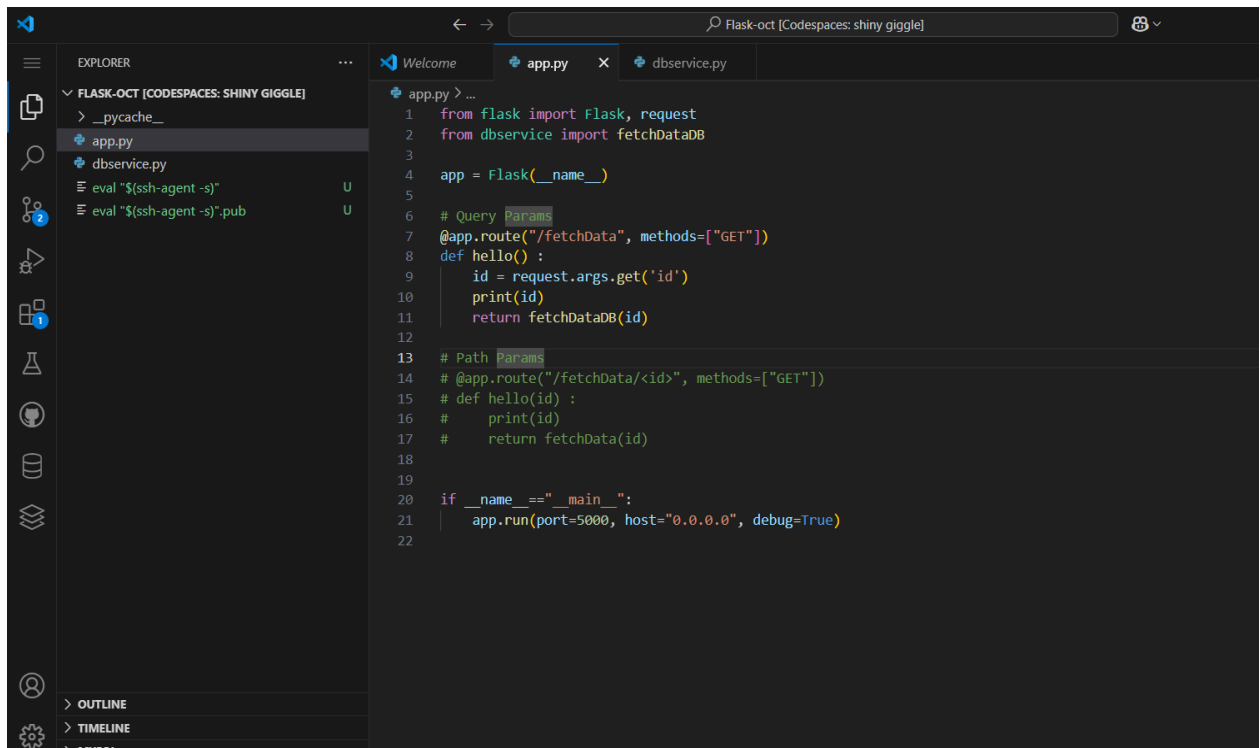
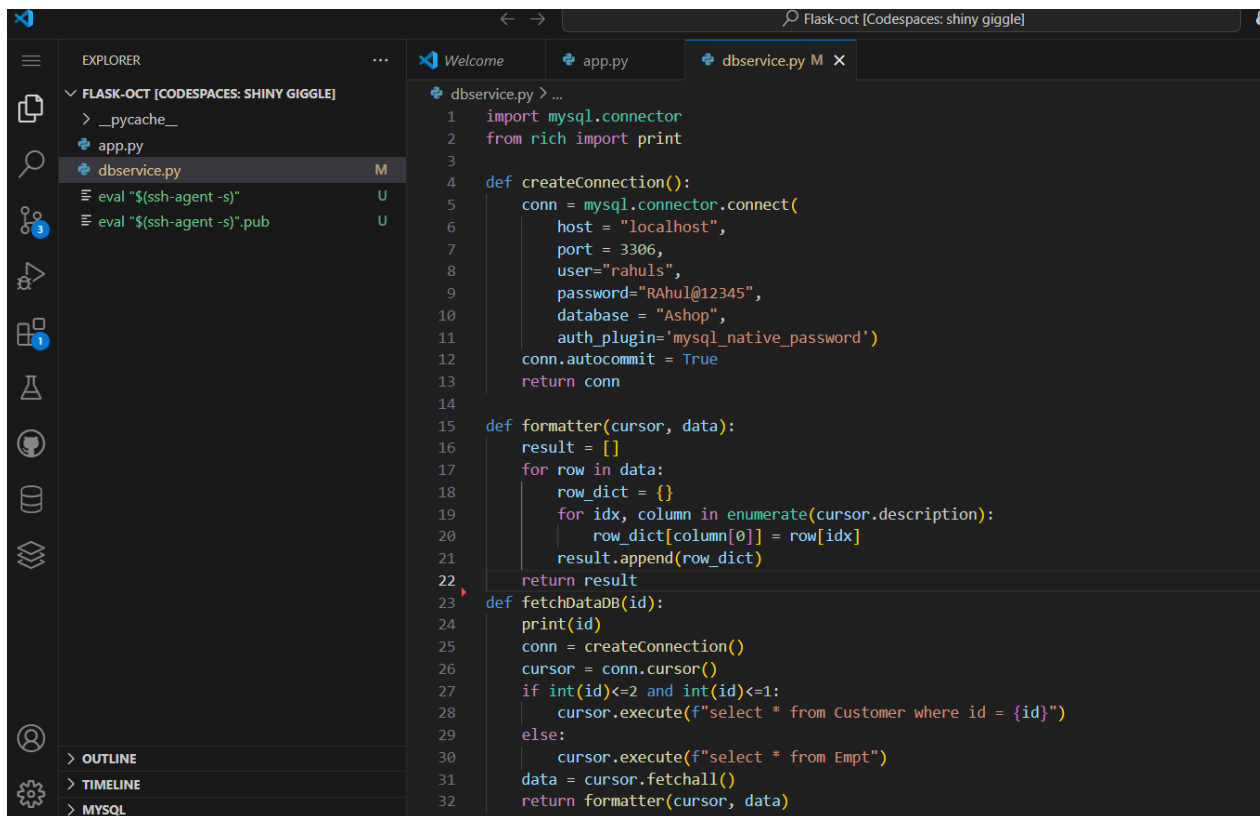


