

**Quantifeed Technical Test**

**Singapore**

**Submitted By:**

Abhijeet Srivastava

+65-86930736

Abhijeet.srivastava@live.com

21/02/2020

**Candidate Exercise:**

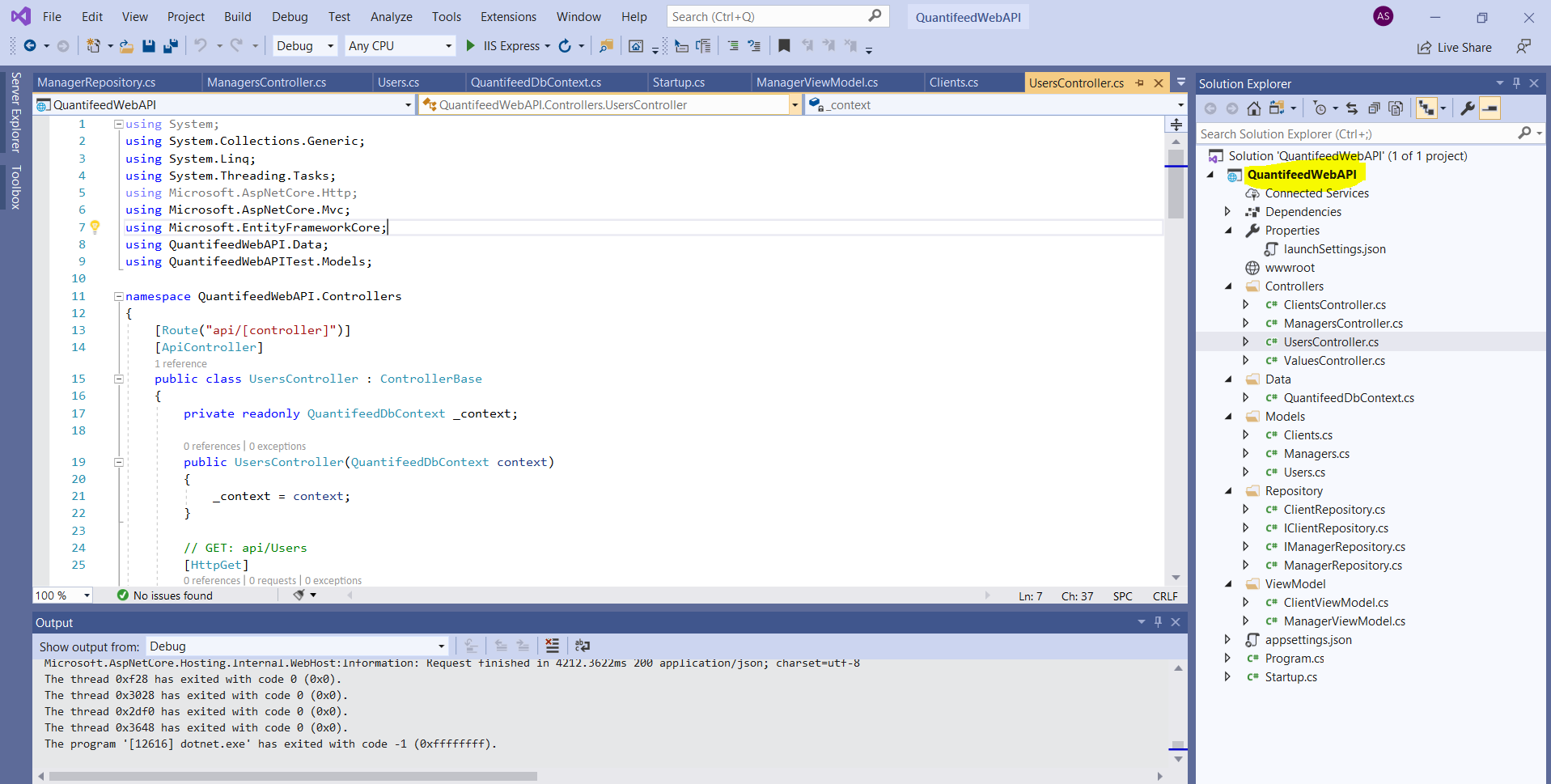
Using AspnetCore 2.1, EF Core 2.1 (database first), .net Standard 2.03 and Microsoft.NET.Test.Sdk 15.8 please design a web api solution that:

1. Can add, update, delete and query users to a database.
2. Contains a unit test library which tests each of the available functions
3. Runs as a console app

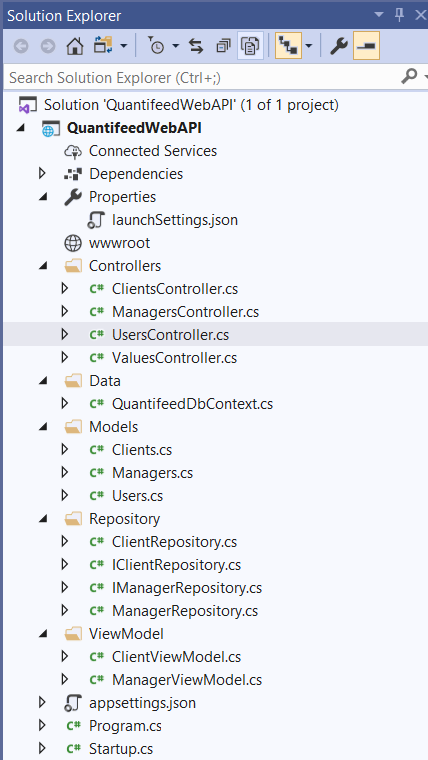
**Solution:**

I have developed the project "**QuantifeedWebAPI**" Web API using the EF Core2.1 and Aspnetcore2.1.

