**Final Project**

**Topic: Azure Search**

**Problem Statement:**

Whenever a company or any entity looking to develop their own search pages that returns results like the famous search service providers in Google, Bing, Amazon, Yahoo, then they are left with choices on its development, implementation and scalability. Building the search page in traditional ways might not be as effective or compelling as in today’s age, where user wants the search to be multi-functional and multi-featured. For example, user might like to see suggestions based on few keywords, see results based on 5-mile radiuses, or get results based on his/her preference and relevance, and desires to get those results back fast. On top of that, user might want to be able to properly navigate between search results and be able to narrow down those results applying different filters. This is when Azure Search Service come to play to address all those factors to provide rich and great user search experience.

As my final project, I have chosen to use “Azure Search Service” to build a search engine that lets user search for the restaurants by their name, type or zip code.

**Overview of Azure Search Service:**

Azure Search is a cloud based search-as-a-service solution with a powerful search experience over the content in web, mobile and applications. Hosted in Microsoft’s Azure cloud platform, it provides an ability to search with queries that can process language naturally (understands what user might be asking for), support geo search (allows user to search for geographic areas based on radius and coordinates), and give out accurate search recommendations.

Azure Search Service is easy to implement and configured to any apps developed in various programming languages (Android, iOS, Windows etc). It provides high scalability (able to scale up or down based on search need changes), and due to its cloud based nature provides optimal performance.

**High Level Steps:**

1. Create a Azure Search Service in the Azure Portal.
2. Create Azure SQL Database.
3. Load data into the table.
4. Import data and create index to Azure Search using the portal.
5. Install Node.js.
6. Download sample code from GitHub.
7. Extract the code into VS Code editor and run npm commands.
8. Make changes to config file to point towards Azure Search Service and Azure Data Center.
9. Compile the code and run the search application.

**Data Source:**

Azure SQL Server.

Manually entered restaurant data collected from google.

**Hardware/OS used:**

Intel 2-core i5-6300U CPU 2.40GHz, 8 GB RAM, 64-bit Windows 7 OS

**Software and Tools used:**

Azure Portal, NodeJs, VSCode.

**YouTube Links:**

2 Min:- <https://youtu.be/BaaeLt4cOe0>

15 Min:- <https://youtu.be/Fwynr6qPYrg>

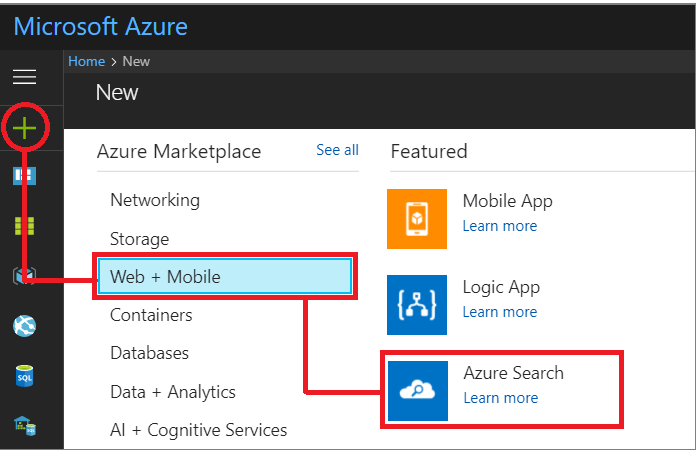
**Github link:**

<https://github.com/maha-shrestha/FinalProjectAzureSearch>

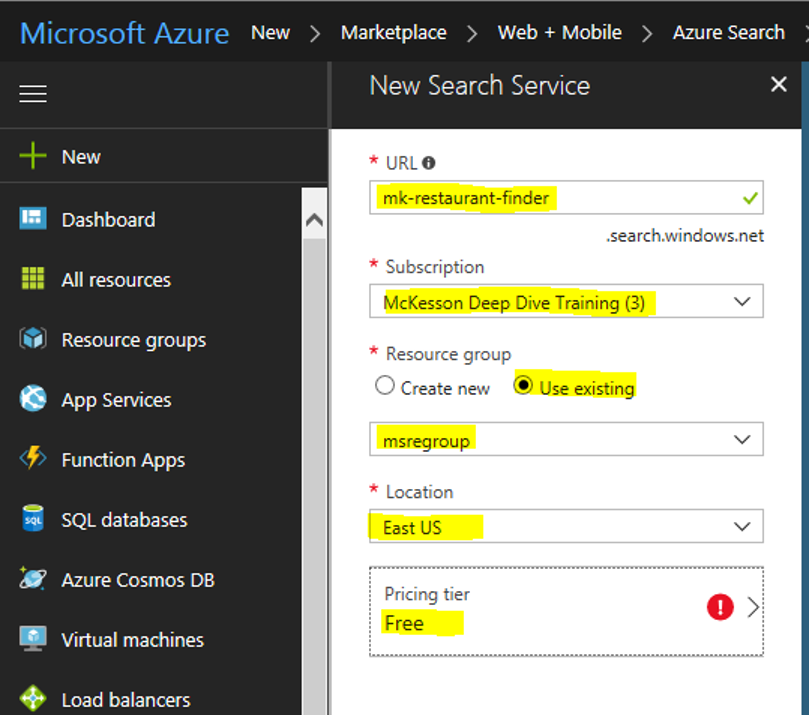
**All the step that I followed to complete my project with the screenshot are as follows:**

**Step 1:Create a Azure Search service in the Azure Portal**

* Sign in to the [Azure portal](https://portal.azure.com/).
* Click the plus sign ("+") in the top left corner.
* Select Web + Mobile > Azure Search.

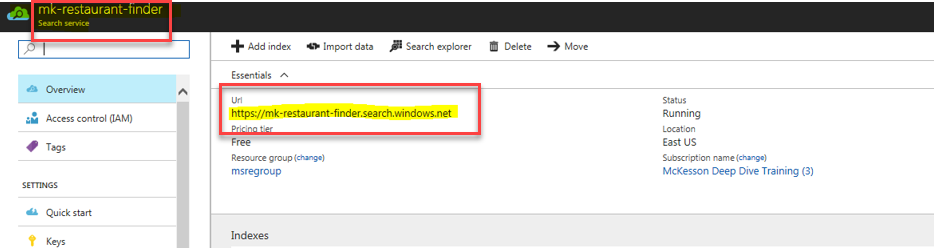


* Enter the service name in the **URL** field.
* Select a subscription
* Select a resource group
* Select a host location
* Select a pricing tier



* Click Create

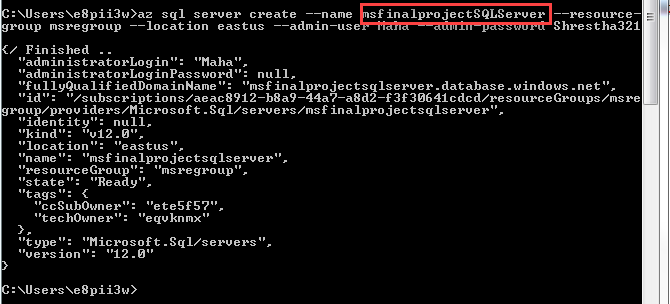
Azure Service ‘mk-restaurant-finder’ is created .



