Quiz 2 Mock Questions

1] Consider the following two tables in sqlite database in which 'TableA' is having primary key as 'Id' and 'TableB' is having primary key as 'RefId', and foreign key as 'Id' which refers to the primary key in TableA.

TableA:

Id	Name	City
2	Rahul	Chennai
3	Amrutha	Delhi
4	Ram	Chennai
5	Jyoti	Chennai

TableB:

RefId	Name	Location	Id
1	Rahul	Chennai	2
2	Amrutha	Delhi	3
3	Jyoti	Chennai	5

Which of the following queries will return records from the tables such that Name = Amrutha or City = Chennai ordered by the 'Name' in ascending order ? [MCQ: 2 points]

1.
   ```
   select * from TableA
   INNER Join TableB
   on TableA.Id=TableB.Id
   where TableA.City='Chennai' OR TableA.Name='Amrutha'
   order by 2 ASC;
   ```

2.
   ```
   select * from TableA
   INNER Join TableB
   on TableA.Id=TableB.Id
   where TableA.Name='Amrutha'
   order by 2 ASC;
   ```

3.
   ```
   select * from TableA
   INNER Join TableB
   on TableA.Id=TableB.Id
   where TableA.City='Chennai'
   order by 2 ASC;
   ```
4. None of the above

Answer: Option 1.

Solution: The query that will fetch the records from Tables where city="Chennai" or Name="Amrutha" and will sort the result in ascending order of a second column is:

```
select * from TableA
INNER Join TableB
```

```
on TableA.Id=TableB.Id
where TableA.City='Chennai' OR TableA.Name='Amrutha'
order by 2 ASC;
```

The second line performs inner join on table B such that `TableA.Id=TableB.Id` and filters the records on the condition `TableA.City='Chennai' OR TableA.Name='Amrutha'` . The last line of the query will order the records according to the second column in ascending order. It is an alternative when we do not want to use the actual `column name` .

In option 2 and 3, the where clause is not correct according to the constraints specified in the question.

2] A table in sqlite database is given below. What will be the output of following sql query? [MCQ: 2 points]

TableA:

Id	Name	City
2	Rahul	Chennai
3	Amrutha	Bangalore
4	Ram	Chennai
5	Tejas	Chennai

```
Select * from TableA group by City HAVING count(Id)>=3 order by City desc;
```

```
1. Id | Name   |  City
   ---|-------|--------
    3 |Amrutha|  Delhi
    2 |Rahul  | Chennai

2. Id | Name   |  City
   ---|-------|--------
    2 |Rahul  | Chennai
    3 |Amrutha|  Delhi

3. Id | Name |  City
   ---|------|--------
    2 |Rahul | Chennai

4. None of the above
```

Answer: Option 3.

Solution: The above SQL query will select one record with all column data from table A and grouping it by 'City' from table A and filter is applied using having clause as count(Id)>=3 and also the result is in descending order of the 'City' column.

3] Consider the table 'Worker' in sqlite database.

Worker_ID	First_name	Last_name	Salary	Department
101	Arvind	Mehra	100000	Accounts

Worker_ID	First_name	Last_name	Salary	Department
102	Ravindra	Rawat	120000	Admin
103	Manoj	Swami	80000	Admin
104	Khushi	Yadav	75000	Accounts
105	Meera	Singh	150000	HR
106	Abhinav	Gupta	120000	HR
107	Ramesh	Mishra	80000	Admin

The correct SQL query that will print details of the Workers whose First_name contains 'a' will be? [MCQ: 3 points]

1. `Select * from Worker where First_name like '%a';`
2. `Select * from Worker where First_name like 'a%';`
3. `Select * from Worker where First_name like '%a%';`
4. `Select * from Worker where First_name like '_a%';`

Answer: Option 3.

Solution:

Option 1 will yield all the workers whose first name ends with 'a'.

Option 2 will yield all the workers whose first name starts with 'a'.

Option 3 will yield the required result.

Option 4 will yield all the workers whose first name has 'a' at second position.

4] Consider the table 'Customer' in sqlite database.

CustomerID	CustomerName	City	Country
1	Manuel Pereira	Portland	USA
2	Mario Pontes	Montreal	Canada
3	John Steel	Paris	France
4	Georg Pipps	Seattle	USA
5	Janete Limeira	Vancouver	Canada
6	Michael Holz	Toulouse	France
7	Paul Henriot	Frankfurt	Germany

What will be the correct SQL query to find the position of the alphabet 'e' in the string 'Manuel Pereira' from Customer table? [MSQ: 3 points]

1. `Select SUBSTR(CustomerName, 'e') from Customer where CustomerName = 'Manuel Pereira';`
2. `Select INSTR(CustomerID, 'e') from Customer where CustomerName = 'Manuel Pereira';`

3. `Select INSTR(CustomerName, 'e') from Customer where CustomerName = 'Manuel Pereira';`
4. `Select INSTR(CustomerName, 'e') from Customer where CustomerID = 1;`

Answer: Option 3 and 4.

