Assignment 7 June 2022

NAME: ASHOK KUMAR ROLL NUMBER: DXC-262-AB-1233

BATCH: DXC-262-ANALYTICS-B12-AZURE COMPANY: DXC TECHNOLOGY

EMPLOYEE DOMAIN: AZURE ANALYTICS TRAINING UNDER: MANIPAL PRO LEARN

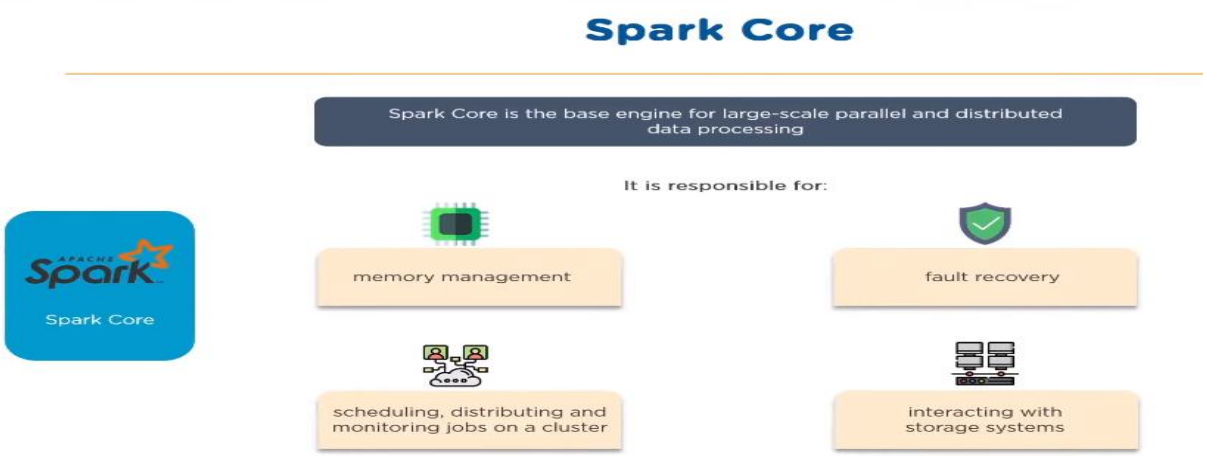
TRAINER NAME: MR. AJAY KUMAR DATE OF SUBMISSION: 7 june 2022

NO. OF Questions: 10

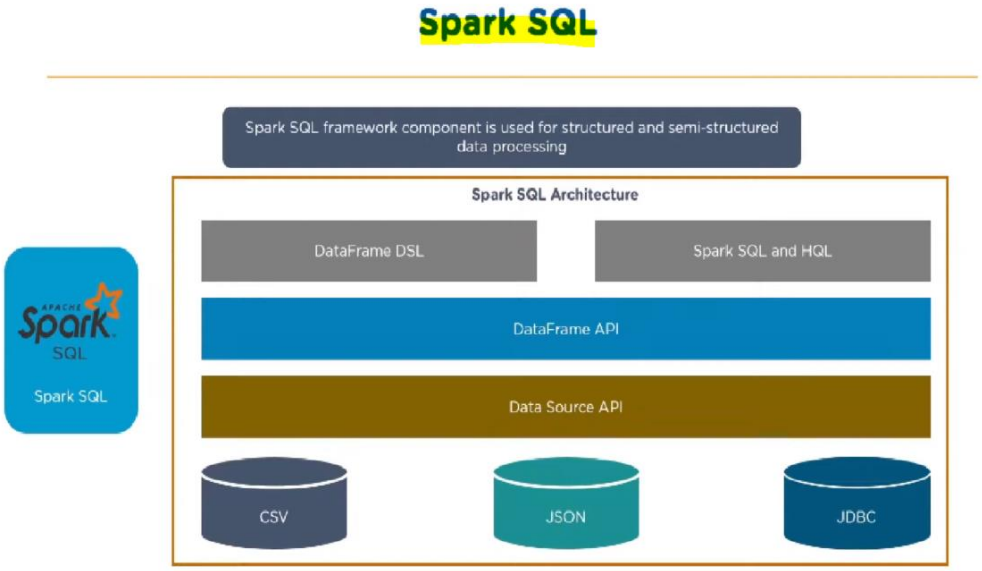
QUESTION 1. Explain what are various components of SPARK with block diagram? explain functionality of every components?

ANSWER: Components of spark:

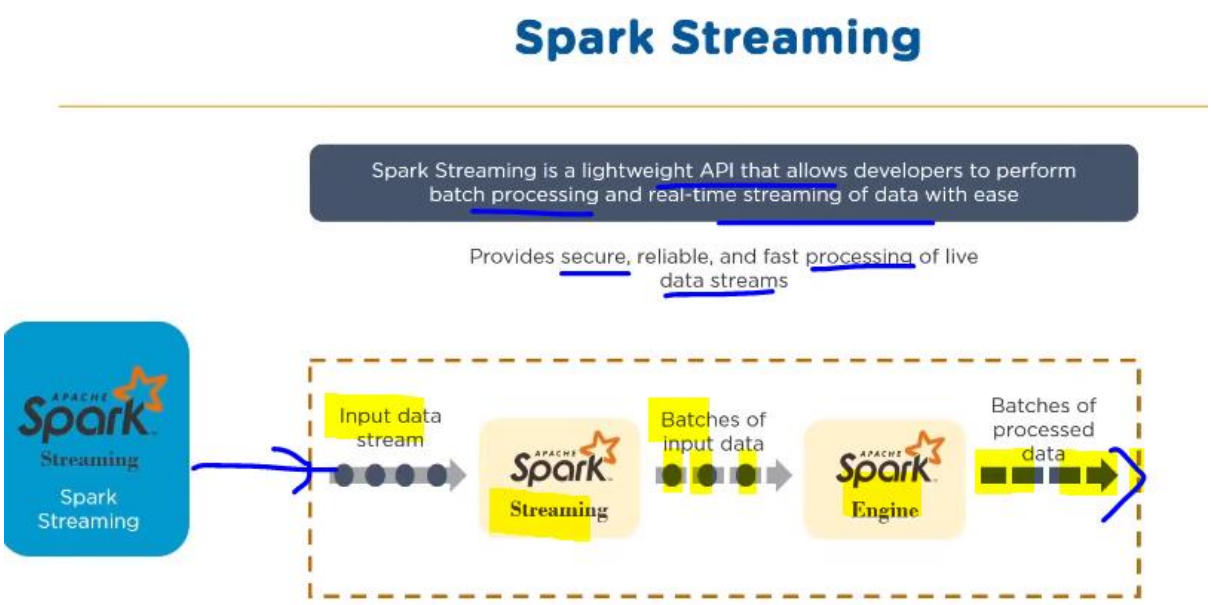
1. Spark Core: All the functionalities being provided by Apache Spark are built on the highest of the Spark Core. It delivers speed by providing in-memory computation capability. Spark Core is the foundation of parallel and distributed processing of giant dataset. It is the main backbone of the essential I/O functionalities and significant in programming and observing the role of the spark cluster. It holds all the components related to scheduling, distributing and monitoring jobs on a cluster, Task dispatching, Fault recovery. The functionalities of this component are:
    - It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
    - Home to API that defines RDDs.



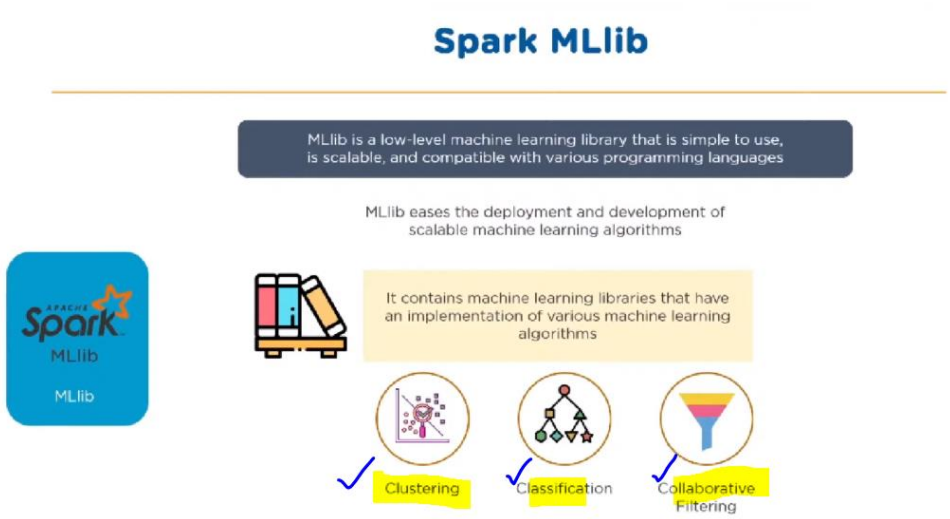
2. Spark SQL Structured data: The Spark SQL component is built above the spark core and used to provide the structured processing on the data. It provides standard access to a range of data sources. It includes Hive, JSON, and JDBC. It supports querying data either via SQL or via the hive language. This also works to access structured and semi-structured information. It also provides powerful, interactive, analytical application across both streaming and historical data. Spark SQL could be a new module in the spark that integrates the relative process with the spark with programming API. The main functionality of this module is:
    - It is a Spark package for working with structured data.
    - It Supports many sources of data including hive tablets, parquet, json.
    - It allows the developers to intermix SQK with programmatic data manipulation supported by RDDs in python, scala and java.



3. Spark Streaming: Spark streaming permits ascendible, high-throughput, fault-tolerant stream process of live knowledge streams. Spark can access data from a source like a flume, TCP socket. It will operate different algorithms in which it receives the data in a file system, database and live dashboard. Spark uses Micro-batching for real-time streaming. Micro-batching is a technique that permits a method or a task to treat a stream as a sequence of little batches of information. Hence spark streaming groups the live data into small batches. It delivers it to the batch system for processing. The functionality of this module is:
    - Enables processing of live streams of data like log files generated by production web services.
    - The API's defined in this module are quite similar to spark core RDD API's.