**Step 2: Create Azure SQL Database.**

* Run the following CLI command to create ‘msfinalprojectSQLServer‘ sql server

az sql server create --name msfinalprojectSQLServer --resource-group msregroup --location eastus --admin-user Maha --admin-password Shrestha321



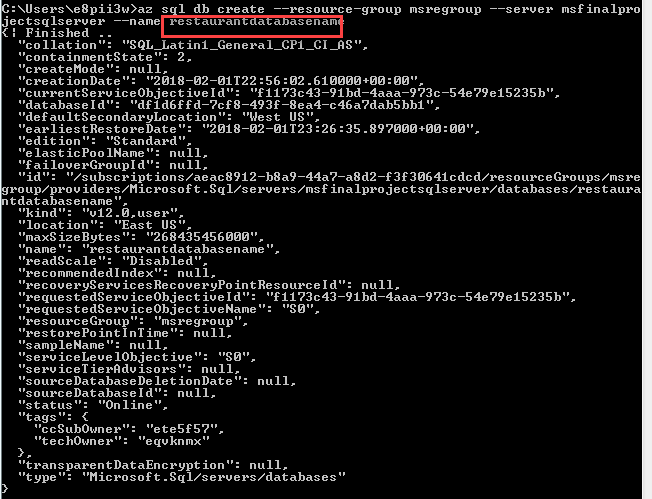
* Run the following CLI command to create firewall-rule.

az sql server firewall-rule create --resource-group msregroup --server msfinalprojectsqlserver -n AllowYourIp --start-ip-address 0.0.0.0 --end-ip-address 255.255.255.255

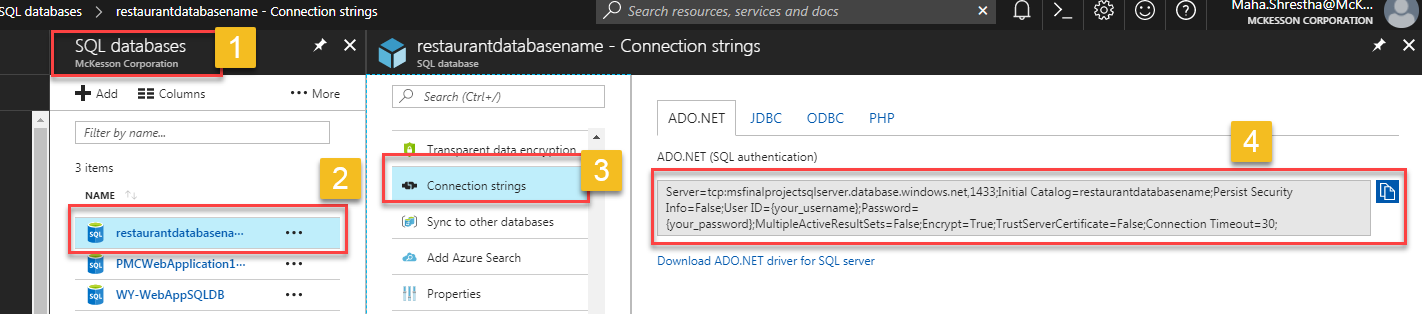


* Run the following CLI command to create ‘restaurantdatabasename‘ database

az sql db create --resource-group msregroup --server msfinalprojectsqlserver --name restaurantdatabasename

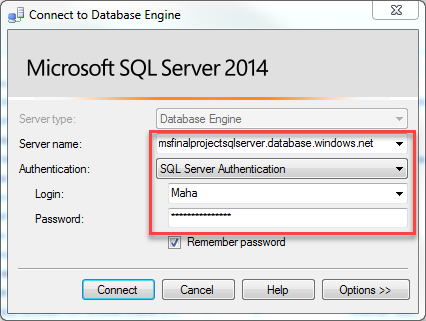


* Verify the SQL Database at Portal side and get the connection String and save it for later use

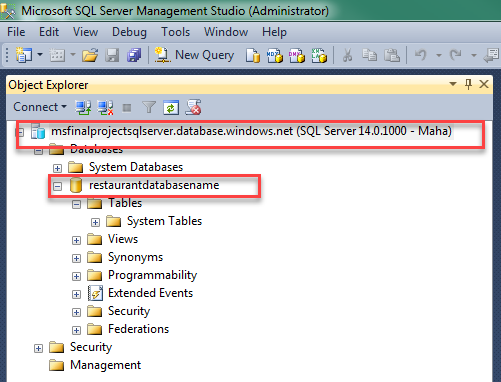


**Step 3: Load test data on to the Table**

* Use SSMS to connect to ‘restaurantdatabasename’ Database.



* Select the Server name
* Provider user login id
* Provider password and click connect



* Run the following query to create a table ‘restaurantdatacollection’

Create table dbo.restaurantdatacollection

(

id int not null,

name varchar(100),

descriptions varchar(250),

isOpen char(1),

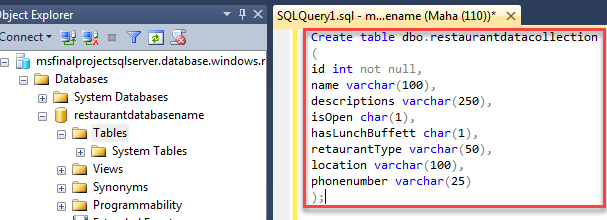
hasLunchBuffett char(1),

retaurantType varchar(50),

location varchar(100),

phonenumber varchar(25)

);



* Run the insert query to load some test data on the table

insert dbo.restaurantdatacollection values ('1','Vietnam Restaurant','Relaxed eatery with a large menu of classic Vietnamese soups, sandwiches, salads & entrees','Y','N', 'Vietnamese','1615 Commerce Parkway ,Bloomington,IL,61704','8154561234');

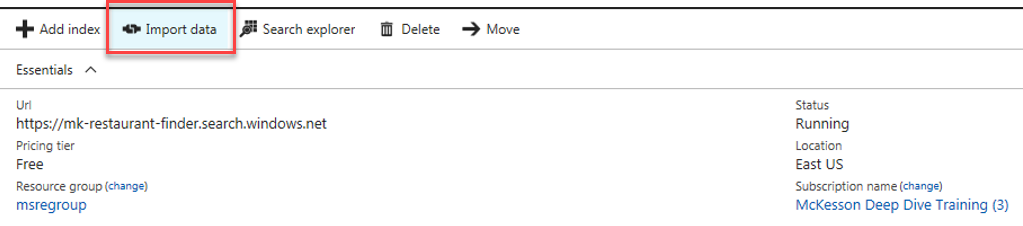
insert dbo.restaurantdatacollection values ('2','Chops and Ribs','A vast menu & thoughtful beer list is offered in an airy dining room with scenic riverside seating.','Y','N','Stakehouse','71 S. Wacker Drive Chicago,Chicago,IL,60606','3125466177');

insert dbo.restaurantdatacollection values ('3','Molinarinari','Delicious food for the Price you pay','Y','N','Mexian','1701 River Drive, Suite 306 Moline,IL,61265','3124567890');