Solution:

Option 1 will return 'Manuel Pereira'.

Option 2 will return 0.

Option 3 will return 5 which is correct answer.

Option 4 will return 5 which is correct answer.

5] Consider the following flask app and a templating file.

File name: app.py

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

Templates file name: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Title Here</title>
</head>
<body>
<form method="POST" action="/">
  <label for="first_name">First name:</label><br>
  <input type="text" id="first_name" name="first_name"><br>
  <label for="last_name">Last name:</label><br>
  <input type="text" id="last_name" name="last_name"><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Which of the following statements is/are true in case we are running the flask application on http://127.0.0.1:5000 in our web browser? [MCQ: 3 points]

1. It will throw a 'Method not Allowed' error whenever we try to submit the form.
2. It will submit the input form data succcessfuly.
3. After running the app.py file, it will throw a 'Page not found' error.

4. None of the above

Answer: Option 1.

Solution: Whenever we run app.py file, it will display a form, and whenever we try to submit any data to that form,it will show a 'Method not Allowed' error. This error can be resolved by editing our app.py file to account for POST requests to our web application. We simply add 'methods=["GET", "POST"]' to our route decorator. This will solve the "Method not Allowed" error.

6] Consider the flask app given below.

```
from flask import Flask
app = Flask(__name__)
Customer_info = { 1:'Government Official', 2:'Businessman',
3:'Entrepreneur', 4:'Freelancer'}

@app.route('/customer/<int:customer_id>')
def random(customer_id):
    Id = customer_id
    descr = Customer_info[Id]
    return '''<h3> Customer Information: </h3>
              <p> The entered customer is '''+ descr +".</p>"

if __name__ == '__main__':
    app.run(debug = True)
```

What will the browser render for url: http://localhost:5000/customer/3 ? [MCQ: 2 points]

1. **Customer Information:**
   The entered customer is Government Official.
2. **Customer Information:**
   The entered customer is Entrepreneur.
3. **Customer Information:**
   The entered customer is Businessman.
4. **Customer Information:**
   The entered customer is Freelancer.

Answer: Option 2.

Solution: The url corresponds to the 3rd key of the dictionary Customer_info. Hence, the 'descr' variable takes 'Entrepreneur' as its value and Option 2 gets rendered.

7] Consider the following flask application.

```
from flask import Flask

app = Flask(__name__)

@app.route('/aboutpage')
def Home():
    return 'This is my home page'

@app.route('/projectpage/')
def projects():
    return 'The project page'

@app.route('/aboutpage/projectpage//')
def result():
    return 'This is about the project page'

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

Which of the following statements is/are true? [MSQ: 3 points]

1. For the URL 'http://127.0.0.1:5000/aboutpage/', the browser will render 'This is my home page'.
2. For the URL 'http://127.0.0.1:5000/projectpage/', the browser will render the same as for the URL 'http://127.0.0.1:5000/projectpage'.
3. For the URL 'http://127.0.0.1:5000/aboutpage/projectpage', the browser will show 'Page not found' error.
4. For the URL 'http://127.0.0.1:5000/aboutpage/projectpage/', the browser will render 'This is about the project page'.

Answer: Option 2 and 4.

Solution: . For the URL 'http://127.0.0.1:5000/aboutpage/', the browser will show a 'Page not found' error. For the URL 'http://127.0.0.1:5000/aboutpage/projectpage', the browser will render 'This is about the project page'. If you access this URL without a trailing slash (/aboutpage/projectpage), Flask redirects you to the URL with the trailing slash (/aboutpage/projectpage/).

8] Consider the following flask app and a templating file.

File name: app.py

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("home.html")

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

Templates file: home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Home Page</title>
</head>
<body>
    <strong> This is my Home Page</strong><br>
    {% for x in range(0, 20) %}
        {% if x % 3==1 and x % 2==1 %}
    <p>{{ x }}</p>
        {% endif %}
    {% endfor %}
</body>
</html>
```

What will be rendered by the browser when app.py file is running on the URL http://127.0.0.1:5000/? [MCQ: 3 points]

1. **This is my Home Page**

   1

   7

   13

   19

2. **This is my Home Page**

   1

   4

   7

   10

   13

   16

   19

3. **This is my Home Page**

   4

   10

   16

4. None of the above

Answer: Option 1.