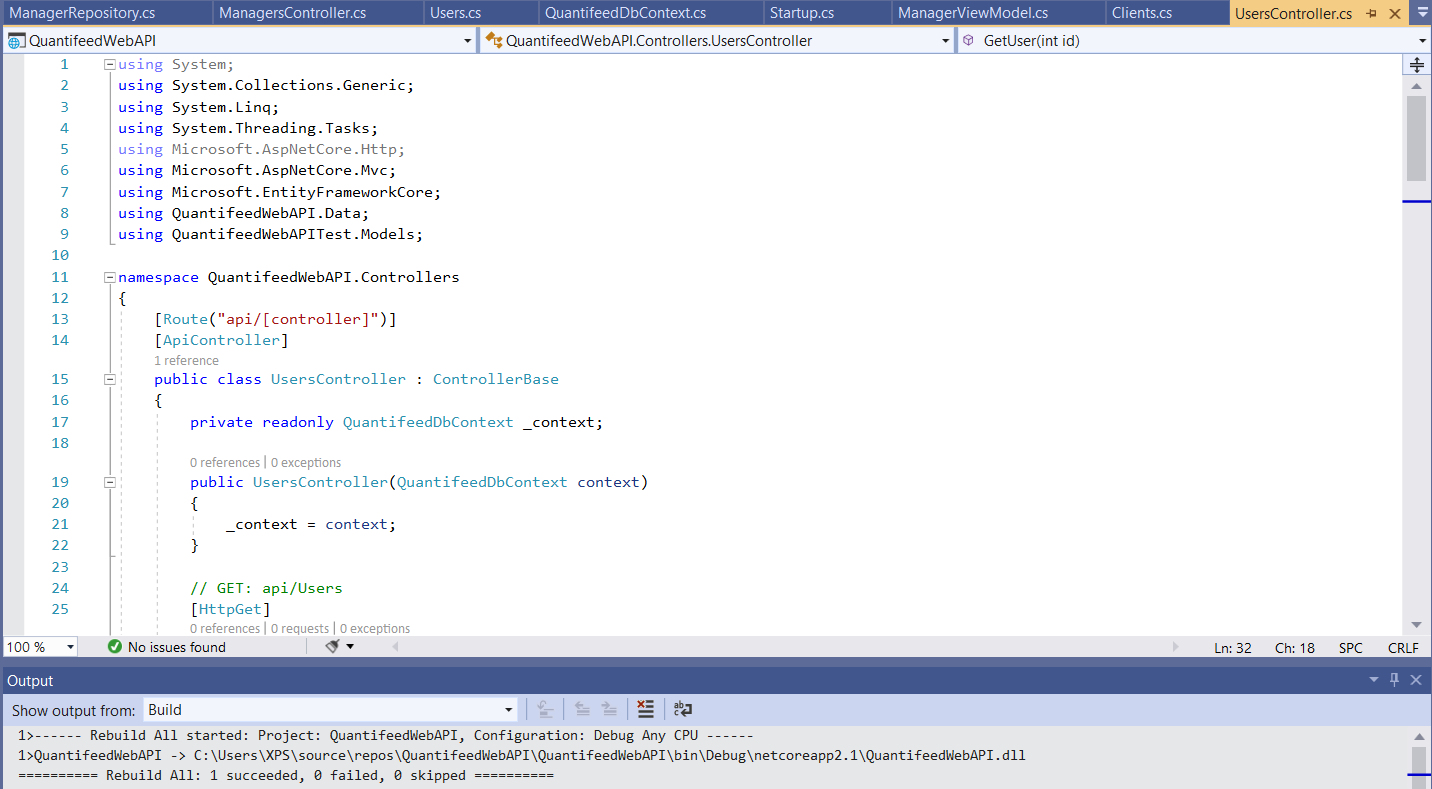
**Step 1:** Using the **QuantifeedWebAPI** project, the end application can add, update, delete and query **users** to a database. Open the **QuantifeedWebAPI project in the VS2019 or earlier version.**



The solution explorer will be shown like below Controllers, Data, Models, Repository and view model are the folders/program created to support the functionality.

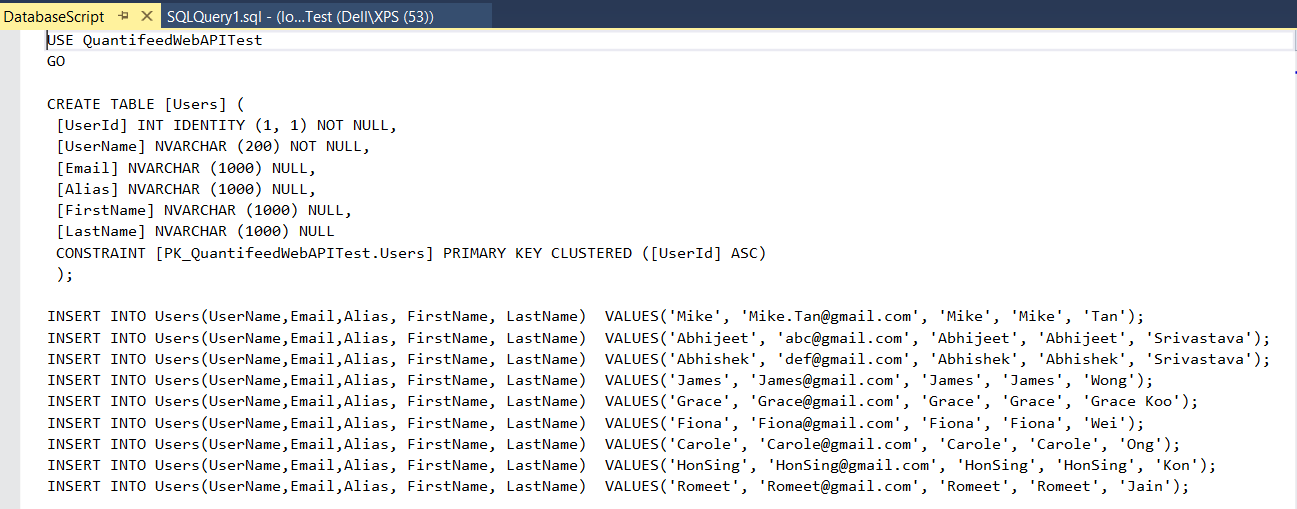


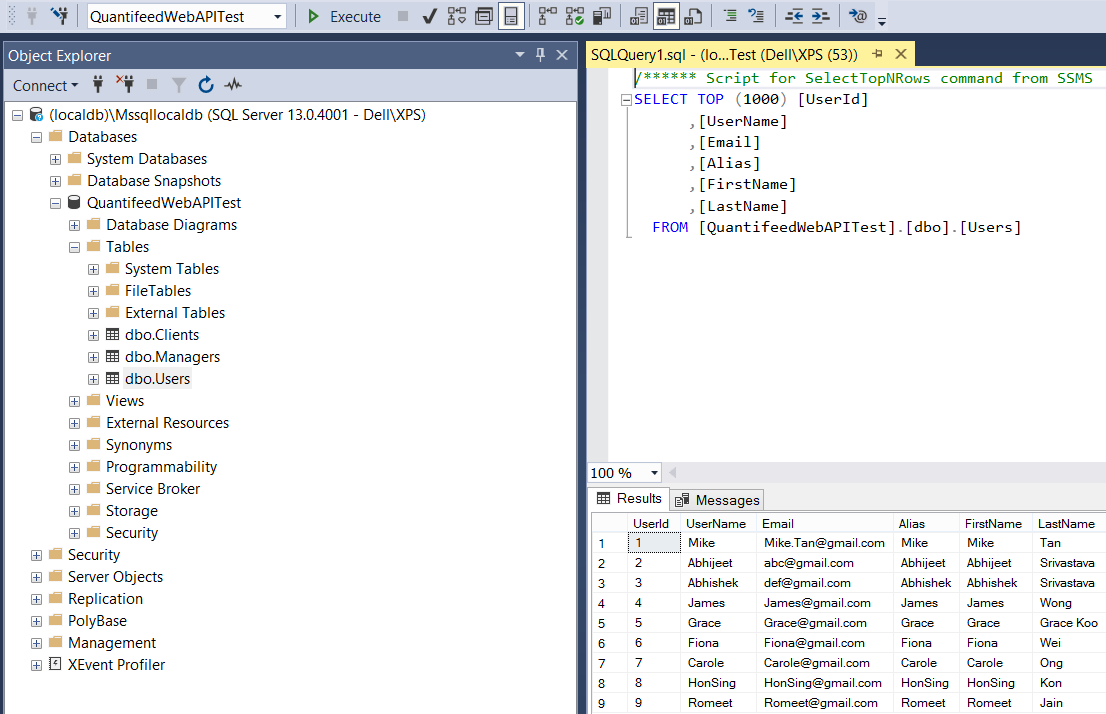
Build the solution, it should show build success message.