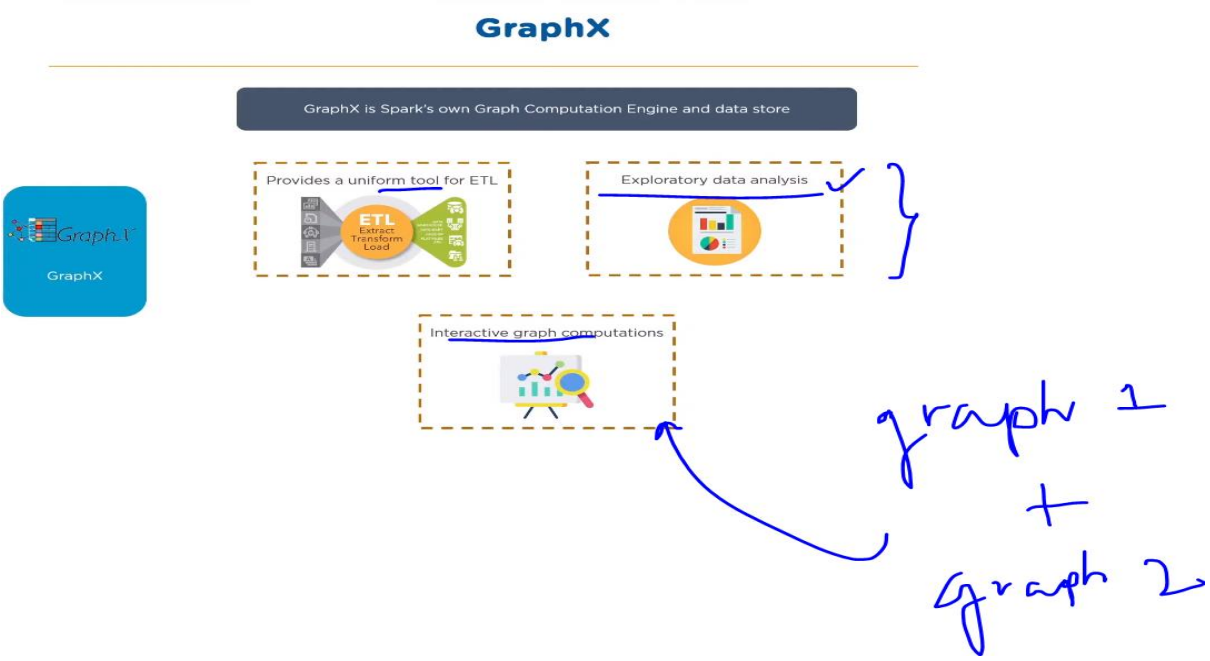


4. Mlib Machine Learning: MLlib in spark is a scalable Machine learning library that contains various machine learning algorithms. The motive behind MLlib creation is to make the implementation of machine learning simple. It contains machine learning libraries and the implementation of various algorithms. For example, clustering, regression, classification and collaborative filtering.



5. GraphX graph processing: It is an API for graphs and graph parallel execution. There is network analytics in which we store the data. Clustering, classification, traversal, searching, and pathfinding is also possible in the graph. It generally optimizes how we can represent vertex and edges in a graph. GraphX also

optimizes how we can represent vertex and edges when they are primitive data types. To support graph computation, it supports fundamental operations like subgraph, joins vertices, and aggregate messages as well as an optimized variant of the Pregel API.



Uses of Apache Spark: The main applications of the spark framework are:

1. The data generated by systems aren't consistent enough to mix for analysis. To fetch consistent information from systems we will use processes like extract, transform and load and it reduces time and cost since they are very efficiently implemented in spark.
2. It is tough to handle the time generated data like log files. Spark is capable enough to work well with streams of information and reuse operations.
3. As spark is capable of storing information in memory and might run continual queries quickly, it makes it straightforward to figure out the machine learning algorithms that can be used for a particular kind of data.

QUESTION 2. Explain Spark core in details & how RDD is related to Spark core - explain with Spark program ?

ANSWER: Spark Core is the base of the whole project. It provides distributed task dispatching, scheduling, and basic I/O functionalities. Spark uses a specialized fundamental data structure known as RDD (Resilient Distributed Datasets) that is a logical collection of data partitioned across machines. RDDs can be created in two ways; one is by referencing datasets in external storage systems and second is by applying transformations (e.g. map, filter, reducer, join) on existing RDDs.

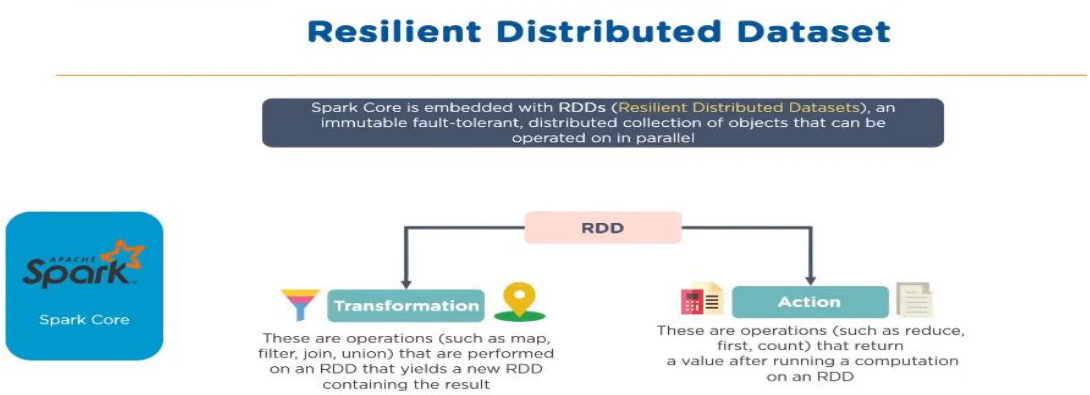
The RDD abstraction is exposed through a language-integrated API. This simplifies programming complexity because the way applications manipulate RDDs is similar to manipulating local collections of data.

- Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations.



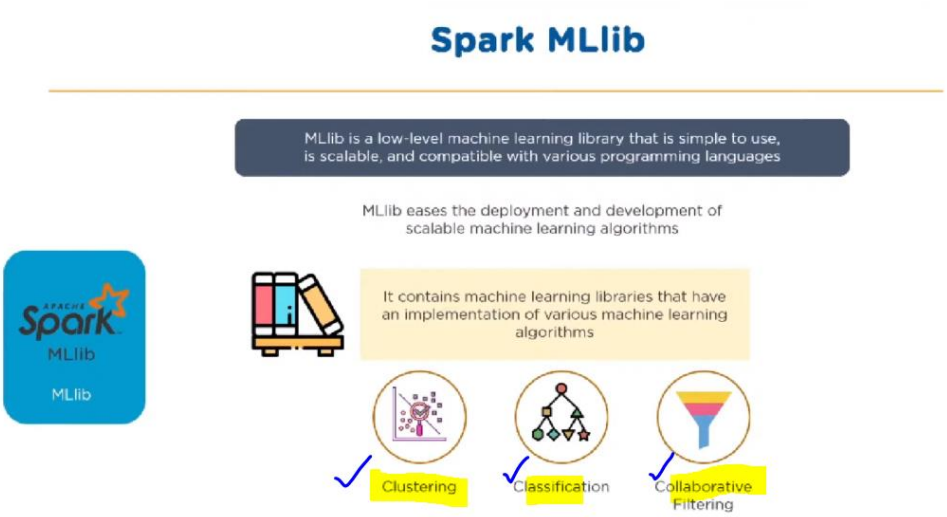
features of Apache Spark RDD are:

QUESTION 3. Explain various Mlib algorithms Spark is supporting ?

ANSWER: *Spark MLlib* is Apache Spark's Machine Learning component. One of the major attractions of Spark is the ability to scale computation massively, and that is exactly what you need for machine learning algorithms. But the limitation is that all machine learning algorithms cannot be effectively parallelized. Each algorithm has its own challenges for parallelization, whether it is task parallelism or data parallelism.

Spark MLlib is used to perform machine learning in Apache Spark. MLlib consists popular algorithms and utilities.

- Clustering
- Classification
- Collaborative filtering



Clustering: Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). So, it is the main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression and computer graphics.

Classification: Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. It is an example of pattern recognition.

Here, an example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.)

Collaborative filtering: Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. spark.MLlib currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. spark.MLlib uses the alternating least squares (ALS) algorithm to learn these latent factors.

We use the Collaborative filtering technique in Pyspark for creating a recommendation system. Apache Spark ML implements alternating least squares (ALS) for collaborative filtering, a very popular algorithm for making recommendations.

Example – Movie Recommendation System
Problem Statement: To build a Movie Recommendation System which recommends movies based on a user's preferences using Apache Spark.

Our Requirements:

- Process huge amount of data
- Input from multiple sources
- Easy to use
- Fast processing

As we can assess our requirements, we need the best Big Data tool to process large data in short time. Therefore, Apache Spark is the perfect tool to implement our Movie Recommendation System.

QUESTION 4. Explain benefits of Spark SQL & how relational data will be inserted into SPARK ?

