Machine test - Django/ Laravel(Database)

Instructions

- **Six questions.**
- **Maximum 10 points for each question.**
- **Points will be awarded for correctness and for the simplicity and elegance of your code.**
- **Avoid using temp tables or variables in your answers, except for the variables needed in question 5.**

1. Using the **stock** table and data given below, create a view called **stockmonthly** that shows the monthly total of the qty column for each item and month. Also include in your query a column called **rnk** that ranks the quantities in descending order for each item. Your result should look like this:

```
item year month qty rnk
----------- ----------- ----------- ----------- ----
1001 2020 1 890 1
1001 2020 2 832 2
1001 2019 11 340 3
1002 2019 12 603 1
1002 2020 3 172 2
1002 2020 4 150 3
1002 2019 11 120 4


CREATE TABLE stock
(item INTEGER NOT NULL
,dt DATE NOT NULL
,qty INTEGER NOT NULL
,PRIMARY KEY (item, dt));

INSERT INTO stock (item, dt, qty)
VALUES
  (1001,'2019-11-11',340)
,(1002,'2019-11-23',120)
,(1002,'2019-12-20',201)
,(1002,'2019-12-16',402)
,(1001,'2020-01-19',890)
,(1001,'2020-02-12',340)
,(1001,'2020-02-27',101)
,(1001,'2020-02-29',391)
,(1002,'2020-03-11',172)
,(1002,'2020-04-21',150);
```

2. Using the same **stock** table as in the previous question, write a query to retrieve the date and the qty for the row(s) with the *second highest* qty in the entire table. Your query should work without needing any modification to give the second highest result for *any* set of data in the table, not just for the data given. Use the stock table in your query; you don't need to use the view you created for the previous question. The result should be:

```
dt qty
---------- -----------
2019-12-16 402
```

3. Using the table called **journey** and the sample data given below, write a query to display the following result. You can assume that the **descr** column always contains a hyphen between the start and end point and that the **distance** column always ends with either "mi" or "km", giving the distance in either miles or kilometers. In your answer, multiply the miles by 1.609 to convert the miles to km for the distance_km column.

```
journey_start journey_end distance_km
--------------- -------------- ------------
London Paris 342.000
New York Los Angeles 3934.005
New York Washington 326.627
Paris Rome 1106.000

CREATE TABLE journey (descr VARCHAR(100) PRIMARY KEY, distance VARCHAR(7) NOT
NULL);

INSERT INTO journey (descr, distance)
VALUES
  ('London-Paris','342km')
,('New York-Los Angeles','2445mi')
,('New York-Washington','203mi')
,('Paris-Rome',' 1106km');
```

4. Using the **programmer** and **skill** tables and the sample data given below, write a query to return the count of the number of people at each location who have the skill **C++** with a rating of **7 or more**. The result should be:

```
location num
------------------- -----------
Dublin 0
London 2
New York 1


CREATE TABLE programmer (employee_name VARCHAR(20) NOT NULL PRIMARY KEY, location
VARCHAR(20)  NOT NULL);
```

```
CREATE TABLE skill (employee_name VARCHAR(20) NOT NULL, language VARCHAR(20) NOT
NULL, rating  INT NOT NULL, PRIMARY KEY (employee_name, language));


INSERT INTO programmer (employee_name, location)s
VALUES
  ('Alan','London')
,('Belinda','New York')
,('Chuck','New York')
,('Diana','London')
,('Ed','New York')
,('Frank','London')
,('Gail','Dublin');

INSERT INTO skill (employee_name, language, rating)
VALUES
  ('Alan','C++',9)
,('Alan','Java',6)
,('Belinda','Java',10)
,('Belinda','Python',8)
,('Chuck','C++',7)
,('Diana','Python',10)
,('Diana','Java',5)
,('Ed','Java',8)
,('Frank','C++',9)
,('Frank','Python',6)
,('Gail','Java',6);
```

5. Using the same tables and data as in the previous question, write a stored procedure that retrieves the names and ratings of people for a given location and language. The location and language should be specified as parameters when you call your procedure.
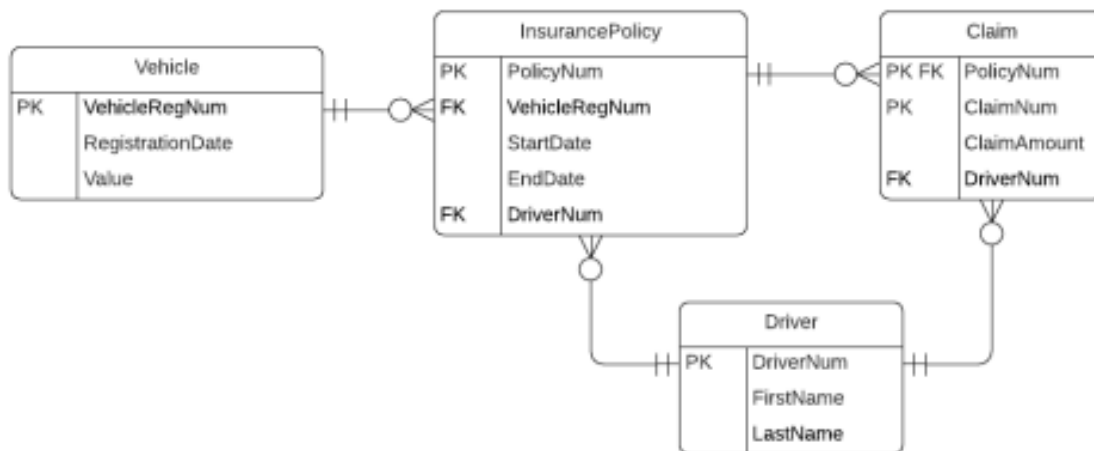
   When you have written your procedure you should be able to test it by executing the following command:

   ```
   EXEC getskill @language = 'Python', @location = 'London';
   ```

   Expected result:

   ```
   employee_name rating
   -------------------- -----------
   Diana 10
   Frank 6
   ```

6. Create four tables as shown in the following ER diagram. You must include the primary keys, foreign keys and some suitable data types. All columns should be non-nullable.

Vehicle

PK	VehicleRegNum
	RegistrationDate
	Value

InsurancePolicy

PK	PolicyNum
FK	VehicleRegNum
	StartDate
	EndDate
FK	DriverNum

Claim

PK FK	PolicyNum
PK	ClaimNum
	ClaimAmount
FK	DriverNum

Driver

PK	DriverNum
	FirstName
	LastName

THE END