**Step 2: Need to create database** "**QuantifeedWebAPITest**"

The User table has been created in SQL server database. The script to create table and populate the dummy data are mentioned below:





**Step 3: Run the service to by clicking Ctrl+F5.**

It will show the all users records from underlying **"Users"** table.



**Step4: To test** add, update, delete and query users to a database.

I have created one console application "**QuantifeedWebAPIClient**" in. NetCore2.1.

Open the application **QuantifeedWebAPIClient** console application to test the API for above mentioned functionalities.



*Update the Base Address in the console application for the Uri of API.*

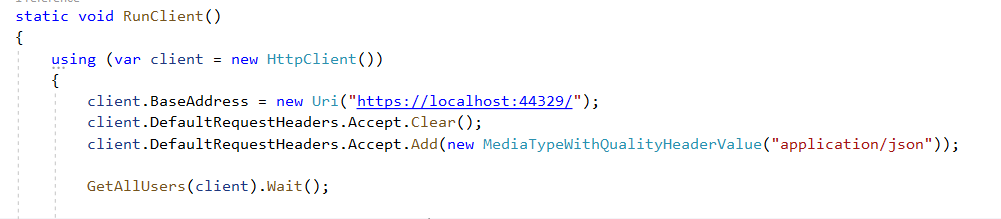
**client.BaseAddress = new Uri("https://localhost:44329/");**

and run the run the client application.

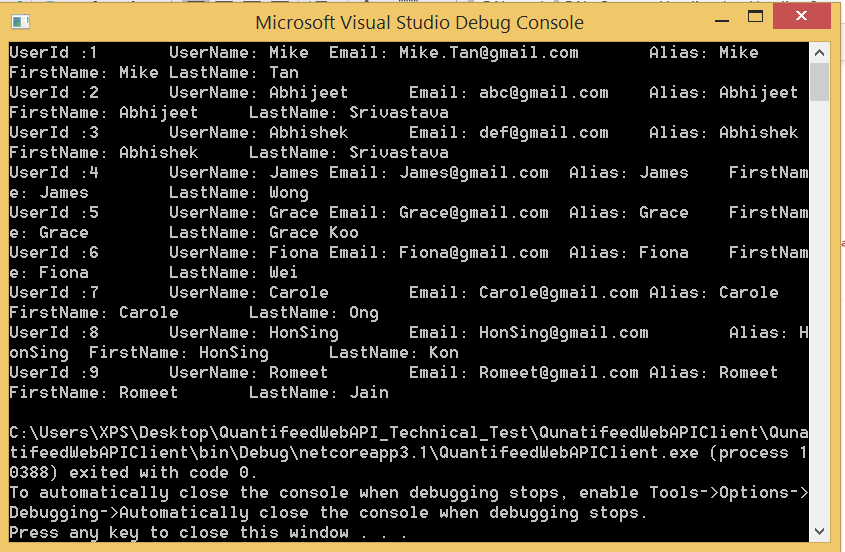
**To get the list of all users from Users Table(query users to a database):**

Uncomment the below line of code and run the application to get the list of all users from Users table:

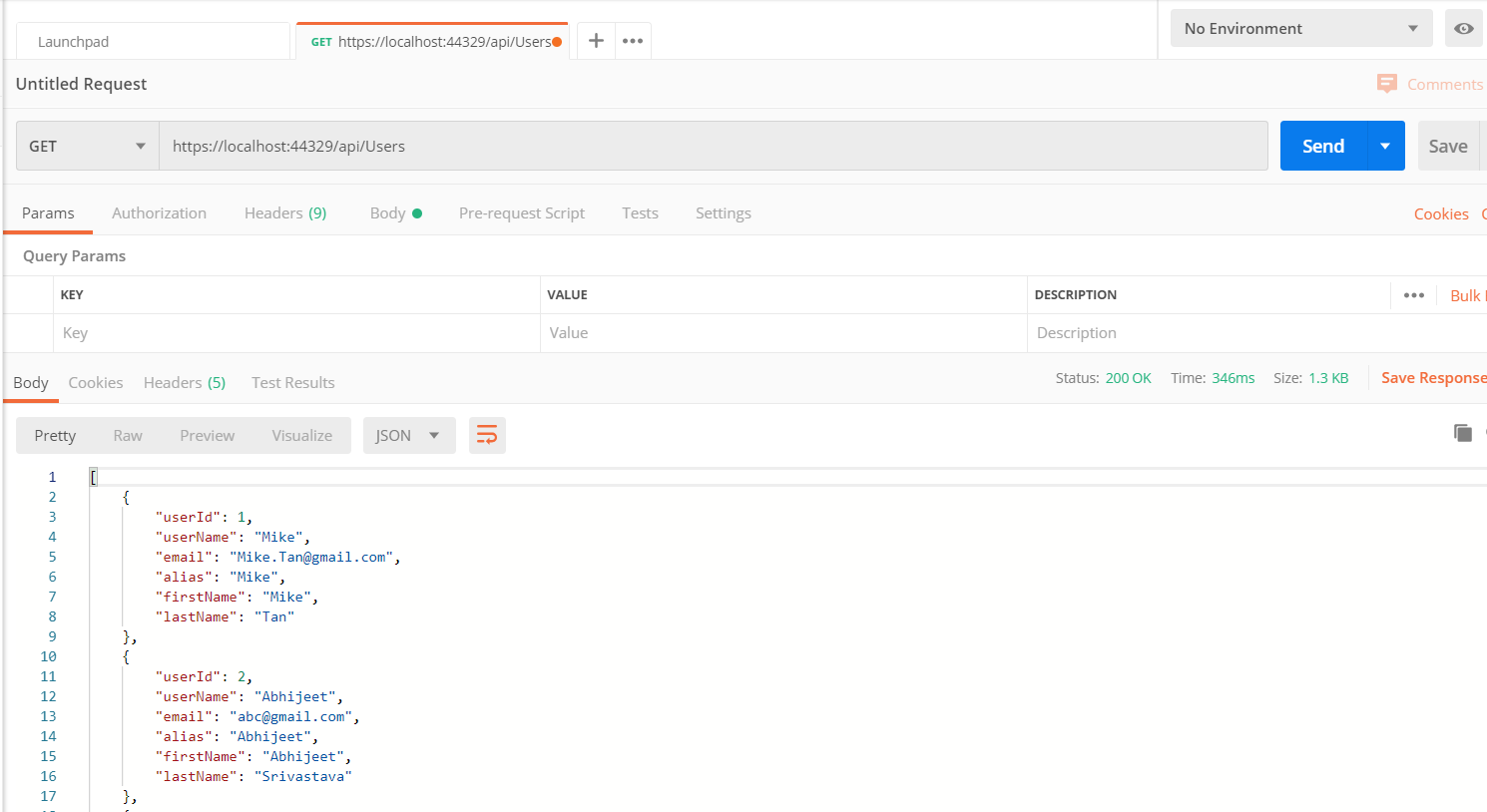
***GetAllUsers(client).Wait();***



**Output :**



**Using the Postman Tool to see the result from API call:**



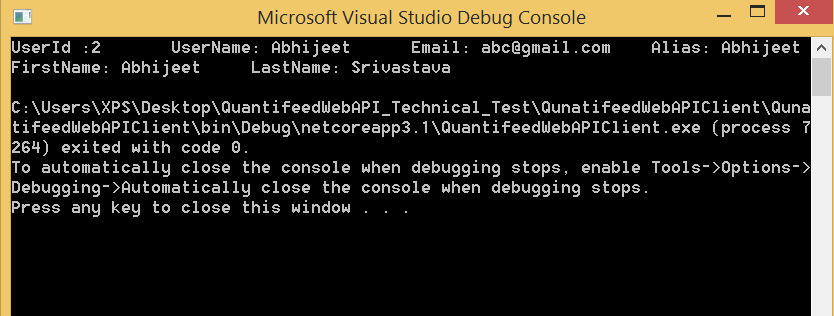
**To** **get a specific user from Users table by passing the UserID as parameter:**

Uncomment the below line of code and run the application to get a specific user from Users table by passing the UserId as parameter:

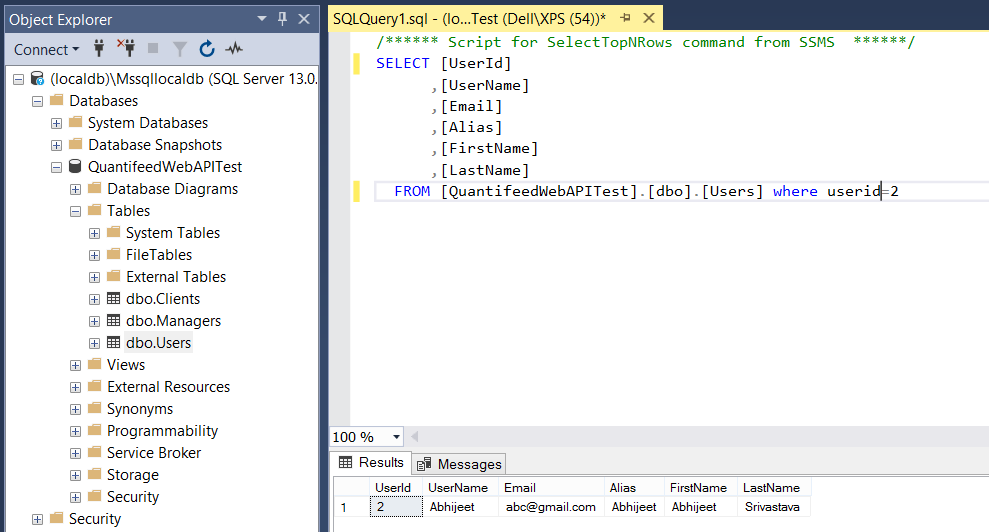
**GetUserByUserID(client, "2").Wait();**

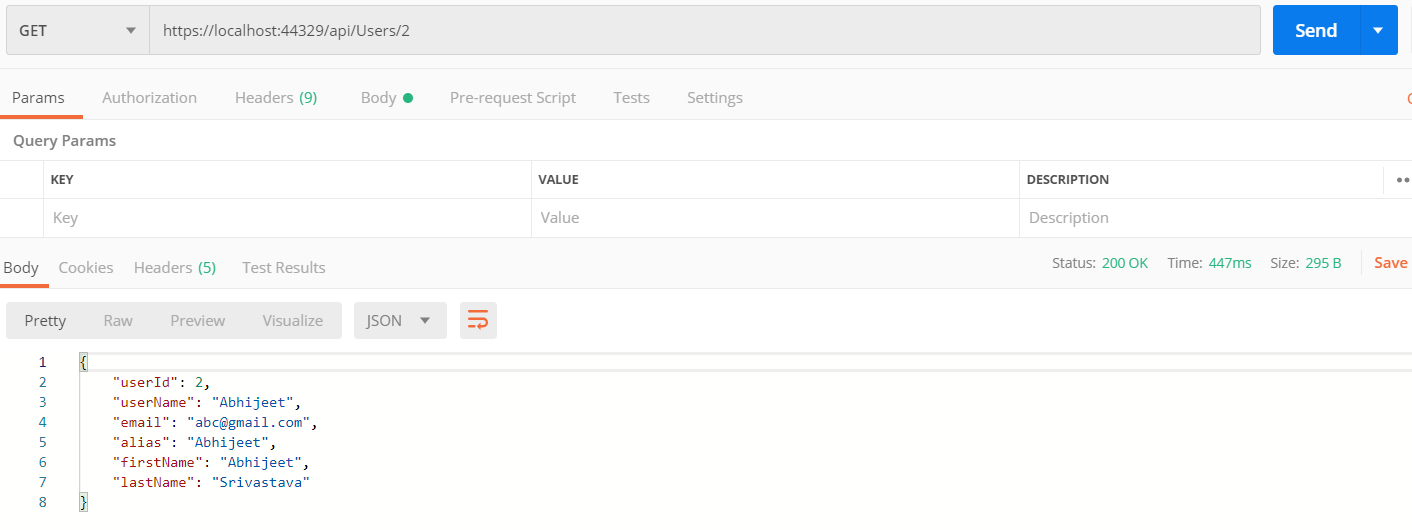


**Output:**



**Underlying value in table for the UserId=2**





**Add new user in the user table using the API Post call:**

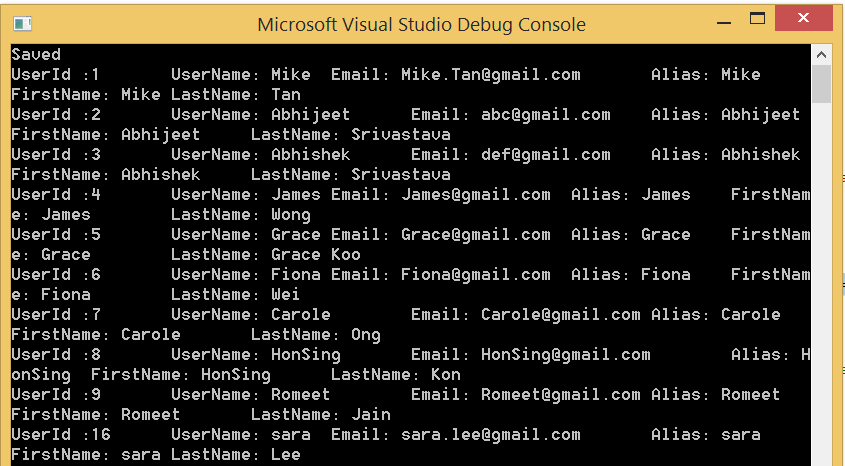
Uncomment the below line of code and run the application to get add specific user to the Users Table:

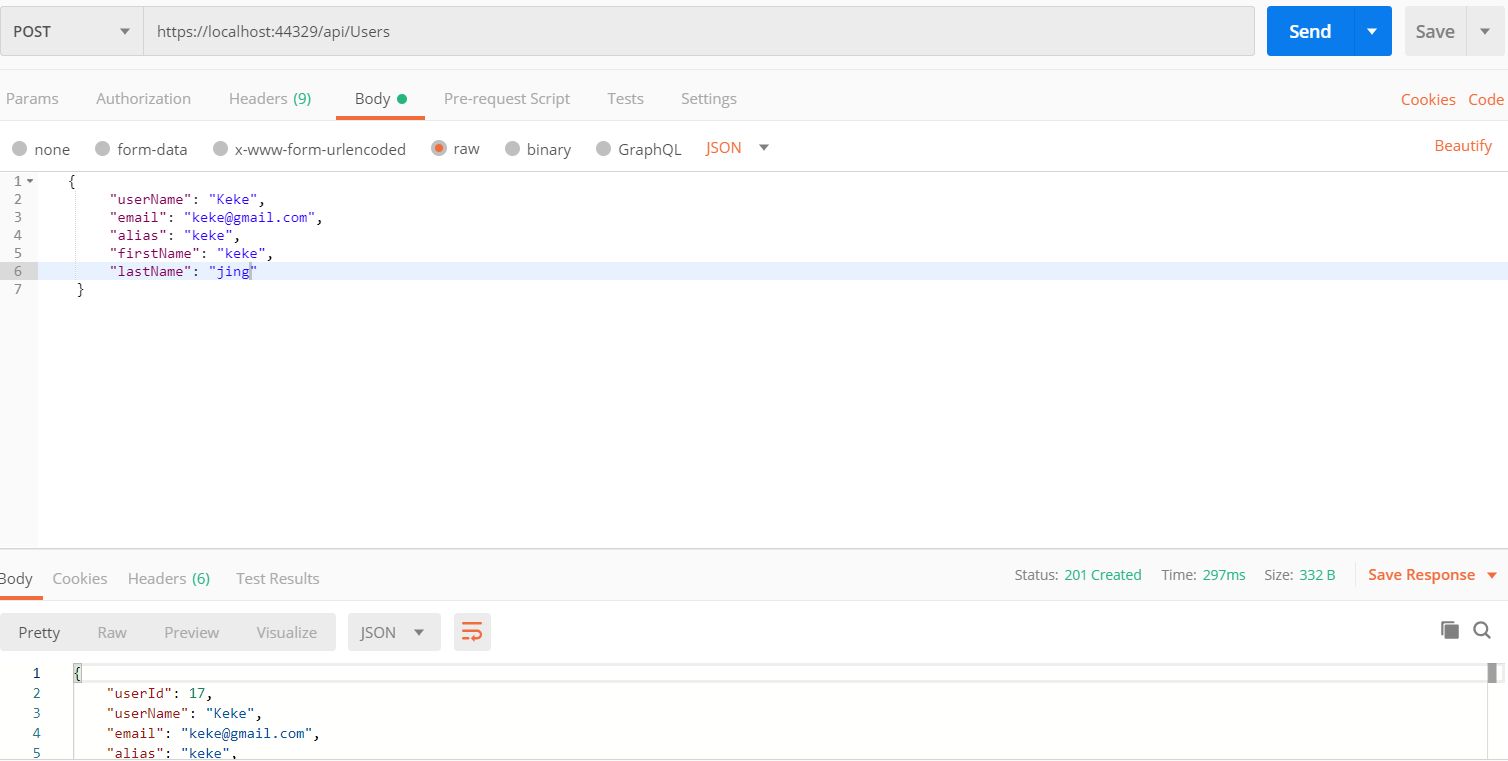
**PostUser(client, new User() { UserName = "sara", Email = "sara.lee@gmail.com", Alias = "sara", FirstName = "sara", LastName = "Lee" }).Wait();**

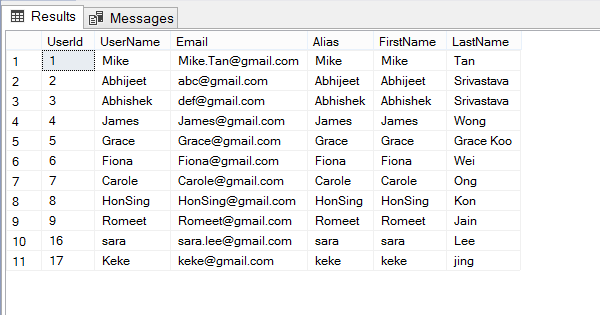
**GetAllUsers(client).Wait();**



**Output:**





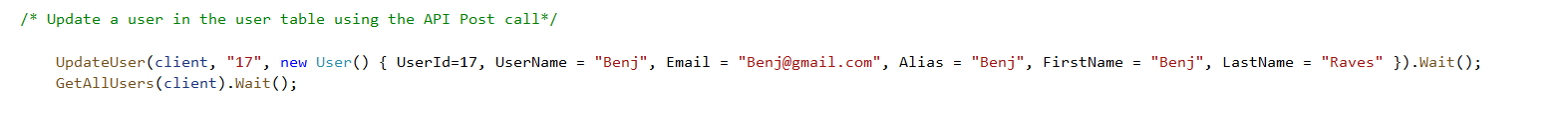


**Update a user in the user table using the API Post call:**

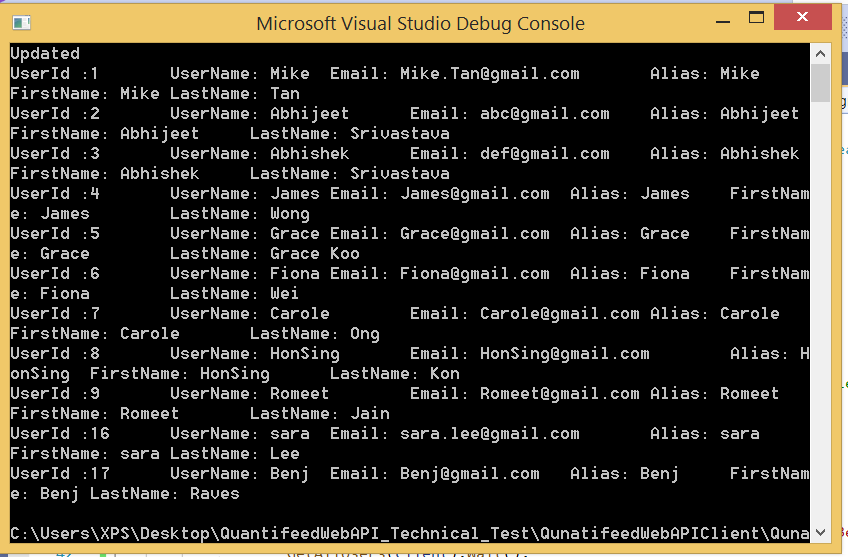
Uncomment the below line of code and run the application to update a specific user to the Users Table:

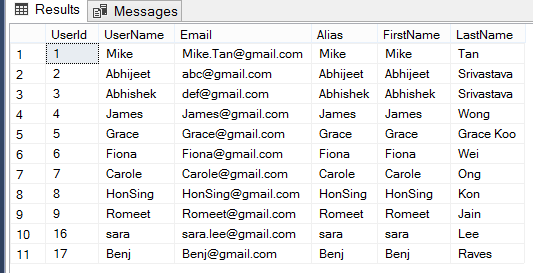
**UpdateUser(client, "17", new User() { UserId=17, UserName = "Benj", Email = "Benj@gmail.com", Alias = "Benj", FirstName = "Benj", LastName = "Raves" }).Wait();**

**GetAllUsers(client).Wait();**



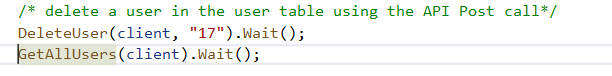
**Output:**





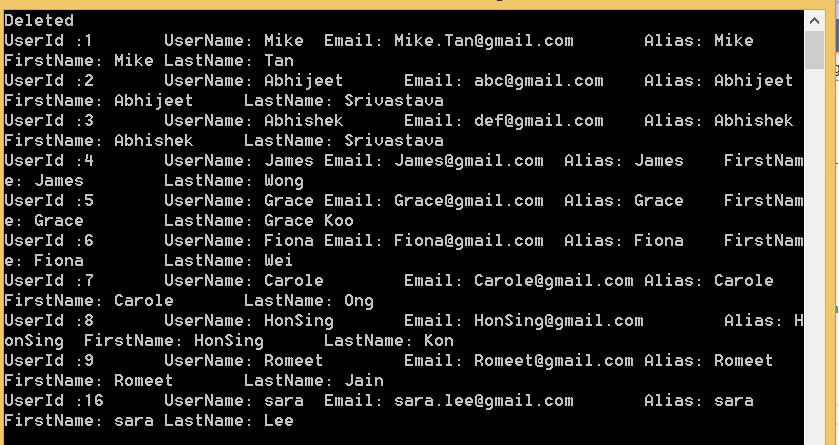
**Delete a user from a User table using the API call**

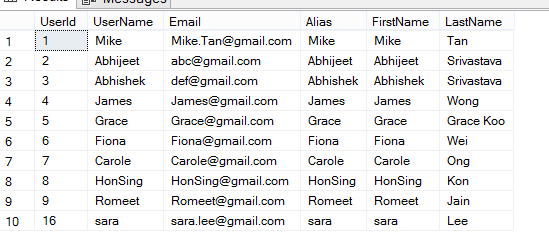
Uncomment the below line of code and run the application to delete a specific user to the Users Table:

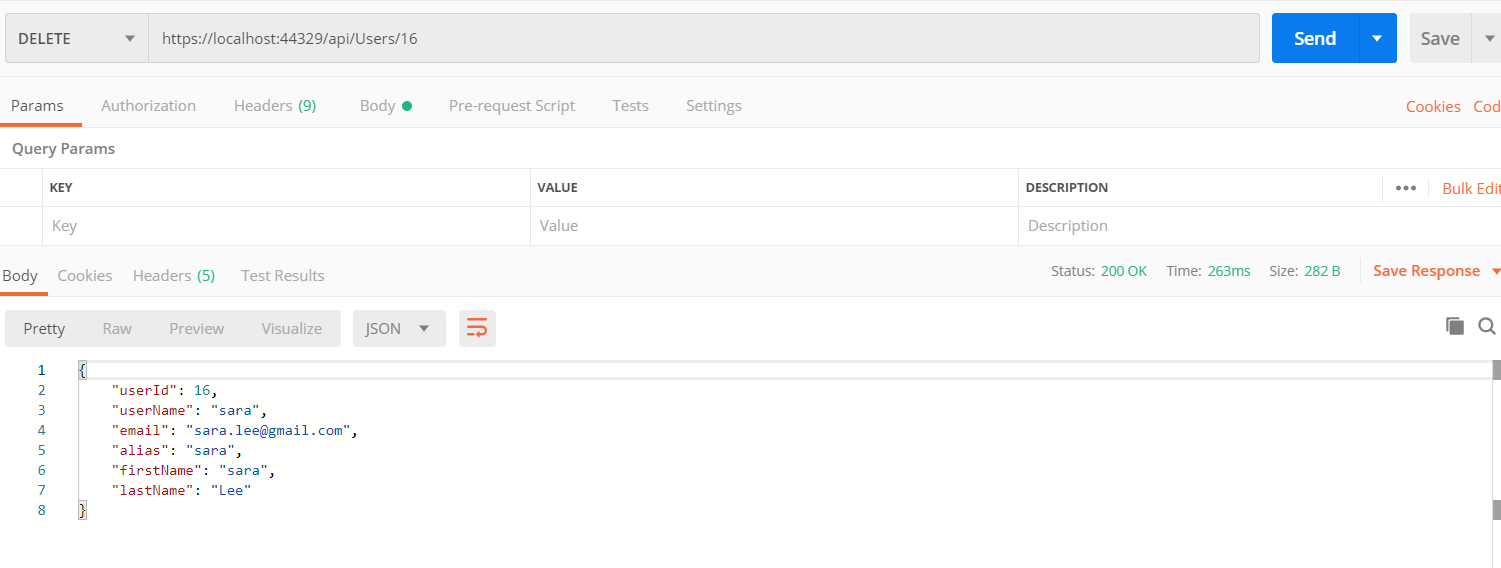


**Output:**

UserId: 17 record deleted from the user table.







**Extra :**

The Users are broken in to two types: Managers and Clients. Managers will have an additional 'Position' property (Junior, Senior) and Clients will have an int 'Level' property. Each manager has many clients.

Enhance the database and code to support this structure and create apis to :

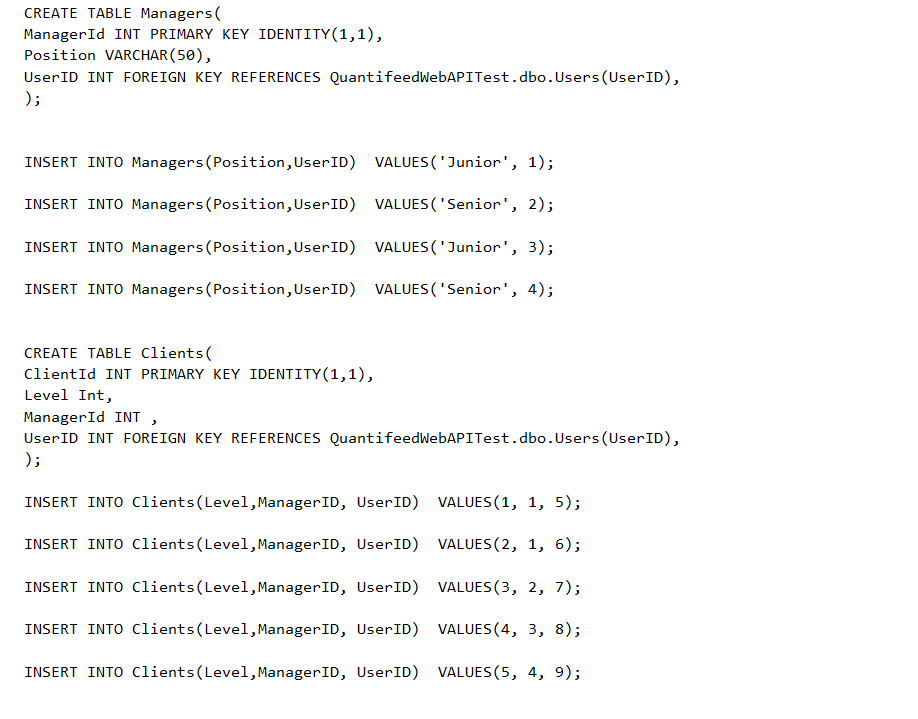
- Retrieve all Managers with their associated clients