ANSWER:

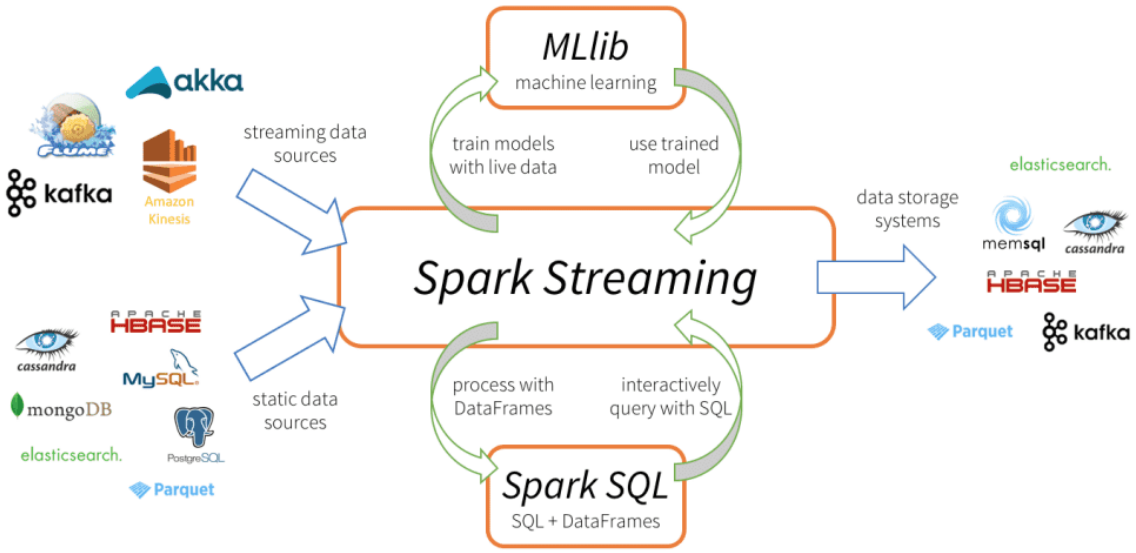
1. **Integrated**: Apache Spark SQL mixes SQL queries with Spark programs. With the help of Spark SQL, we can query structured data as a distributed dataset (RDD). We can run SQL queries alongside complex analytic algorithms using tight integration property of Spark SQL.
2. **Unified Data Access**: Using Spark SQL, we can load and query data from different sources. The Schema-RDDs lets single interface to productively work structured data. For example, Apache Hive tables, parquet files, and JSON files.
3. **High compatibility**: In Apache Spark SQL, we can run unmodified Hive queries on existing warehouses. It allows full compatibility with existing Hive data, queries and UDFs, by using the Hive fronted and MetaStore.
4. **Standard Connectivity**: It can connect through JDBC or ODBC. It includes server mode with industry standard JDBC and ODBC connectivity.
5. **Scalability**: To support mid-query fault tolerance and large jobs, it takes advantage of RDD model. It uses the same engine for interactive and long queries.
6. **Performance Optimization**: The query optimization engine in Spark SQL converts each SQL query to a logical plan. Further, it converts to many physical execution plans. Among the entire plan, it selects the most optimal physical plan for execution.

QUESTION 5. Explain Spark streaming in detail ?

ANSWER: Apache Spark Streaming is a scalable fault-tolerant streaming processing system that natively supports both batch and streaming workloads. Spark Streaming is an extension of the core Spark API that allows data engineers and data scientists to process real-time data from various sources including (but not limited to) Kafka, Flume, and Amazon Kinesis. This processed data can be pushed out to file systems, databases, and live dashboards. Its key abstraction is a Discretized Stream or, in short, a DStream, which represents a stream of data divided into small batches. DStreams are built on RDDs, Spark's core data abstraction. This allows Spark Streaming to seamlessly integrate with any other Spark components like MLlib and Spark SQL. Spark Streaming is different from other systems that either have a processing engine designed only for streaming, or have similar batch and streaming APIs but compile internally to different engines. Spark's single execution engine and unified programming model for batch and streaming lead to some unique benefits over other traditional streaming systems.

Four Major Aspects of Spark Streaming

- Fast recovery from failures and stragglers
- Better load balancing and resource usage
- Combining of streaming data with static datasets and interactive queries
- Native integration with advanced processing libraries (SQL, machine learning, graph processing)



This unification of disparate data processing capabilities is the key reason behind Spark Streaming's rapid adoption. It makes it very easy for developers to use a single framework to satisfy all their processing needs.

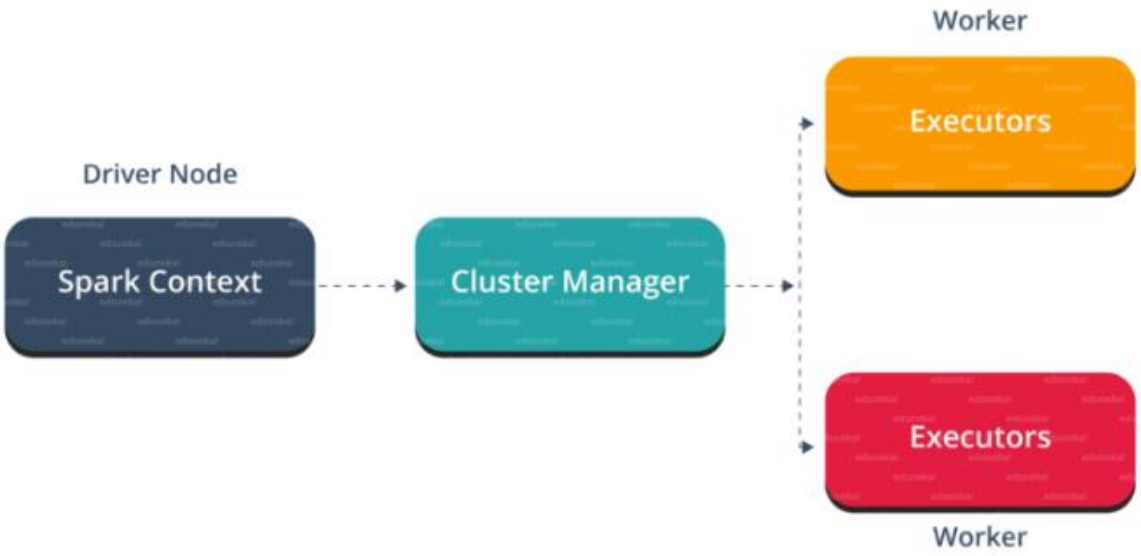
QUESTION 6. Explain SPARK architecture? what is Master - Slave architecture ?

ANSWER: Apache Spark has a well-defined layered architecture where all the spark components and layers are loosely coupled. This architecture is further integrated with various extensions and libraries.

Apache Spark Architecture is based on two main abstractions:

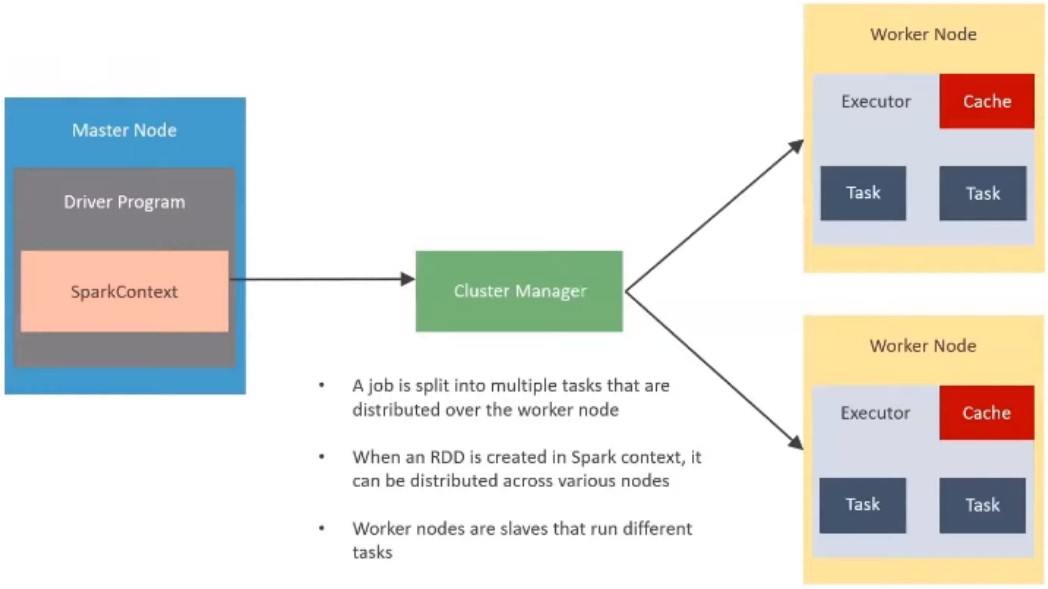
- Resilient Distributed Dataset (RDD)
- Directed Acyclic Graph (DAG)

The Apache Spark framework uses a master-slave architecture that consists of a driver, which runs as a master node, and many executors that run across as worker nodes in the cluster. Apache Spark can be used for batch processing and real-time processing as well.



In your **master node**, you have the driver program, which drives your application. The code you are writing behaves as a driver program or if you are using the interactive shell, the shell acts as the driver program.

Spark Architecture



Inside the driver program, the first thing you do is, you create a **Spark Context.** Assume that the Spark context is a gateway to all the Spark functionalities. It is similar to your database connection. Any command you execute in your database goes through the database connection. Likewise, anything you do on Spark goes through Spark context.

Now, this Spark context works with the **cluster manager** to manage various jobs. The driver program & Spark context takes care of the job execution within the cluster. A job is split into multiple tasks which are distributed over the worker node. Anytime an RDD is created in Spark context, it can be distributed across various nodes and can be cached there.

Worker nodes are the slave nodes whose job is to basically execute the tasks. These tasks are then executed on the partitioned RDDs in the worker node and hence returns back the result to the Spark Context.

Spark Context takes the job, breaks the job in tasks and distribute them to the worker nodes. These tasks work on the partitioned RDD, perform operations, collect the results and return to the main Spark Context.

If you increase the number of workers, then you can divide jobs into more partitions and execute them parallelly over multiple systems. It will be a lot faster.