Solution: Here we are using jinja2 template with our flask application. Here the template file (e.g., home.html) contains variables and/or expressions, which get replaced with values when a template is rendered . The output of the above Python code is:

This is my Home Page

1

7

13

19

9] Why would the user want to build URLs using the URL reversing function `url_for()` instead of hard-coding them into the templates? [MSQ: 3 points]

1. User can change URLs in one go instead of needing to remember to manually change hard-coded URLs.
2. `url_for()` easily handles files that are even outside the templates folder.
3. The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.
4. If the application is placed within a folder inside the URL root, for example, in `/myapplication` instead of `/`, `url_for()` properly handles that.

Answer: Option 1, 3 and 4.

Solution: When compared to hard-coding URLs, reversing is typically more descriptive. Instead of remembering to manually change hard-coded URLs, you can modify all of your URLs at once. The escaping of special characters is handled transparently during URL creation. The routes generated are always absolute, avoiding unexpected behaviour in browsers due to relative paths. If your application is in a directory other than the URL root, such as `/myapplication` instead of `/`, `url_for()` will take care of it.

10] The correct sequence of inserting data into the database using Flask-SQLAlchemy is. [MCQ: 2 points]

1. Create a session ---> Add the Python object to the session ---> commit the session
2. Create a session ---> commit the session ---> Add the Python object to the session
3. Create a Python object ---> Add the Python object to the session ---> commit the session
4. Create a Python object ---> commit the session ---> Add the Python object to the session

Answer: Option 3.

Solution: Inserting data into the database using flask- sqlalchemy is done in three steps:
1) Creating the Python object
2) Adding it to the session
3) Finally commiting the session.

11] Consider the following table. A model class 'Company' is used to create a table 'company' in the database, which is shown in the figure. [MCQ: 3 points]

emp_id	emp_name	emp_salary	emp_address
1	rahul	2000	Patna
2	Amrutha	2500	Chennai
3	Rohan	4000	Delhi
4	Aakash	8000	Gujarat
5	Harshit	4500	Bangalore

What will be the output of the following Python code if the code is executed from the Python console? [MCQ: 3 points]

```
>>> record = Company.query.filter(Company.emp_name.like('A%')).all()
>>> for records in record:
...     print(records.emp_name)
```

1. <Company 2>
   <Company 4>
2. rahul
   Amrutha
   Rohan
   Aakash
   Harshit
3. Amrutha
   Aakash
4. None of the above

Answer: Option 3.

Solution: The first line of code segment will filter only those records whose name starts with a letter 'A'. After that, in the next line, the for loop prints all the employees name one-by-one.

12] Consider the following code.

```
from flask import Flask

app = Flask(__name__)

@app.route('/mypathroute/<path:url_path>')
def mypathroute(url_path):
    return 'The path is: ' +url_path

if __name__ == "__main__":
    app.run(debug=True)
```

Let this flask application is running on http://127.0.0.1:5000, then what will the browser render for URL http://127.0.0.1:5000/mypathroute/thisismydirectory/thisismysubdirectory/homepage? [MCQ: 4.5 points]

1. It will throw 404 'page not found' error.
2. It will render as 'The path is: thisismydirectory/thisismysubdirectory.
3. It will render as 'The path is: mypathroute/thisismydirectory/thisismysubdirectory
4. It will render as 'The path is: thisismydirectory/thisismysubdirectory/homepage.

Answer: Option 4.

Solution: Here we are using the path convertor. A path convertor is like string but also accepts slashes. So here in this case, for the URL 'http://127.0.0.1:5000/mypathroute/thisismydirectory/thisismysubdirectory/homepage' the browser will render: The path is: thisismydirectory/thisismysubdirectory/homepage

13] A Model class 'Product' creates a table 'product' in the database as given below with 'Product_ID' as the primary key.

Product_ID	Product_Name	Units	Unit_price
110	Pipes	25	85
111	bolts	4500	15
112	Cement	10	250
113	Bricks	5000	19
114	Iron bars	60	94
115	Grills	40	450

The user wants to retrieve a list which consists only names of the listed products. The correct function that does that is? [MSQ: 3 points]

1.
```
@app.route("/products", methods=["GET"])
def Names():
    Products = [product.Product_Name for product in Product.query.all()]
    return "The list of products is:", Products
```
2.
```
@app.route("/products", methods=["POST"])
def Names():
    Products = [product.Product_Name for product in Product.query.all()]
    return "The list of products is:", Products
```
3.
```
@app.route("/products", methods=["GET"])
def Names():
    Products = [Product_Name for product in Product.query.all()]
    return "The list of products is:", Products
```
4.
```
@app.route("/products", methods=["GET"])
def Names():
    Products = Product.query.all()
    return "The list of products is:", Products
```

Answer: Option 1 and 2.