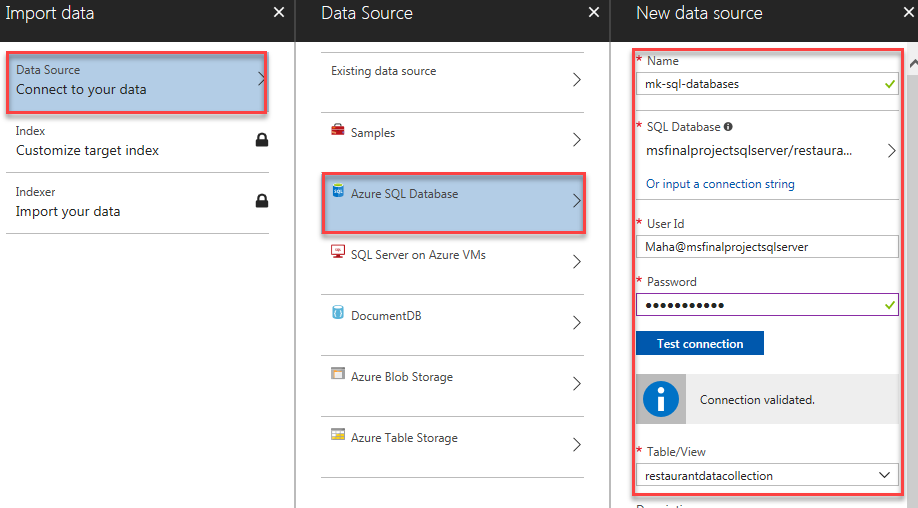
insert dbo.restaurantdatacollection values ('4','Siam Cusine','Casual, Brooklyn-based pizzeria chain serving brick-oven pies & calzones plus wine & beer','Y','N','Pizza','3201 W White Oaks Drive, Suite 204,Springfield,IL,62704','8154561237');

**Step 4: Import data and create index to Azure Search using the portal**

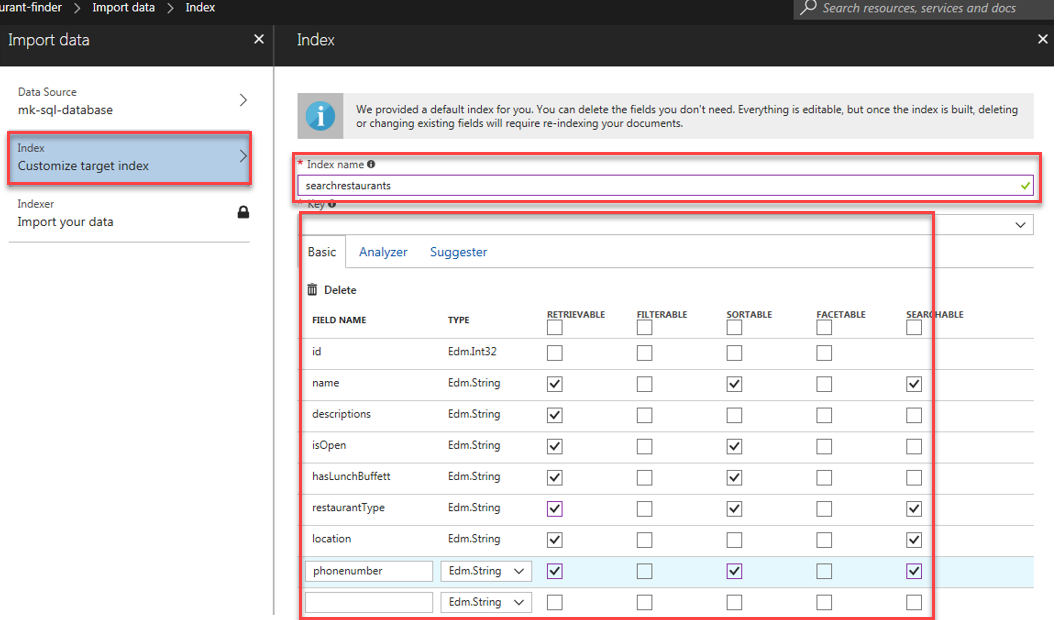
* Select **Import data** wizard on the Azure Search



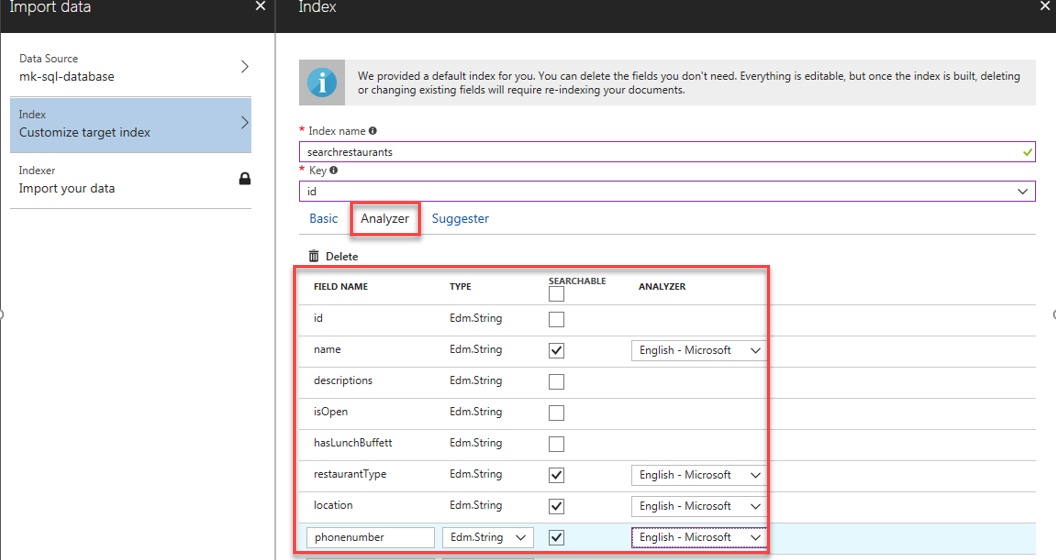
* Click **Connect to your data** to specify a data source definition on Import Data blade
* Select Azure SQL Database
* Give a name to the datasource
* Select Service name,
* Provide the credentials for a database
* Test the connection
* Select the table