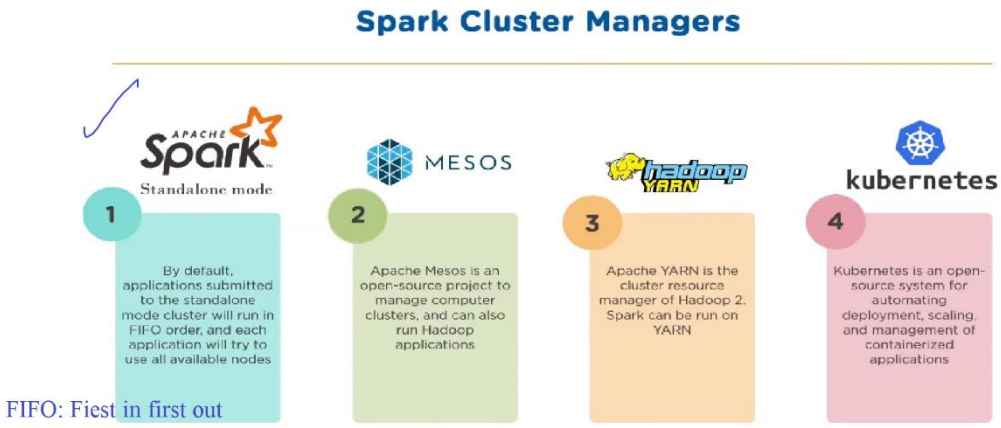
With the increase in the number of workers, memory size will also increase & you can cache the jobs to execute it faster.

QUESTION 7. Explain various cluster managers in SPARK?

ANSWER: Cluster manager is a platform (cluster mode) where we can run Spark. Simply put, cluster manager provides resources to all worker nodes as per need, it operates all nodes accordingly.
We can say there are a master node and worker nodes available in a cluster. That master nodes provide an efficient working environment to worker nodes.

Spark supports these cluster managers:

1. Standalone cluster manager
2. Mesos
3. Hadoop Yarn
4. Kubernetes

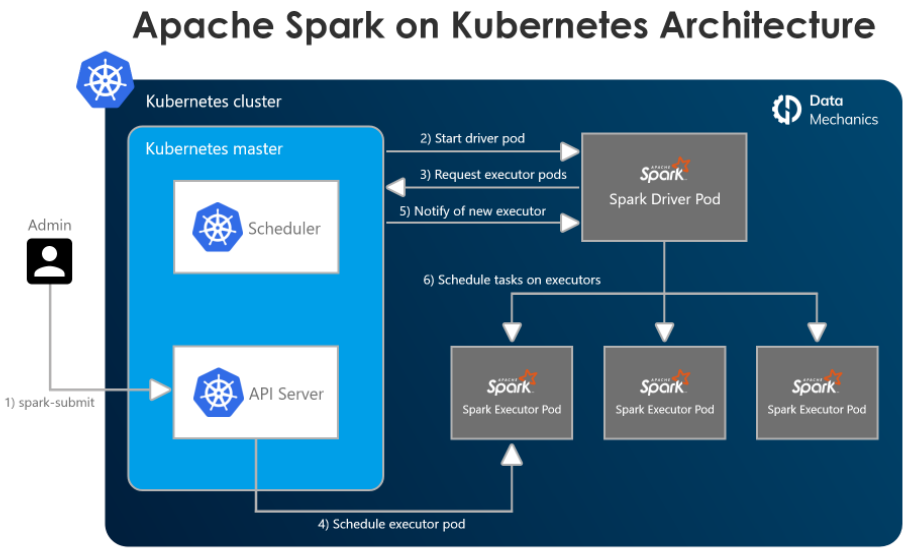


7. Standalone cluster manager: It is a part of spark distribution and available as a simple cluster manager to us. Standalone cluster manager is resilient in nature, it can handle work failures. It has capabilities to manage resources according to the requirement of applications. We can easily run it on Linux, Windows, or Mac. It can also access HDFS (Hadoop Distributed File System) data. This is the easiest way to run Apache spark on this cluster.

8. Mesos: It is a distributed cluster manager. As like yarn, it is also highly available for master and slaves. It can also manage resource per application. We can run spark jobs, Hadoop MapReduce or any other service applications easily. Apache has API's for Java, Python as well as c++. We can run Mesos on Linux or Mac OSX also.

9. Hadoop Yarn: This cluster manager works as a distributed computing framework. It also maintains job scheduling as well as resource management. In this cluster, masters and slaves are highly available for us. We are also available with executors and pluggable scheduler. We can also run it on Linux and even on windows. Hadoop yarn is also known as MapReduce 2.0. It also bifurcates the functionality of resource manager as well as job scheduling.

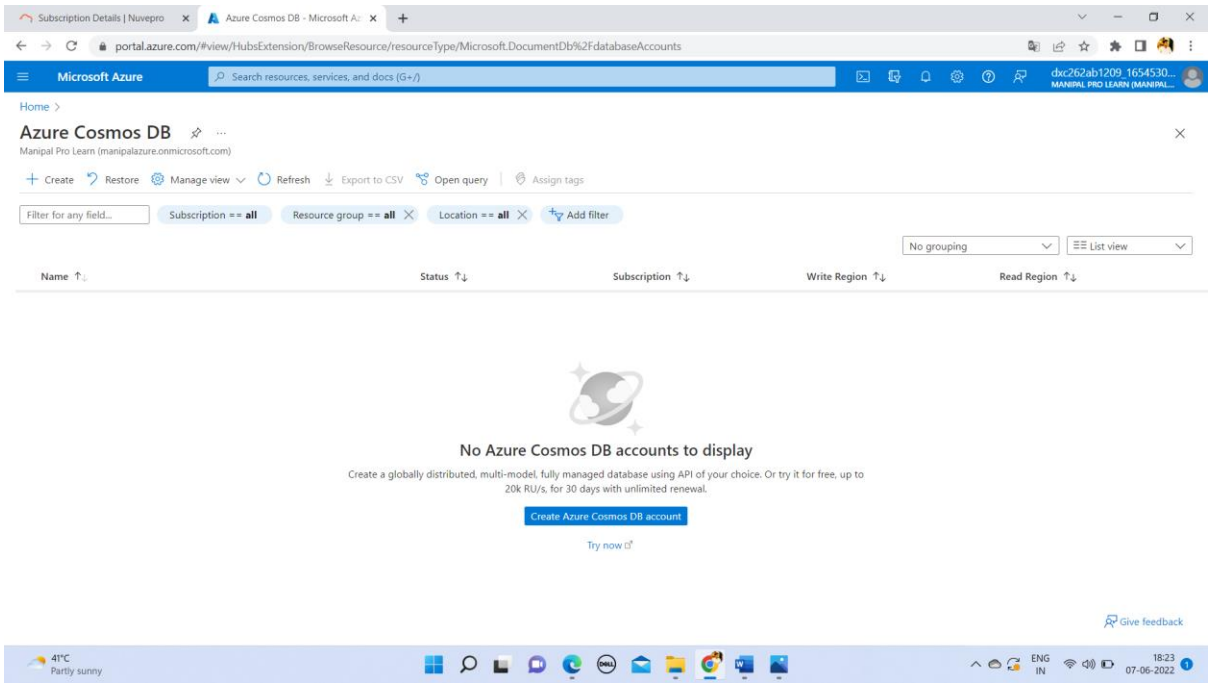
10. Kubernetes: A Kubernetes cluster consists of a set of nodes on which you can run containerized Apache Spark applications (as well any other containerized workloads). Each Spark app is fully isolated from the others and packages its own version of Spark and dependencies within a Docker image.
When you submit a Spark app, it starts a Spark driver pod (a Docker container, to put it simply) on the Kubernetes cluster. The driver pod and Kubernetes directly talk to each other to start Spark executor pods. The start and removal of executors is automated based on load when dynamic allocation is enabled.



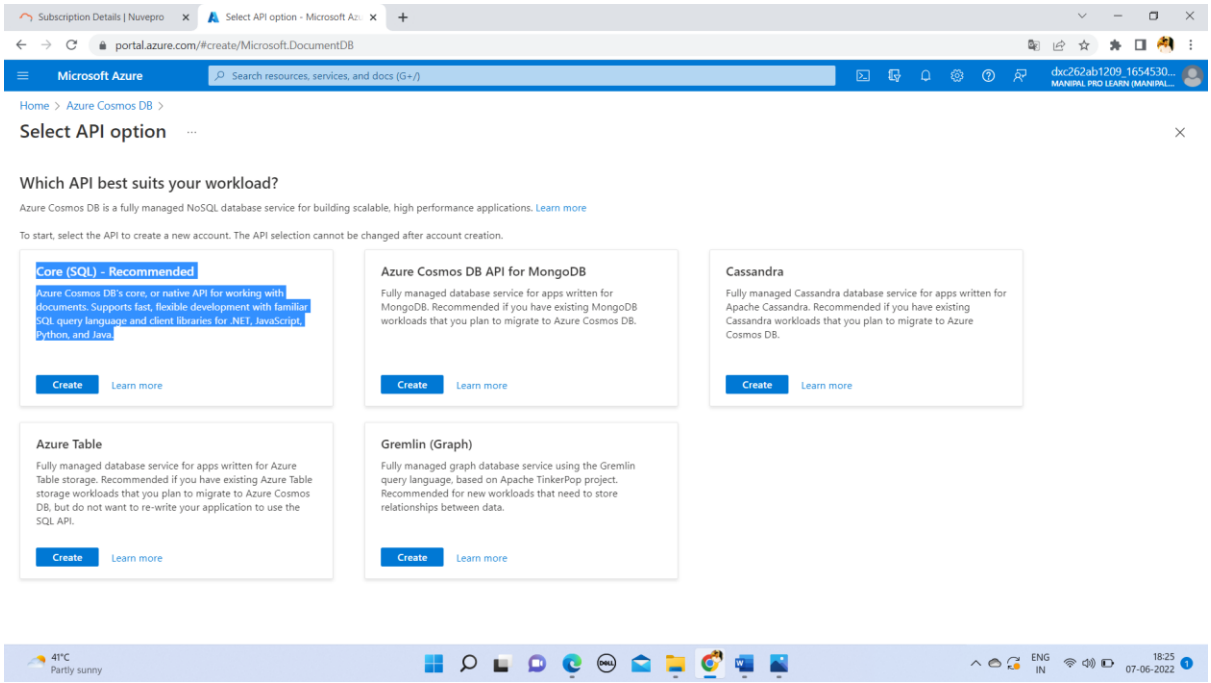
QUESTION 8. Explain with screenshots & steps how to create Cosmos DB ?