Code:



The screenshot shows the VS Code editor interface with the Explorer sidebar on the left. The Explorer sidebar shows the project structure for 'FLASK-OCT [CODESPACES: SHINY GIGGLE]'. The file 'app.py' is selected. The main editor area displays the code for 'app.py'.

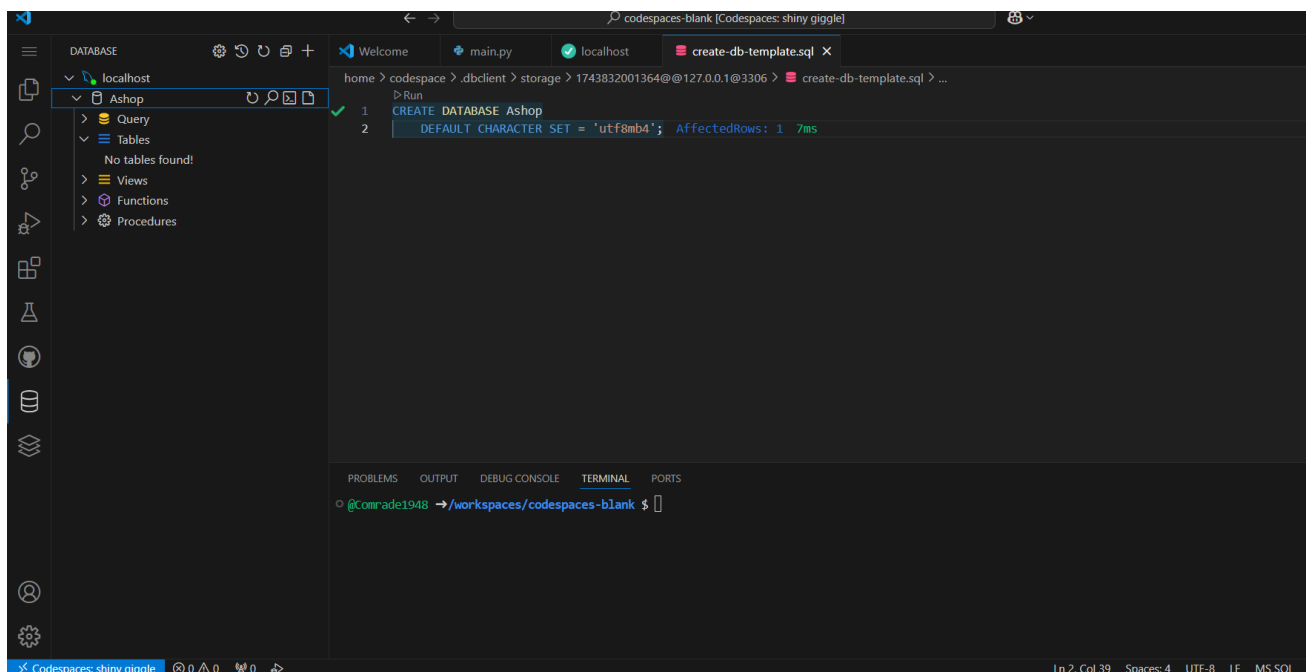
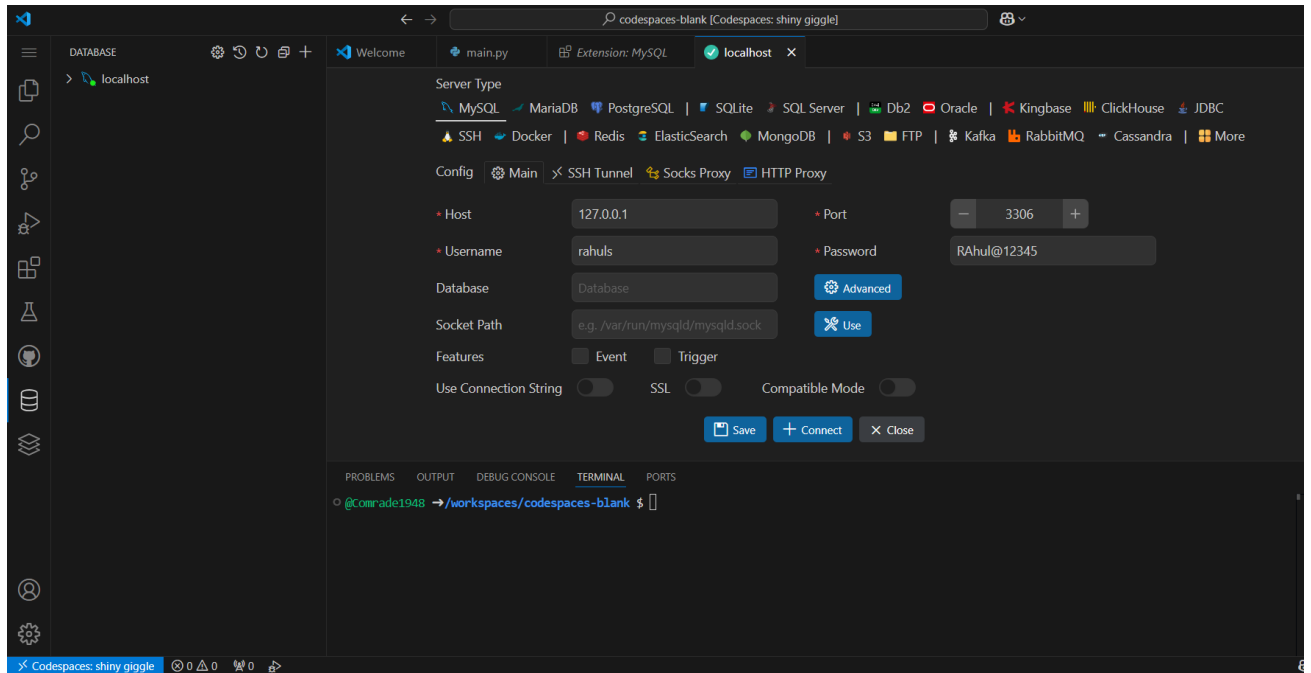
```
1 from flask import Flask, request
2 from dbservice import fetchDataDB
3
4 app = Flask(__name__)
5
6 # Query Params
7 @app.route("/fetchData", methods=["GET"])
8 def hello():
9     id = request.args.get('id')
10    print(id)
11    return fetchDataDB(id)
12
13 # Path Params
14 # @app.route("/fetchData/<id>", methods=["GET"])
15 # def hello(id):
16 #     print(id)
17 #     return fetchData(id)
18
19
20 if __name__ == "__main__":
21     app.run(port=5000, host="0.0.0.0", debug=True)
22
```