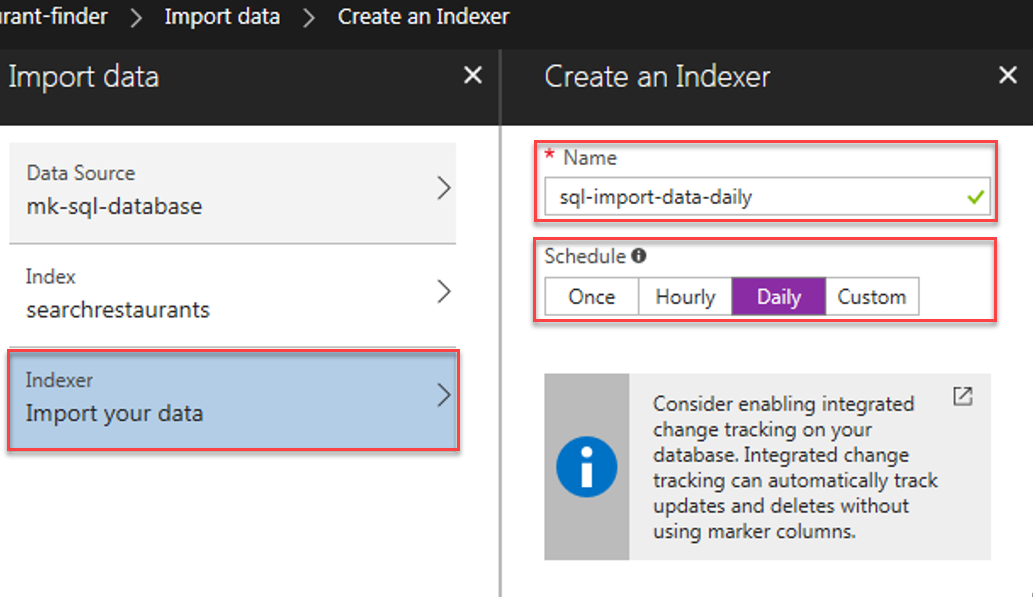
* In **Customize target index**, specify the name and a **Key** used to uniquely identify each document
* Set index attributes for each field



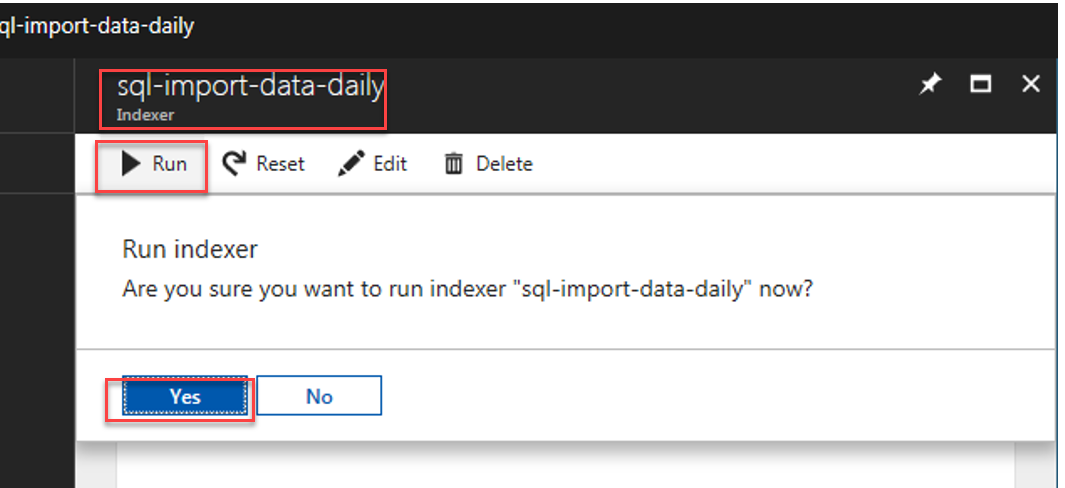
* Click on Analyzer and change the English - Microsoft and click OK



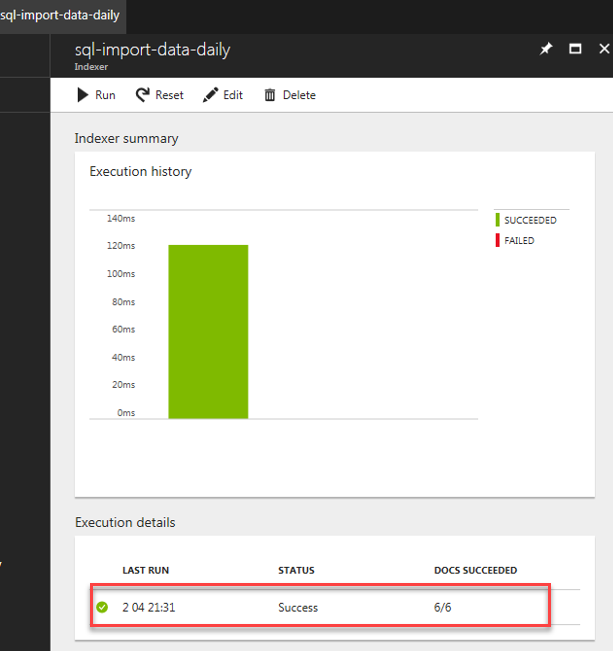
* Provide a name for the indexer in **Import your data**.
* Specify the schedule
* Click ok