ANSWER: Go to **https://portal.azure.com/#home** and search Azure Cosmos DB and click on it.

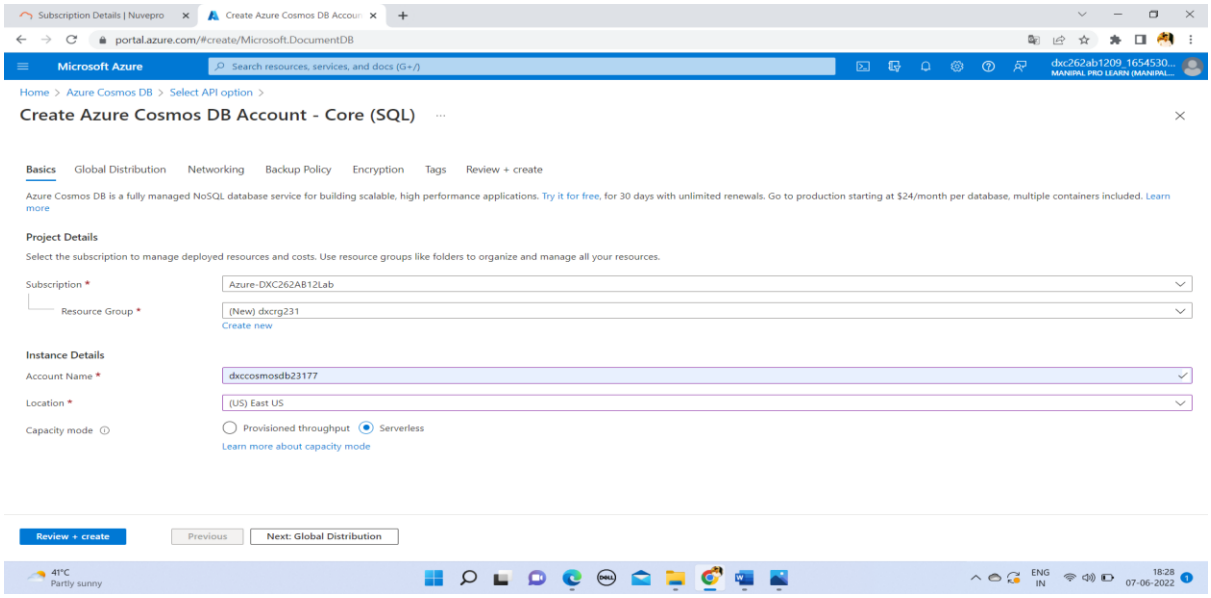
You will see this page.

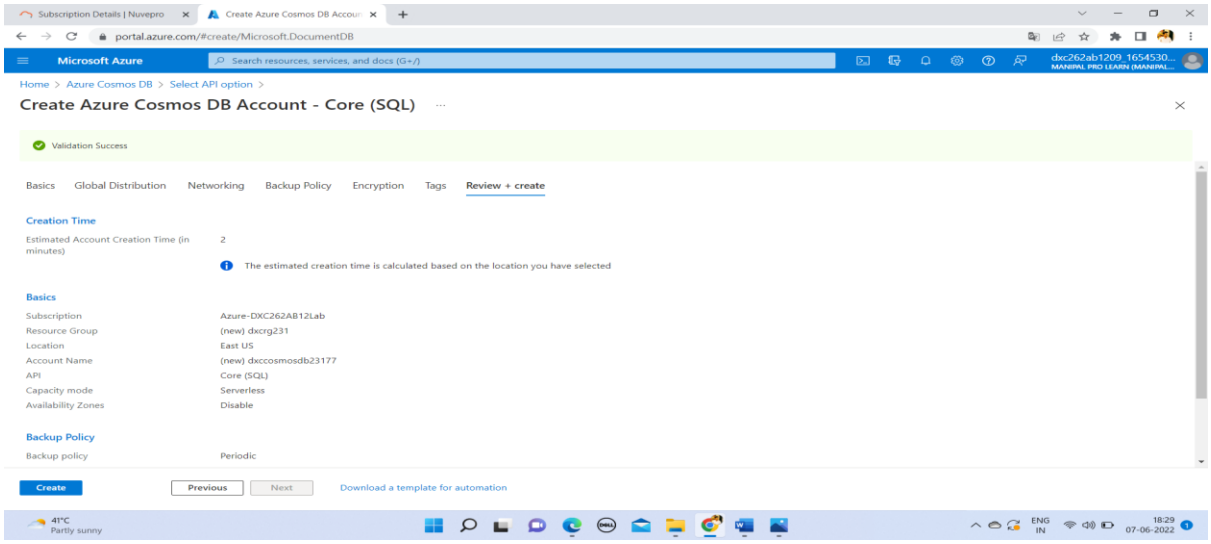


Click on create bottom.

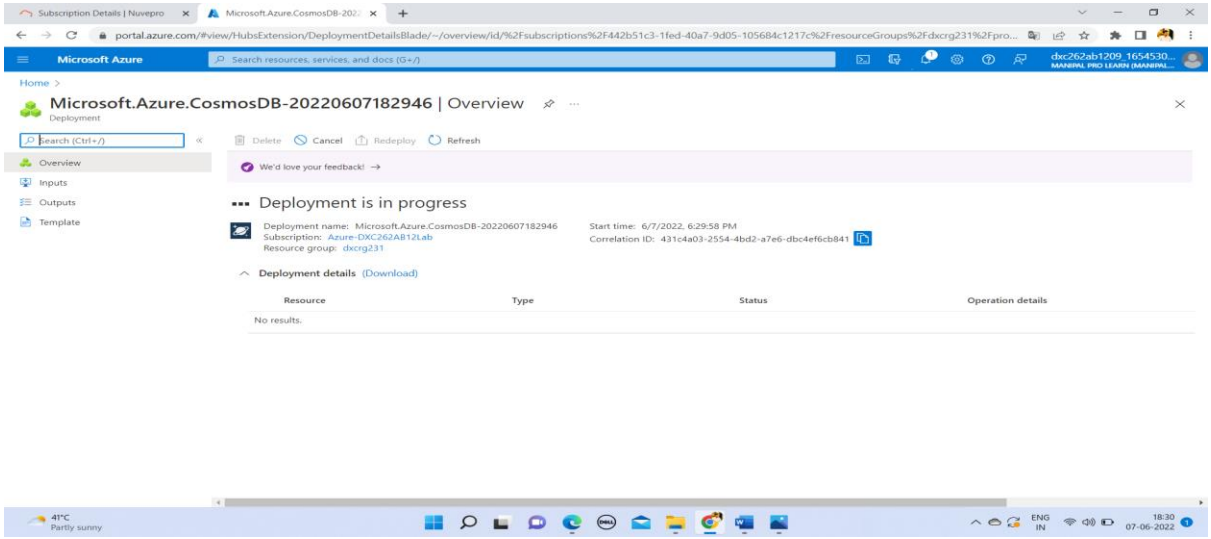


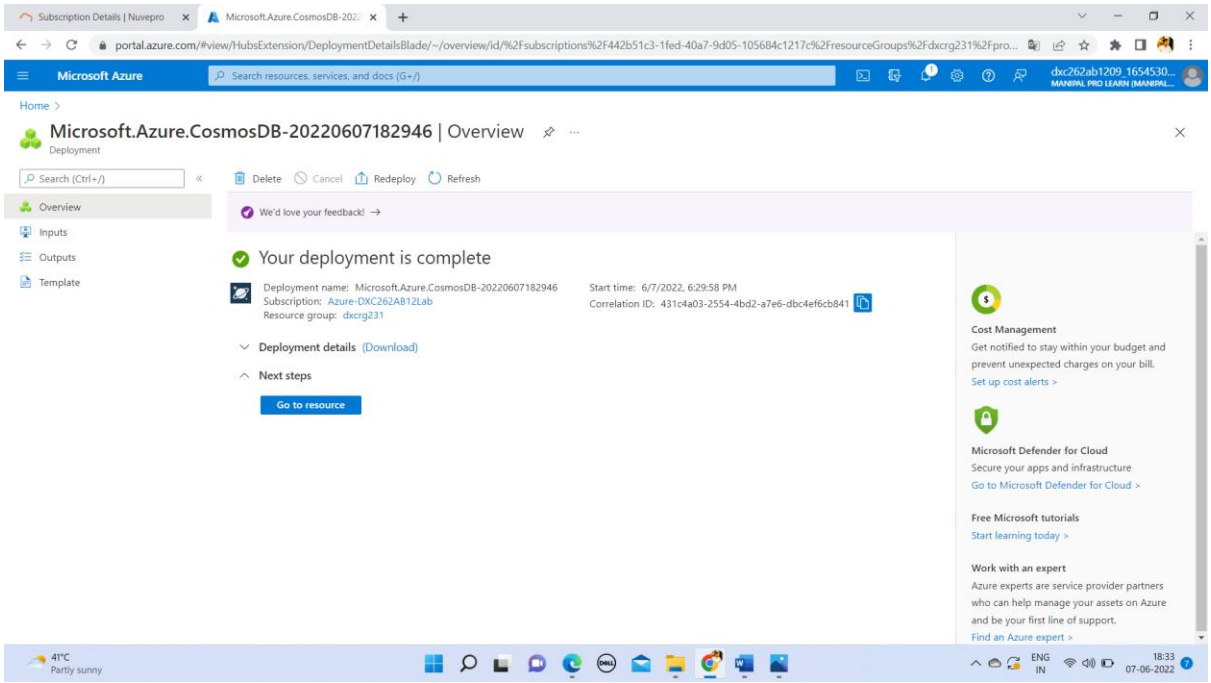
Now click on create bottom of **Core (SQL) – Recommended** then fill the required details like resource group and account name. then click on Review+Create.



