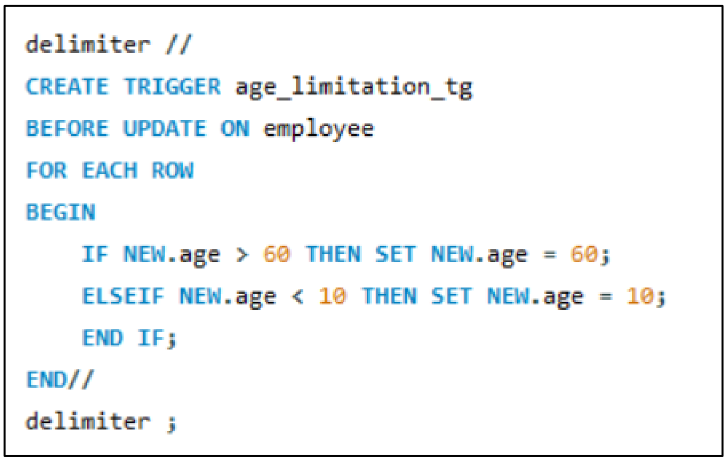
```
create user 'HoD'@'localhost' identified by 'admin1234';

grant all on university.Student to 'HoD'@'localhost';

revoke DELETE, INSERT on university.Student from 'HoD'@'localhost';
```

Q. Write SQL code to create a trigger that is activated when any update is going to happen on **Employee** table. The trigger considers that age of employees must be between 10 and 60, and will do the following:

- If the updated age becomes more than 60, the trigger sets the Age to 60,
- If the updated age becomes less than 10, the trigger sets the Age to 10.

Employee Table

EID	Ename	Age
1	Hasan	44
2	Lana	36



```
delimiter //
CREATE TRIGGER age_limitation_tg
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
    IF NEW.age > 60 THEN SET NEW.age = 60;
    ELSEIF NEW.age < 10 THEN SET NEW.age = 10;
    END IF;
END//
delimiter ;
```