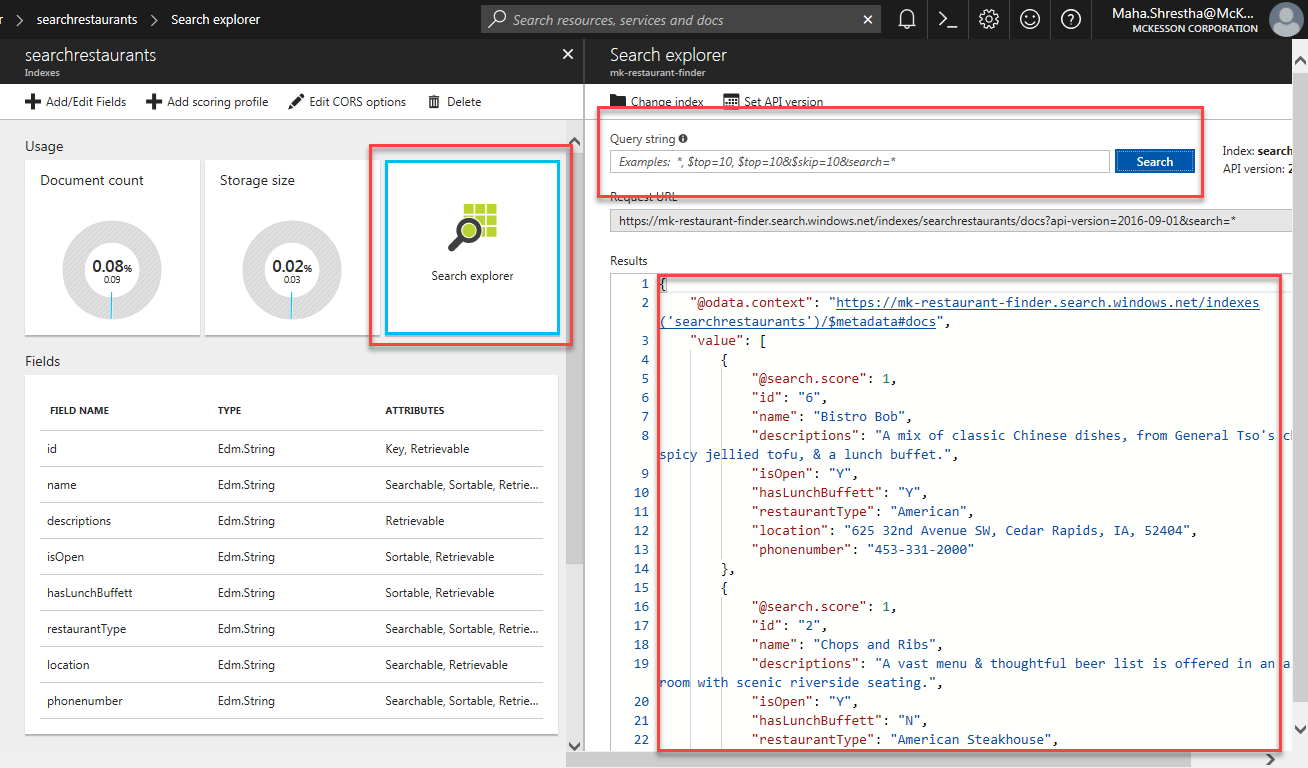
* Open the indexer and click Run to import the Data



* After data Import completes, we can monitor the indexing.

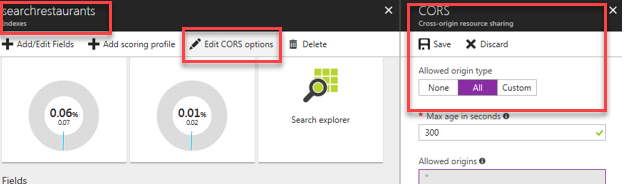


* Using Search Explorer in the Azure Portal we can Query an Azure Search index
* Click on the Search Explorer tile to slide open the search bar and results pane
* In **Query string**, type a query and then press **Search**
* In **Results**, query results are presented



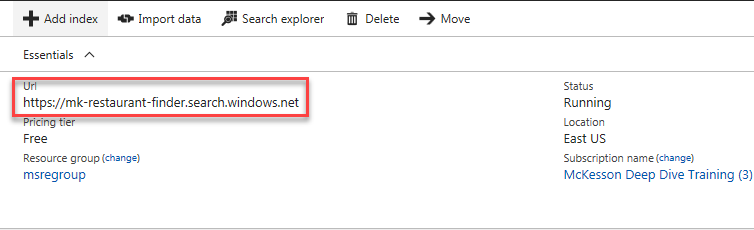
Change the CORS Setting

* Select Edit CORS options
* Select All allowed origin type and Save

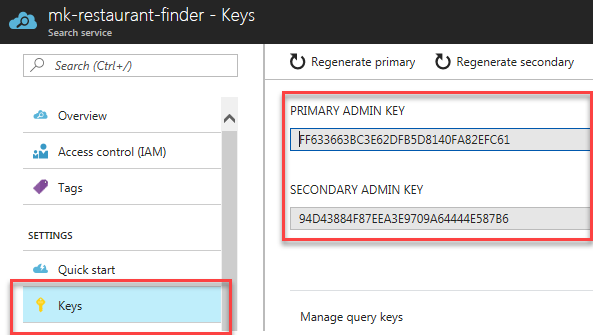


**Step 5: Find the service name and api-key of your Azure Search service**

* Copy URL form the service dashboard.



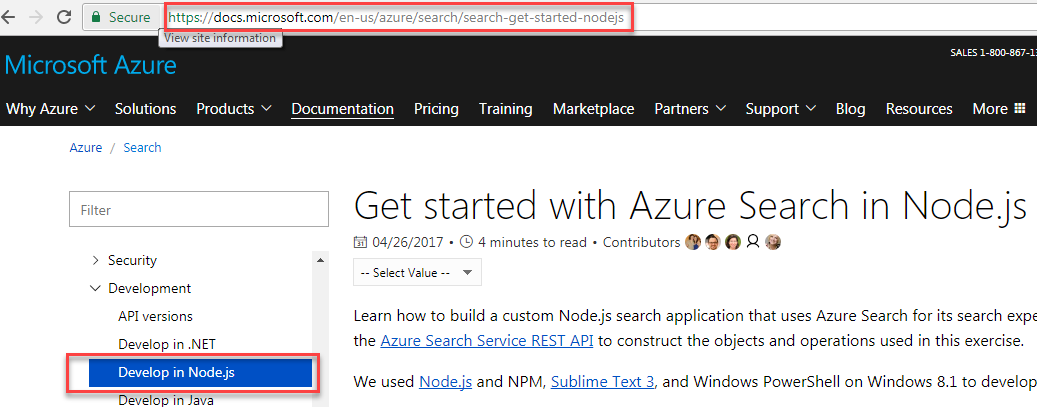
* Click on Key icon for accessing the admin keys and save it



**Step 6: Download Azure Search sample code in NODE.js**

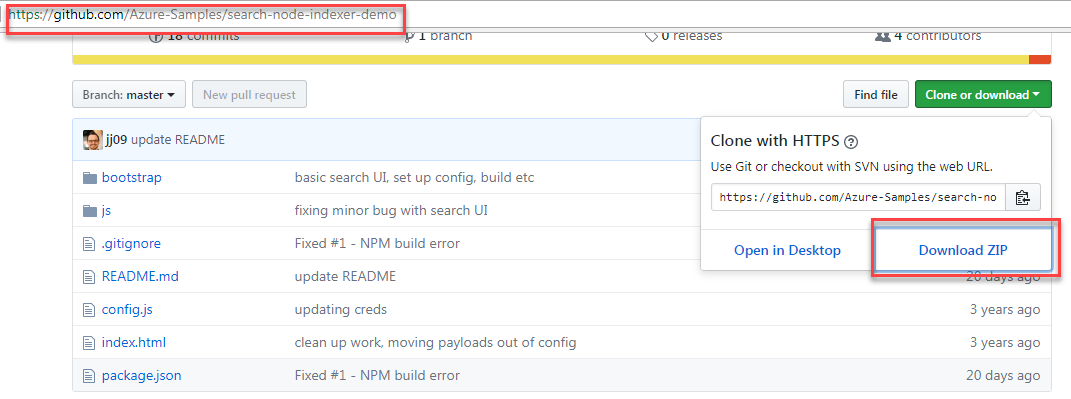
* Go to link below.