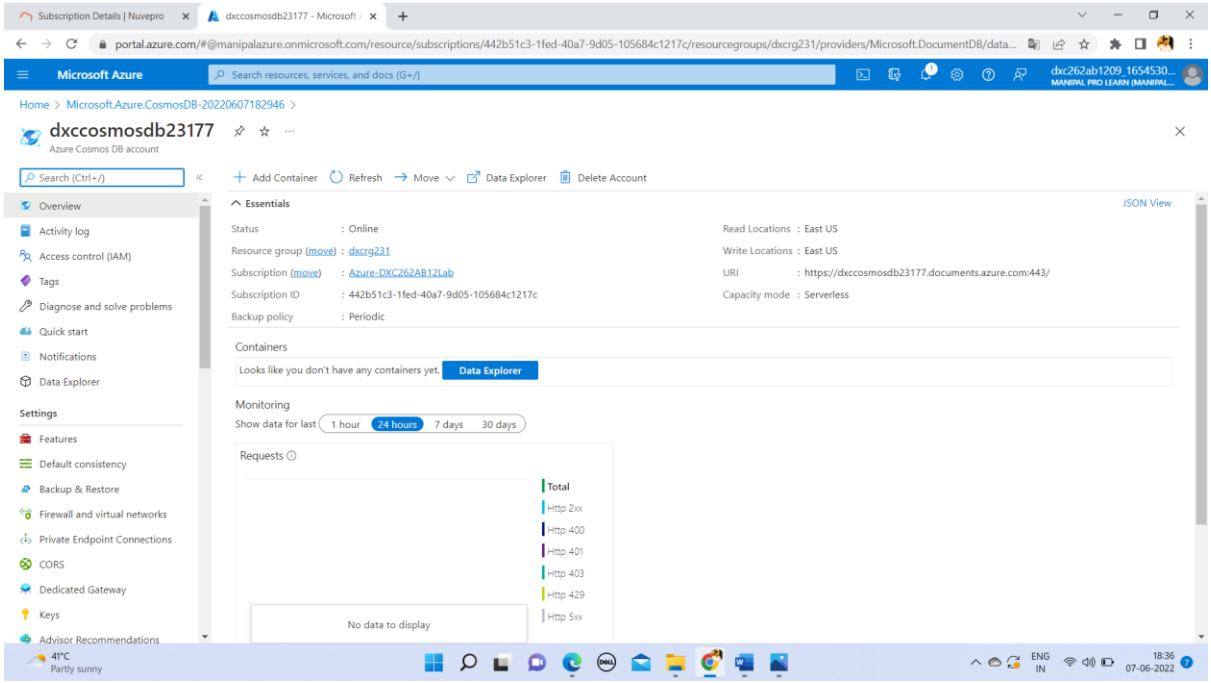


Review the account settings, and then select **Create**. It takes a few minutes to create the account. Wait for the portal page to display **Your deployment is complete**.





Click on Go to resources to go to the Azure Cosmos DB account page

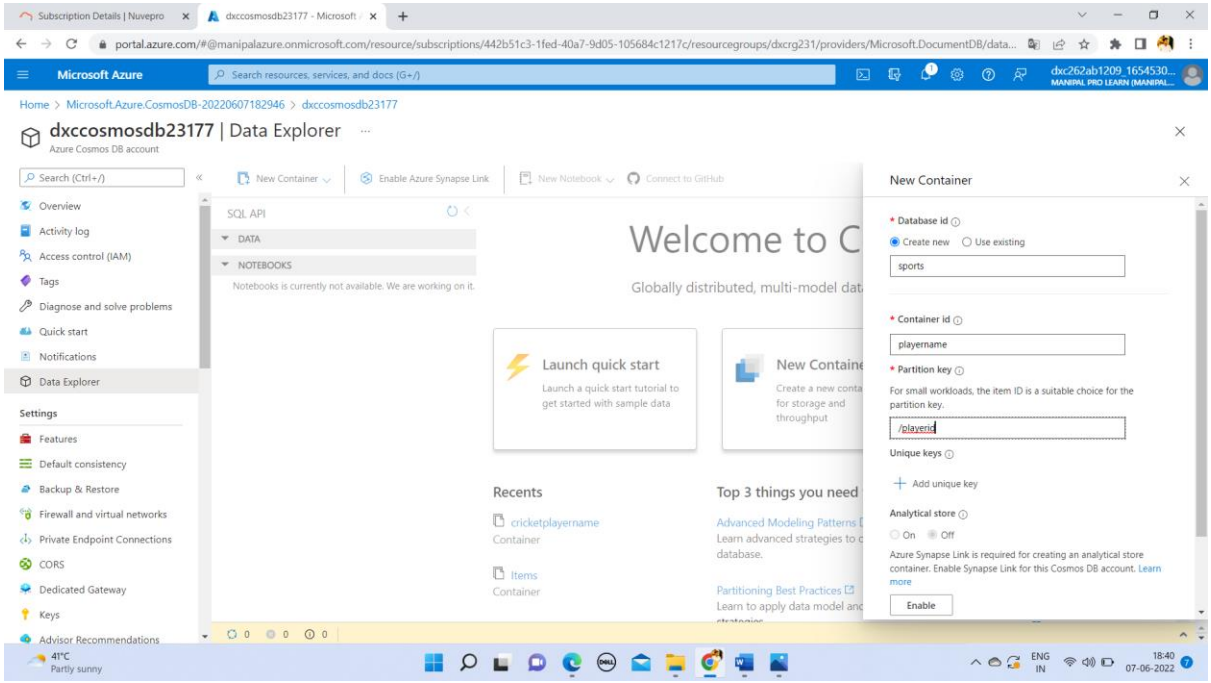


QUESTION 9. Explain with sceenshots & step how to insert data into Cosmos DB?

ANSWER:

Add a database and a container →

Click on **Data Explorer** from the left navigation on Azure Cosmos DB account page, and then click on **New Container** and fill the details for a new container.