Solution: Option 1 will give the list of all the products in the 'product' table. In option 2, the only difference is the declaration of 'methods' parameter as 'POST' instead of 'GET' which does not make any difference as the same can be done even with using POST method.

14] A Model class 'Product' creates a table 'product' in the database object 'db' as given below with 'Product_ID' as the primary key.

Product_ID	Product_Name	Units	Unit_price
110	Pipes	25	85
111	bolts	4500	15
112	Cement	10	250
113	Bricks	5000	19
114	Iron bars	60	94
115	Grills	40	450

The owner of the inventory wants to add a new product 'Gypsum' in the 'product' table whose 75 units are available with Rs. 60 as the cost of one unit. The function that correctly does this is?

[MCQ: 4.5 points]

1.
   ```
   @app.route("/products", methods=["GET","POST"])
   def New():
       if request.method == "POST":
          new_prod = Product(Product_Name = 'Gypsum',Units = 75, Unit_price
   = 60)
          db.session.add(new_prod)
       return redirect('/')
   ```

2.
   ```
   @app.route("/products", methods=["GET","POST"])
   def New():
       new_prod = Product(Product_Name = 'Gypsum',Units = 75, Unit_price =
   60)
       db.session.add(new_prod)
       db.session.commit()
       return redirect('/')
   ```

3.
   ```
   @app.route("/products", methods=["GET","POST"])
   def New():
       db.session.add(Product_Name = 'Gypsum')
       db.session.commit()
       return redirect('/')
   ```

4.
   ```
   @app.route("/products")
   def New():
       if request.method == "POST":
          new_prod = Product(Product_Name = 'Gypsum',Units = 75, Unit_price
   = 60)
          db.session.add(new_prod)
          db.session.commit()
       return redirect('/')
   ```

Answer: Option 2.

Solution:

Option 1. This code snippet misses the line `db.session.commit()` which is a required step to save entry into the database.
Option 2. It correctly adds the entry.
Option 3. Only Name is defined for the product.
Option 4. It does not define methods in the route function.

15] The array 'brands', rendered as an API whose access url is: "http://127.0.0.1:5000/api/brands/all" with the help of a Flask app is shown below. The route function that retrieves a particular gadget from the API is also given below. [MCQ: 4.5 points]

```
brands = [  { 'brand': 'Acer', 'description': 'it is a Taiwanese
multinational hardware and electronics corporation.'},
            { 'brand': 'Apple', 'description': 'it is an American
multinational electronics company.'},
            { 'brand': 'Noise', 'description': 'it is one of the leading
Indian tech brands.'},
            { 'brand': 'Xiaomi', 'description': 'it is a Chinese designer
and manufacturer of consumer electronics.'}
]
```

```
from flask import Flask, request, render_template
app = Flask(__name__)
@app.route('/api/brands', methods= ["GET"])
def descr():
    if 'brand' in request.args:
        this_brand = request.args['brand']
    output = []
    for brand in brands:
        if brand['brand'] == this_brand:
           output.append(brand)
           descrp = brand['description']
    return render_template('out.html', this_brand = this_brand, descrp =
descrp)

if __name__ == '__main__':
    app.run(debug = True)
```

The templates folder in the root directory has an HTML file `out.html` given as:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{{ this_brand }}</title>
</head>
<body>
    <p>You have selected <strong>{{ this_brand }}</strong> and {{ descrp }}
</p>
</body>
</html>
```

What will be the output on browser if the user hits the url: "http://127.0.0.1:5000/api/brands?brand=Noise" ?

1. You have selected **Noise** and it is a Taiwanese multinational hardware and electronics corporation.
2. You have selected **Noise** and it is an American multinational electronics company.
3. You have selected **Noise** and it is one of the leading Indian tech brands.
4. You have selected **Noise** and it is a Chinese designer and manufacturer of consumer electronics.

Answer: Option 3.

Solution: The given Flask app retrieves that dictionary from the API whose value of "brand" is "Noise" and displays the brand name and its decription according to the template `out.html` .

16] Why PUT method is considered as idempotent whereas POST method is not in the context of the REST? [Descriptive: 4.5 points]

Answer: The PUT method is used to update a resource on the server. The subsequent identical PUT requests are safely repeatable as the data remains the same, whereas the identical POST requests will end up creating duplicate resources if repeated multiple times. Thus, POST is not considered as idempotent.