<https://docs.microsoft.com/en-us/azure/search/search-get-started-nodejs>



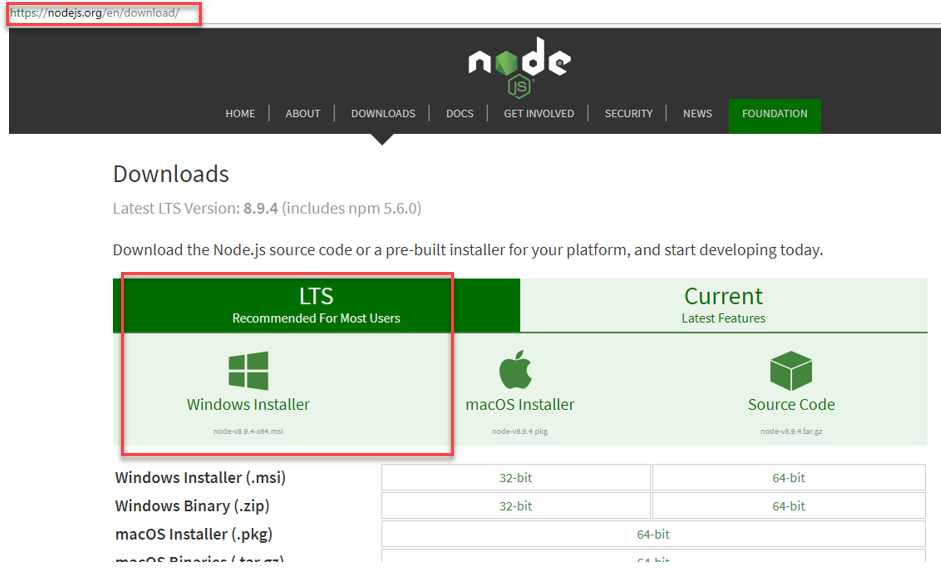
* Get the link to download the sample files

<https://github.com/Azure-Samples/search-node-indexer-demo>

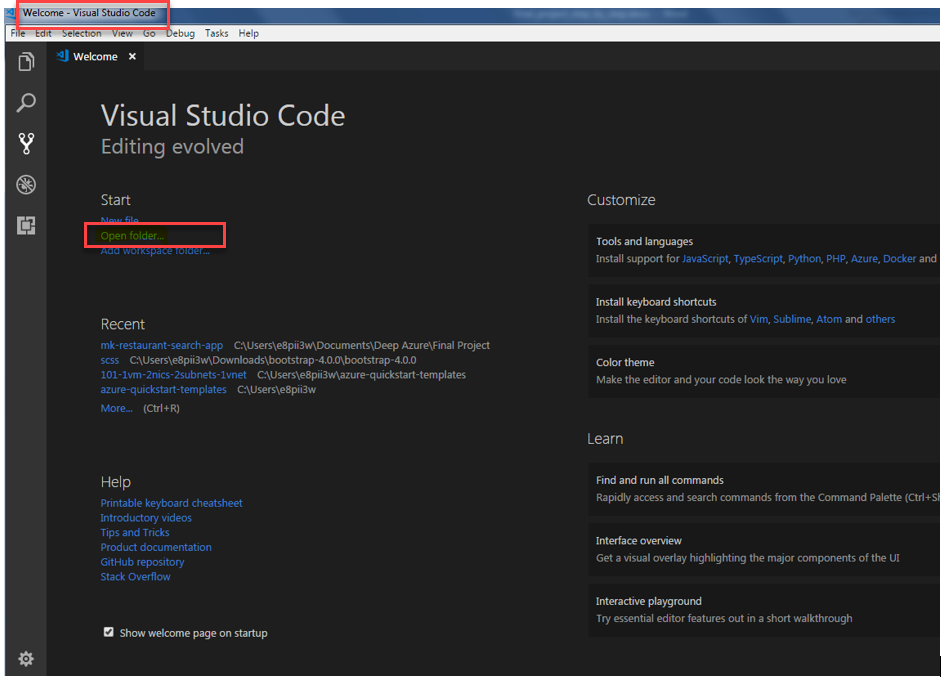


* Download and install nodejs from below link for Window

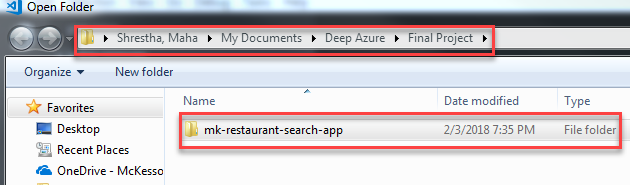
<https://nodejs.org/en/download/>



* Open the Visual Studio Code and select Open folder



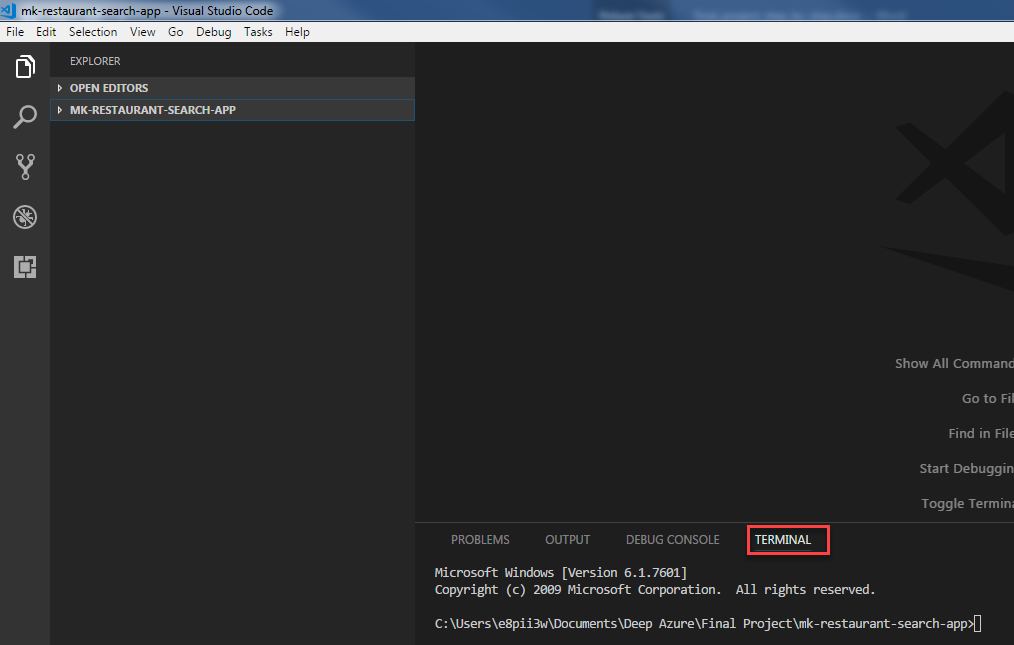
* Select the folder where we have saved the downloaded folder from github