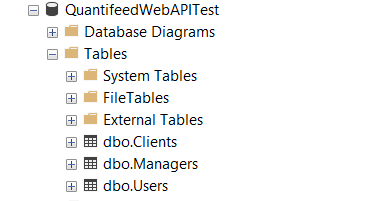
- Retrieve all Clients including their Manager

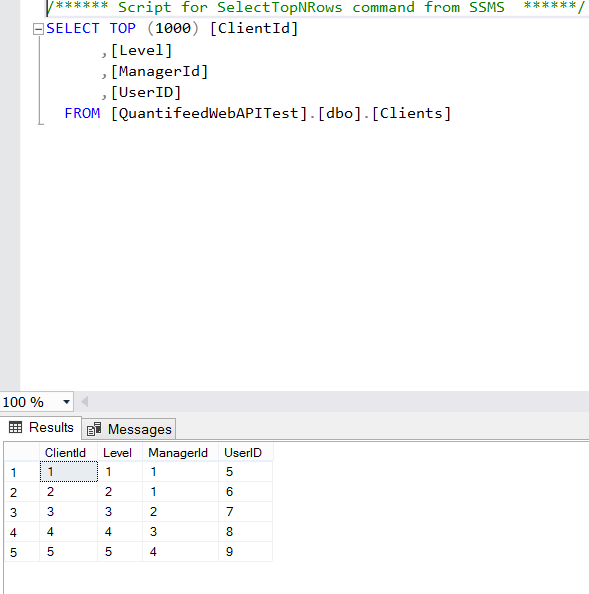
- For a specified Manager username, retrieve a list of its clients

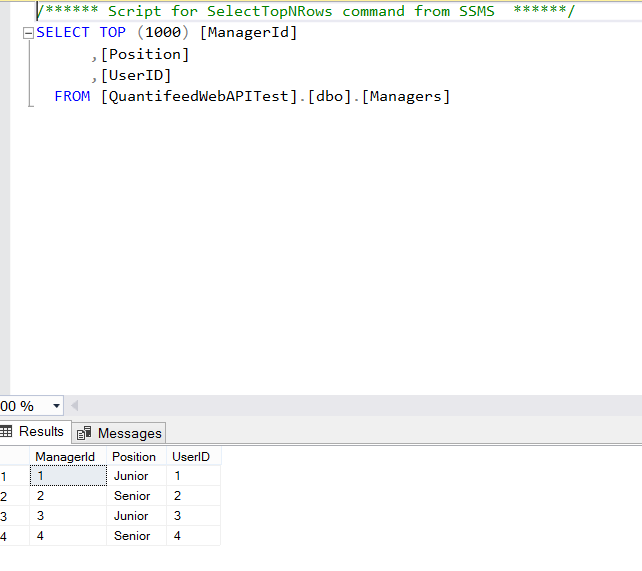
**Solutions :**

Create two tables **Managers** and **Clients** in the QuantifeedWebAPITest Database and populate the data using the scripts present in the DB script file.









The API method to achieve the above-mentioned queries have been implemented in the **QuantifeedWebAPI** project.

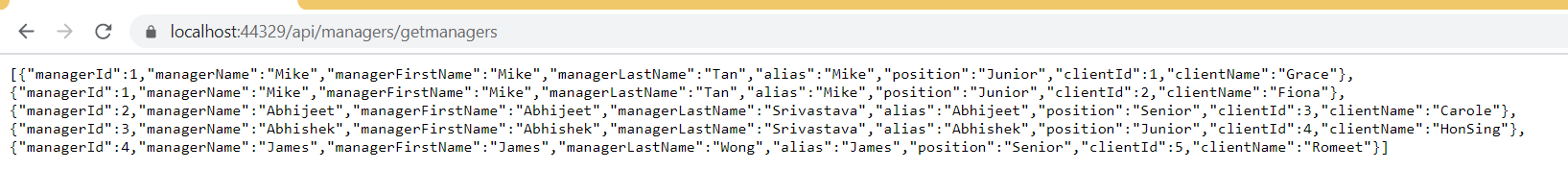
**Problem No. 1:** - Retrieve all Managers with their associated clients.

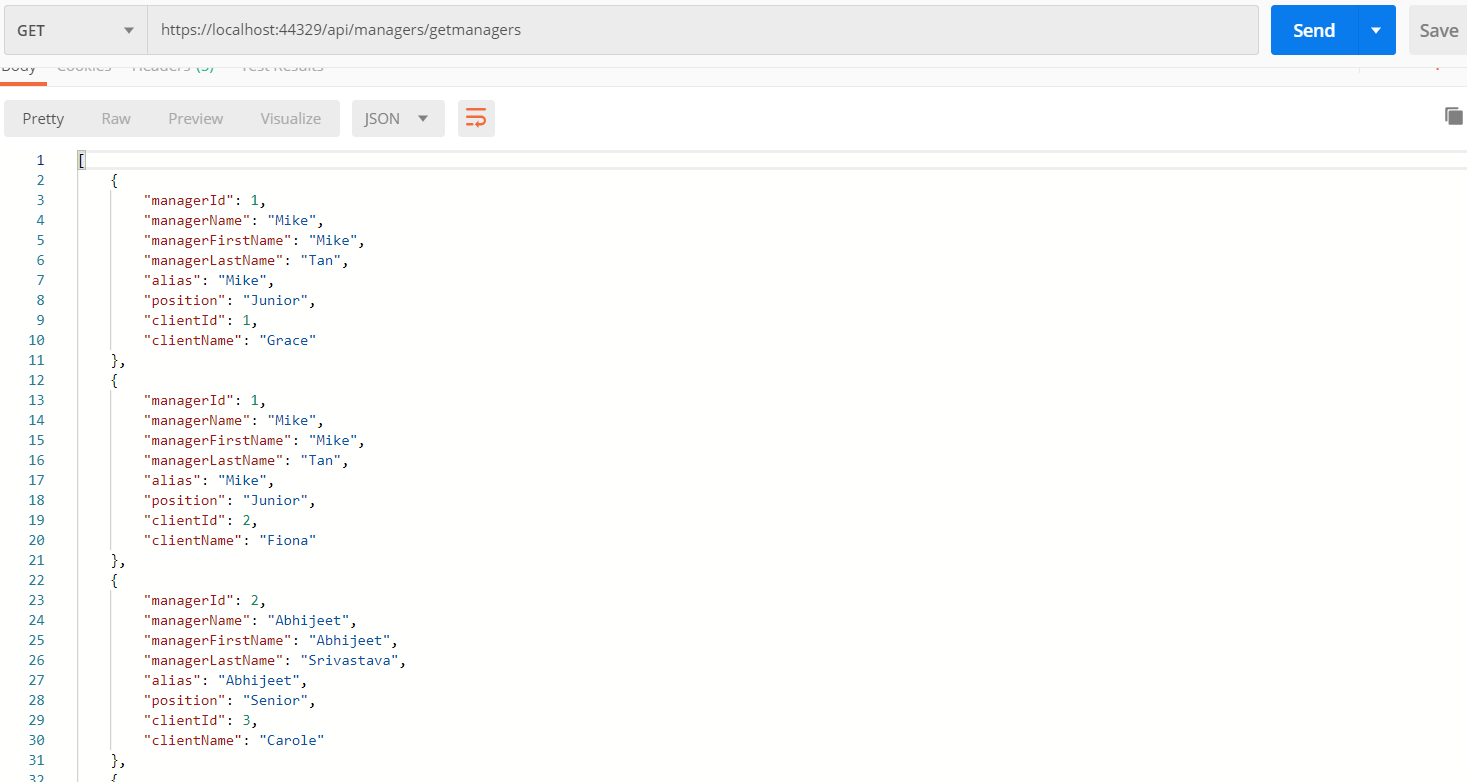
Run the QuantifeedWebAPI project, Call the API getmanagers method using the below URL.