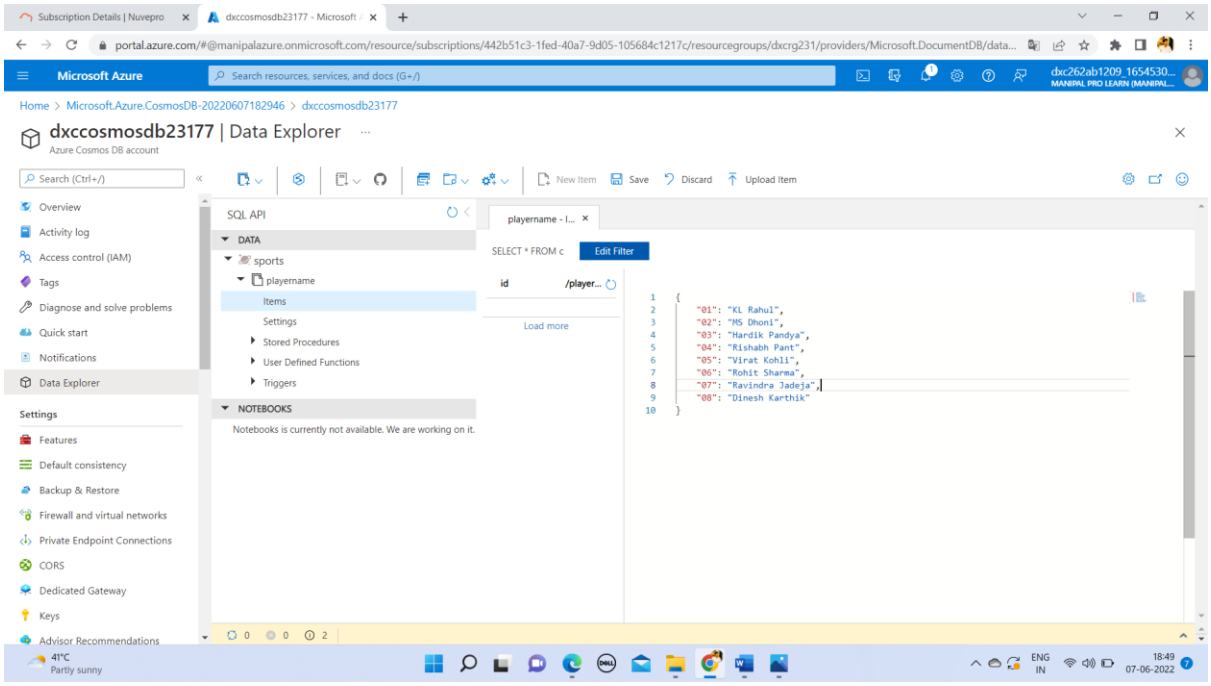


Add data to your database →

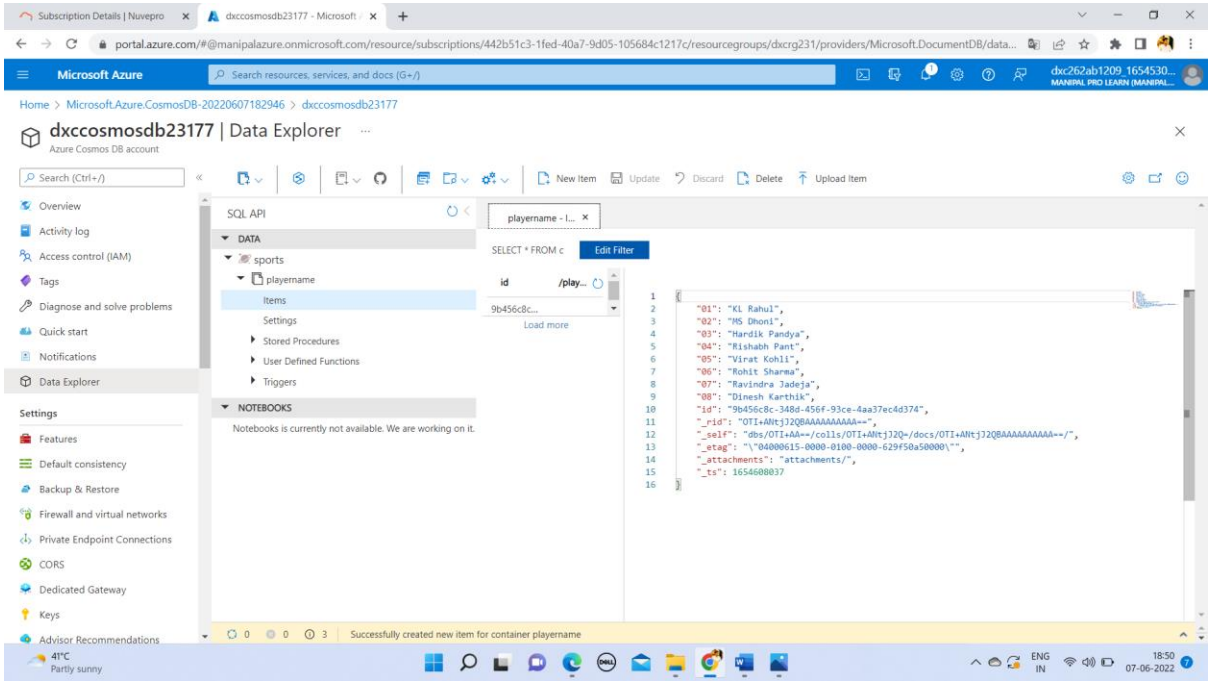
In **Data Explorer**, expand the **sports** database that we created, and expand the **Items** container. And click on **Items**, and then click on **New Item**.

And Add the following structure.

```
{
    "01": "KL Rahul",
    "02": "MS Dhoni",
    "03": "Hardik Pandya",
    "04": "Rishabh Pant",
    "05": "Virat Kohli",
    "06": "Rohit Sharma",
    "07": "Ravindra Jadeja",
    "08": "Dinesh Karthik"
}
```



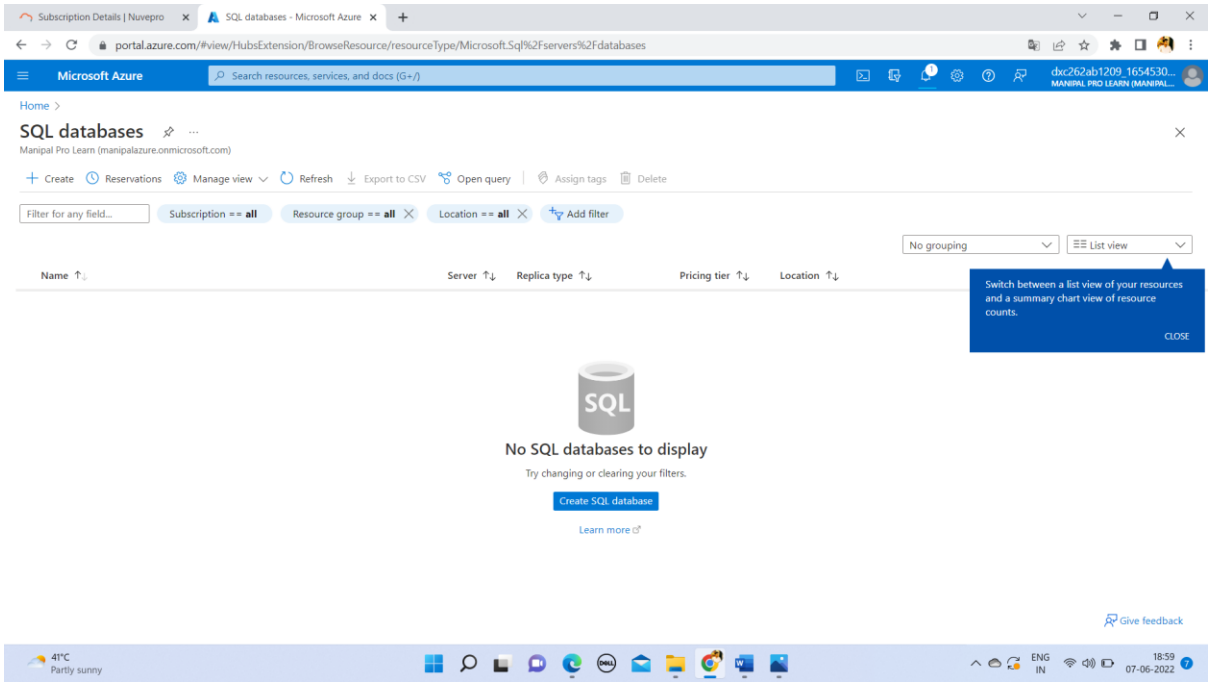
Now click on save bottom.



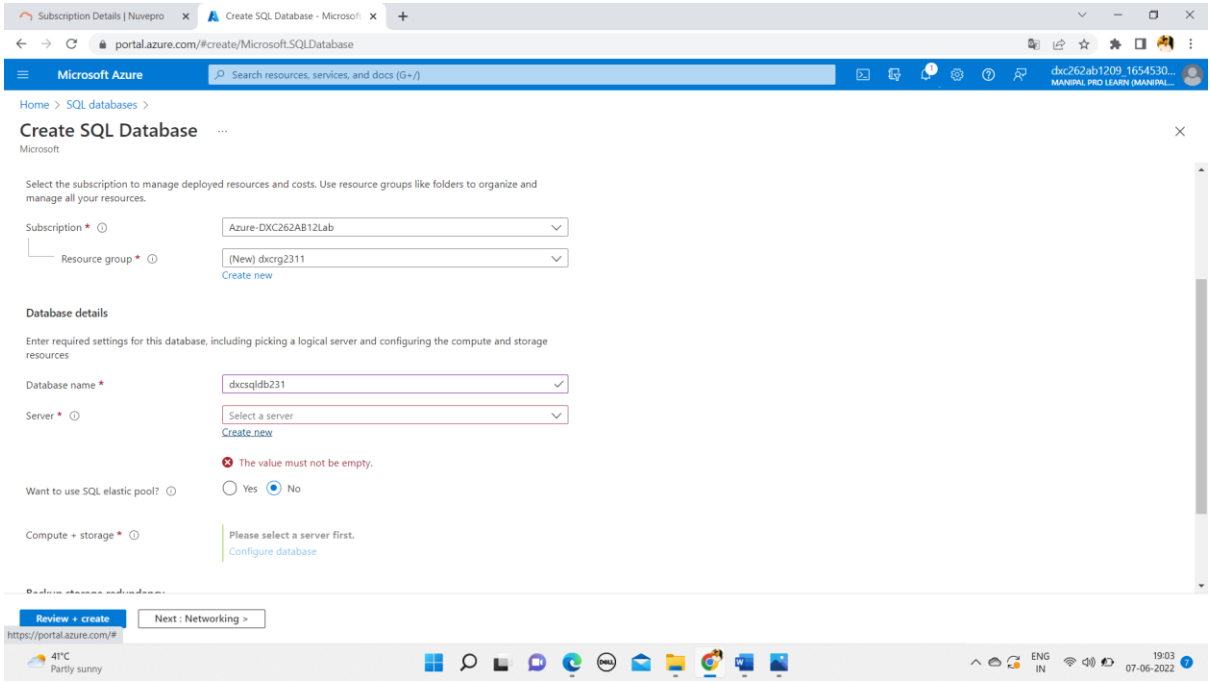
Click on **New Item** again, and create and save another document with a unique id, and any other properties and values you want. Your documents can have any structure, because Azure Cosmos DB doesn't impose any schema on your data.

QUESTION 10. Explain with sceenshots & step how to create Azure SQL Db & also explain how to insert data into Azure SQL D?