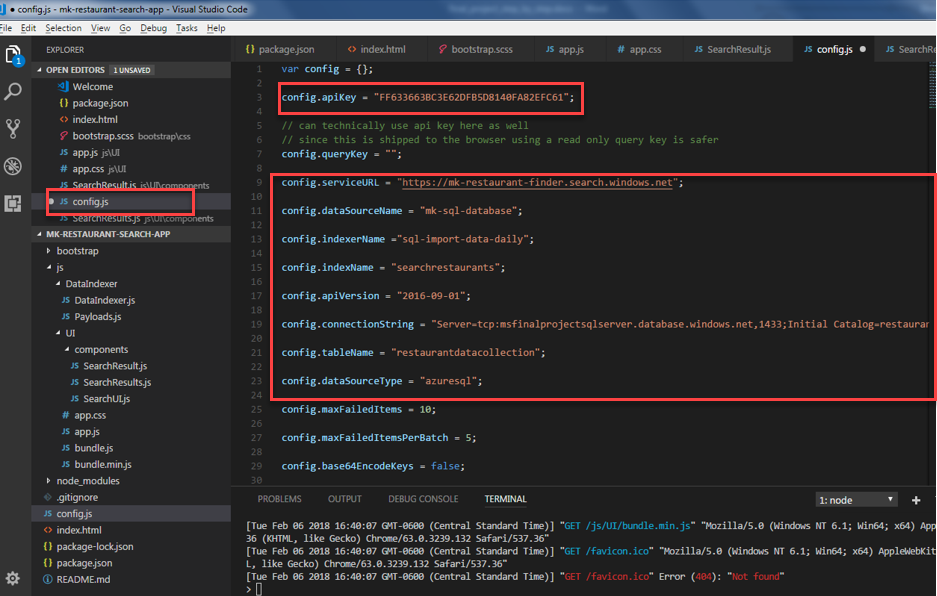
* Open the terminal and run the following command

npm Install

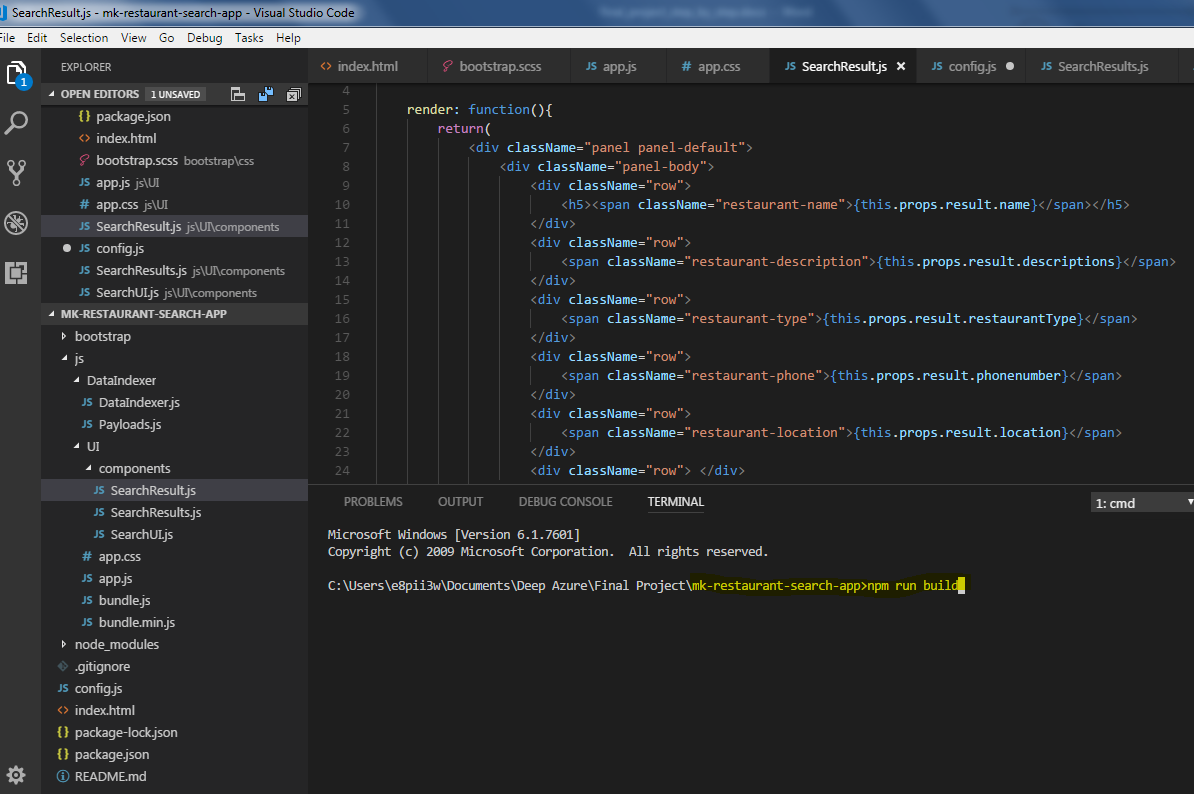
npm install -g http-server



* Select config.js file form the folder
* Update the config.apikey with PRIMARY ADMIN KEY from the Search service
* Update the config.serviceURL with url from the Search service
* Update the config.dataSourceName with Data Source Name from the Search service
* Update the config.indexerName with Indexer Name from the Search service
* Update the config.indexName with Index Name from the Search service
* Keep the config.apiVersion as it is
* Update the config.connectionString with Connection String from the SQL database we created from the Azure portal.
* Update the config.tableName with SQL table Name from the azure SQL database
* Keep the config.dataSourceType as azuresql

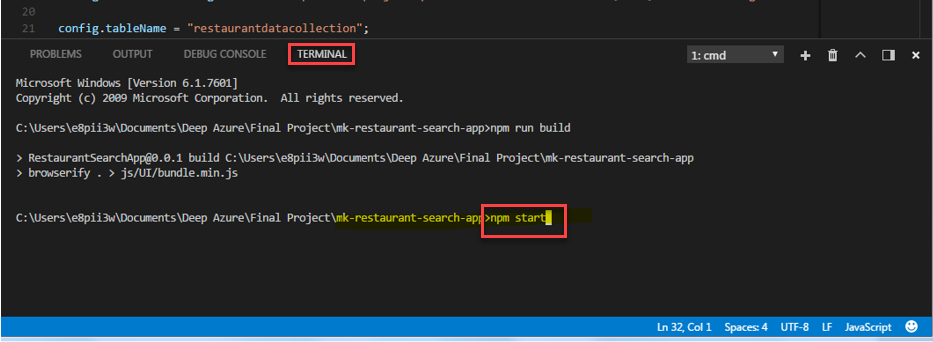


* Change the SearchResult.js compoent with the corresponding fields as a calss name