<https://localhost:44329/api/managers/getmanagers>

**Output:**

List of all Managers with their associated clients.



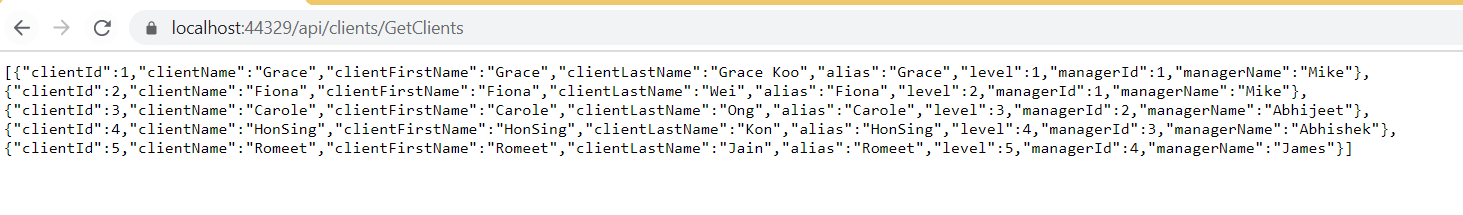


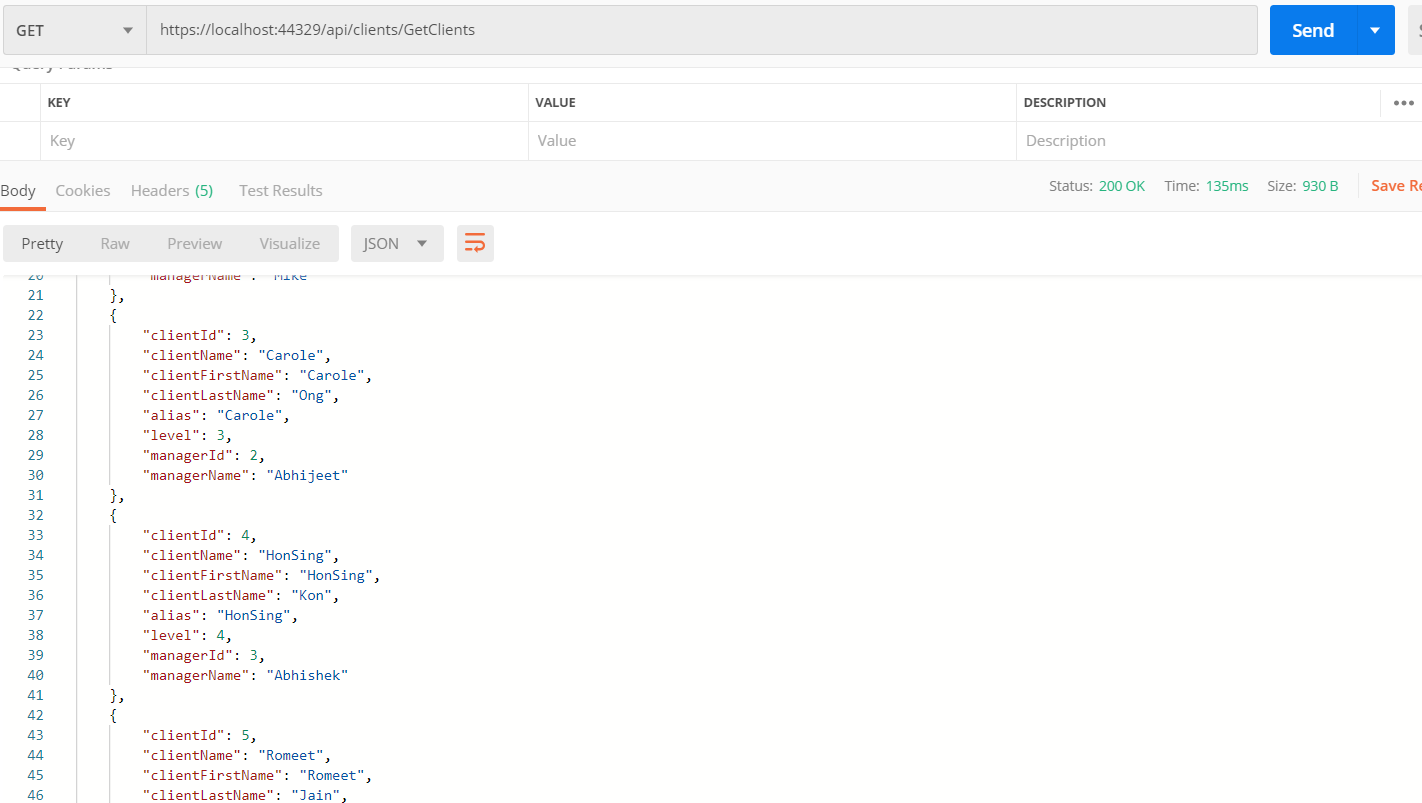
**Problem No. 2:** - Retrieve all Clients including their Manager.

Run the QuantifeedWebAPI project, Call the API getclients method using the below URL.

[**https://localhost:44329/api/clients/GetClients**](https://localhost:44329/api/clients/GetClients)

**Output:**





**Problem No. 3:** - For a specified Manager username, retrieve a list of its clients.

Run the QuantifeedWebAPI project, Call the API Managers/Username method using the below URL.

[**https://localhost:44329/api/managers/mike**](https://localhost:44329/api/managers/mike)

**Output:**