The screenshot shows the VS Code editor interface with the Explorer sidebar on the left. The file 'dbservice.py' is selected. The main editor area displays the code for 'dbservice.py'.

```
1 import mysql.connector
2 from rich import print
3
4 def createConnection():
5     conn = mysql.connector.connect(
6         host = "localhost",
7         port = 3306,
8         user="rahuls",
9         password="RAhul@12345",
10        database = "Ashop",
11        auth_plugin='mysql_native_password')
12    conn.autocommit = True
13    return conn
14
15 def formatter(cursor, data):
16     result = []
17     for row in data:
18         row_dict = {}
19         for idx, column in enumerate(cursor.description):
20             row_dict[column[0]] = row[idx]
21         result.append(row_dict)
22     return result
23
24 def fetchDataDB(id):
25     print(id)
26     conn = createConnection()
27     cursor = conn.cursor()
28     if int(id)<=2 and int(id)>=1:
29         cursor.execute(f"select * from Customer where id = {id}")
30     else:
31         cursor.execute(f"select * from Empt")
32     data = cursor.fetchall()
33     return formatter(cursor, data)
34
```

Database Creation:



Output:

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar shows a collection named "rahul". The main panel displays a POST request to "127.0.0.1:5000/fetchdata". The request is successful, returning a 200 OK status with a response time of 169 ms and a body size of 573 B. The response body is displayed in the "Body" tab, showing a JSON array of three user objects.

Request:

- Method: POST
- URL: 127.0.0.1:5000/fetchdata

Response:

```
2 {
3   "email_id": "rahul@gmail.com",
4   "id": 1001,
5   "name": "rahul"
6 },
7 {
8   "email_id": "akash@gmail.com",
9   "id": 1002,
10  "name": "Akash"
11 },
12 {
13   "email_id": "deepak@gmail.com",
```

The status bar at the bottom shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and "Help".

The screenshot shows the Postman interface with a workspace named "My Workspace". The left sidebar shows a collection named "rahul". The main panel displays a GET request to "127.0.0.1:5000/fetchdata". The request is failed, returning a 404 NOT FOUND status with a response time of 80 ms and a body size of 199 B. The response body is displayed in the "Body" tab, showing a single string message.

Request:

- Method: GET
- URL: 127.0.0.1:5000/fetchdata

Response:

```
1 "There is no record found"
```

The status bar at the bottom shows "Online", "Find and replace", "Console", "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and "Help".