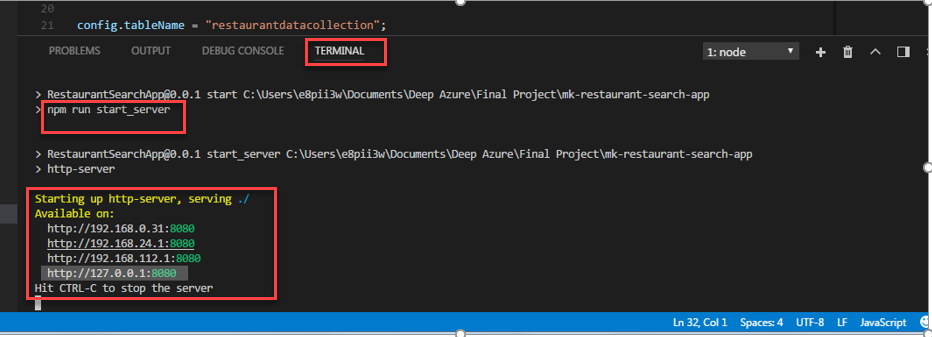
* After all modification, run the below command on terminal

npm start



* Run below command to start the server

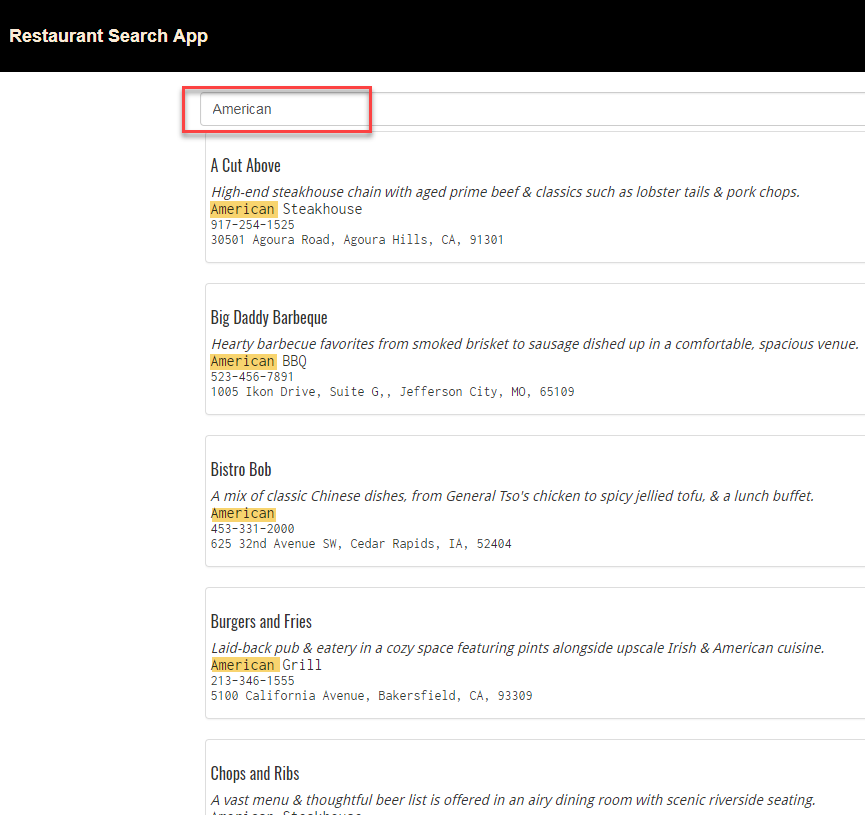
npm run start\_server



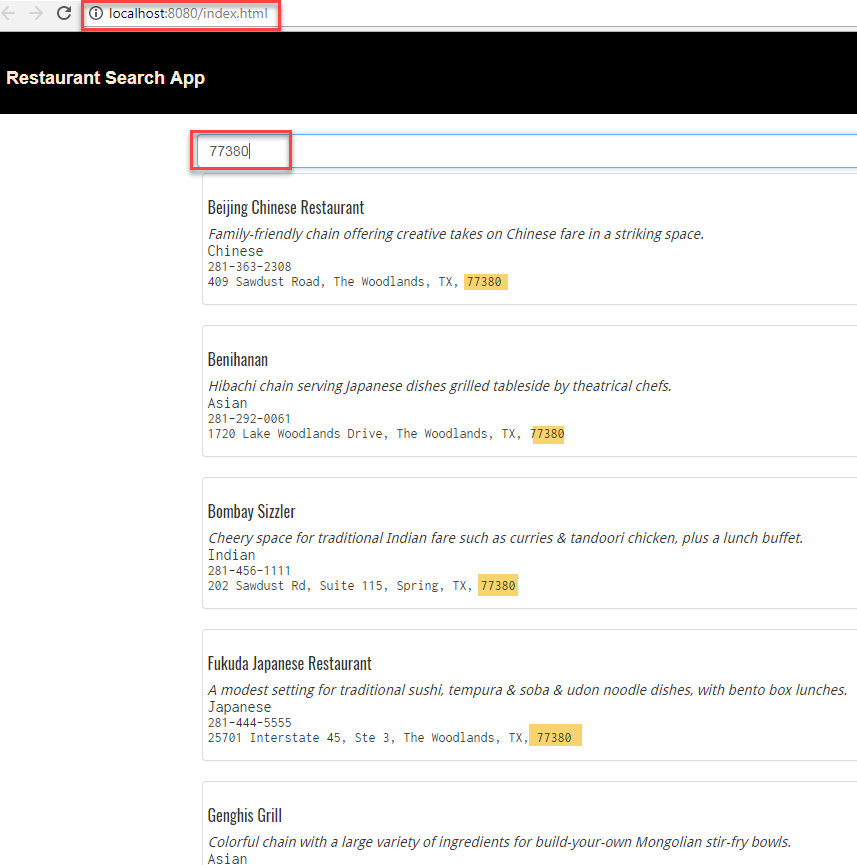
* Get one of the available http-server and open it on web browser
* On the search bar type what you are looking for and click search



* For example this application is bulild for the restaurant rearch.
* So I entered American in the serarch bar and clicked search and it displayed below result which are abaliable on my Azuer SQL table.



* Similarly we can select restaurant avaliable by Zipcode as shown below



**Pros:**

* Easy to setup Azure services through portal
* Easy to integrate to the app with search box UI
* Very straightforward to add the searchable and queryable fields
* Ability to import data instantly or schedule it.
* No need to install and host server like SQL
* Search service is faster than most others
* Able to provide geo-spatial support, search suggestion and add search profiles

**Cons:**

* Existing fields cannot be changed or deleted, any schema updates would require re-indexing

**References:**

**https://docs.microsoft.com/en-us/azure/search/search-what-is-azure-search**