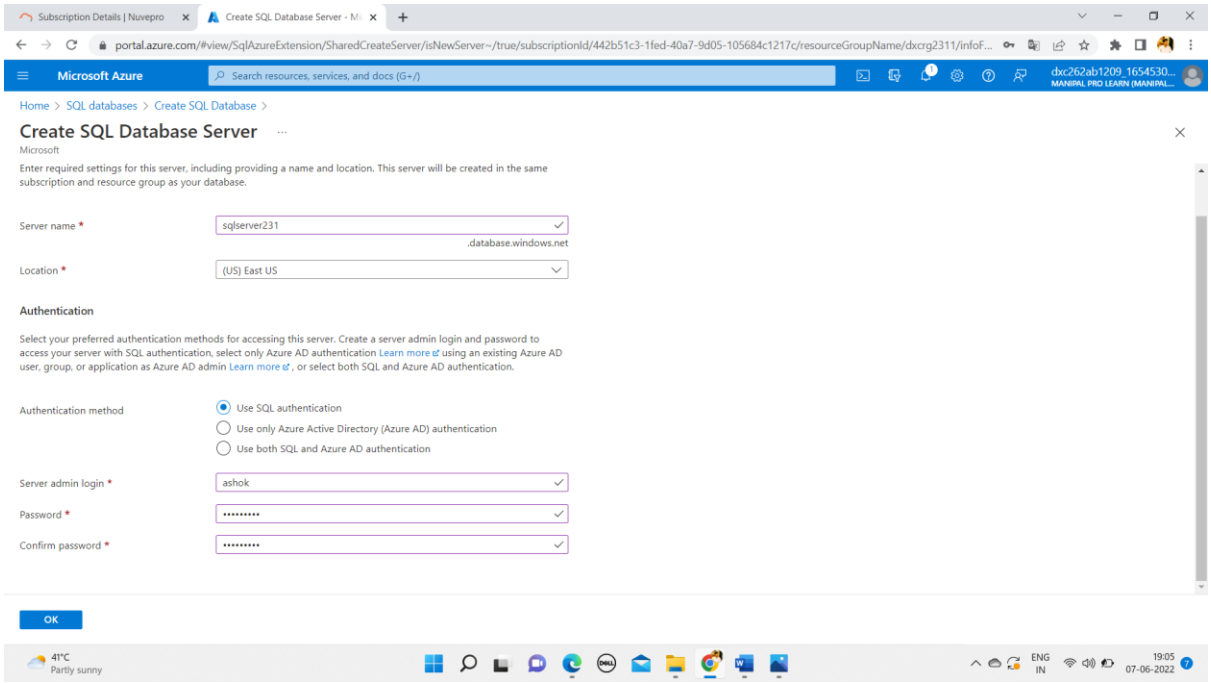
ANSWER: Go to https://portal.azure.com/#home and search **sql databases**



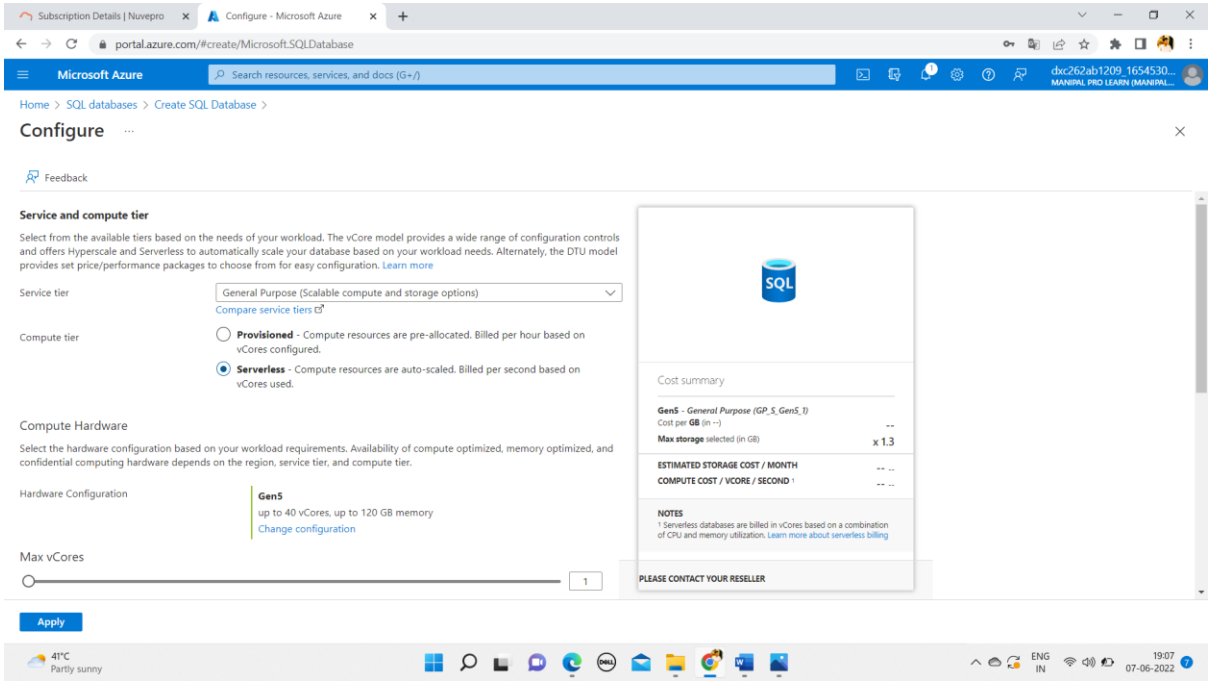
Click on create and give fill details like resource group and database name.

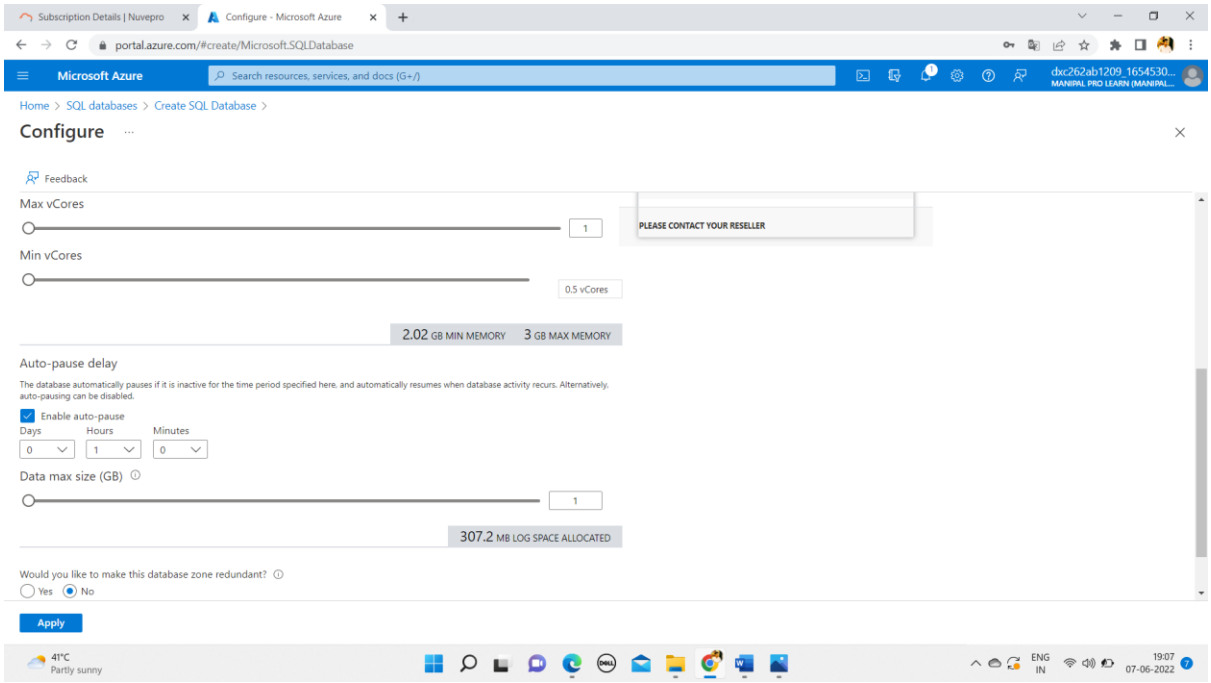


click on create new to create new server and fill details for server like server name and server admin details.

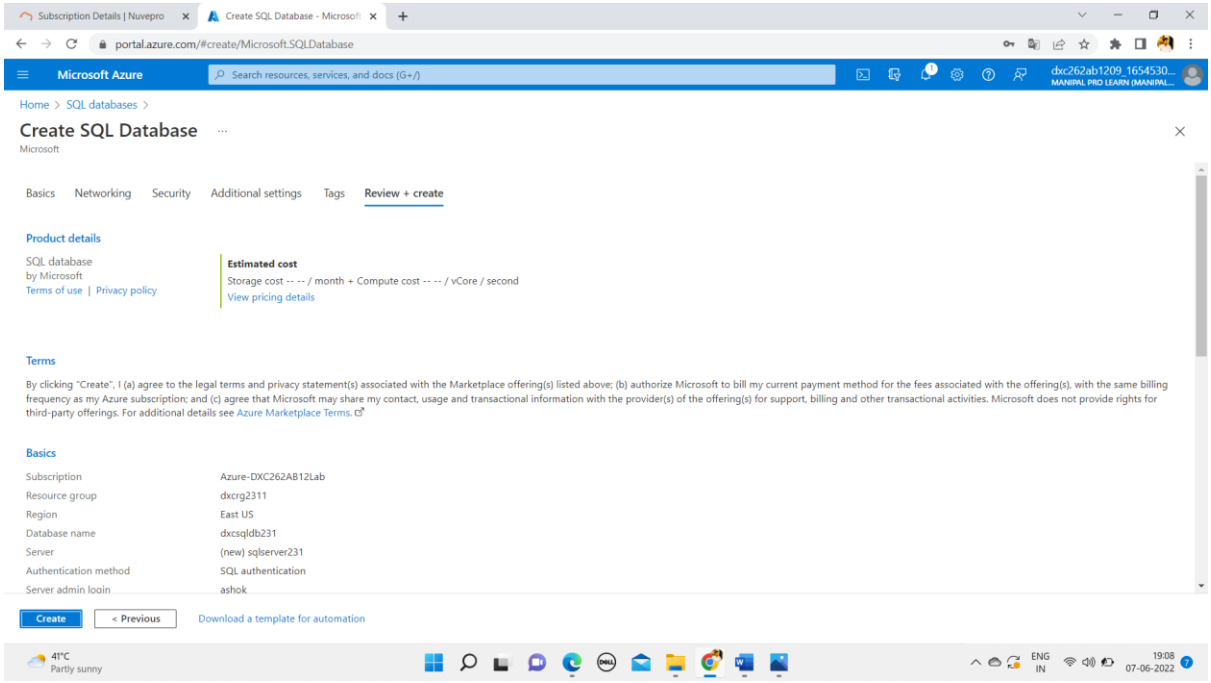


Now click on configure database and do the needful and click apply.

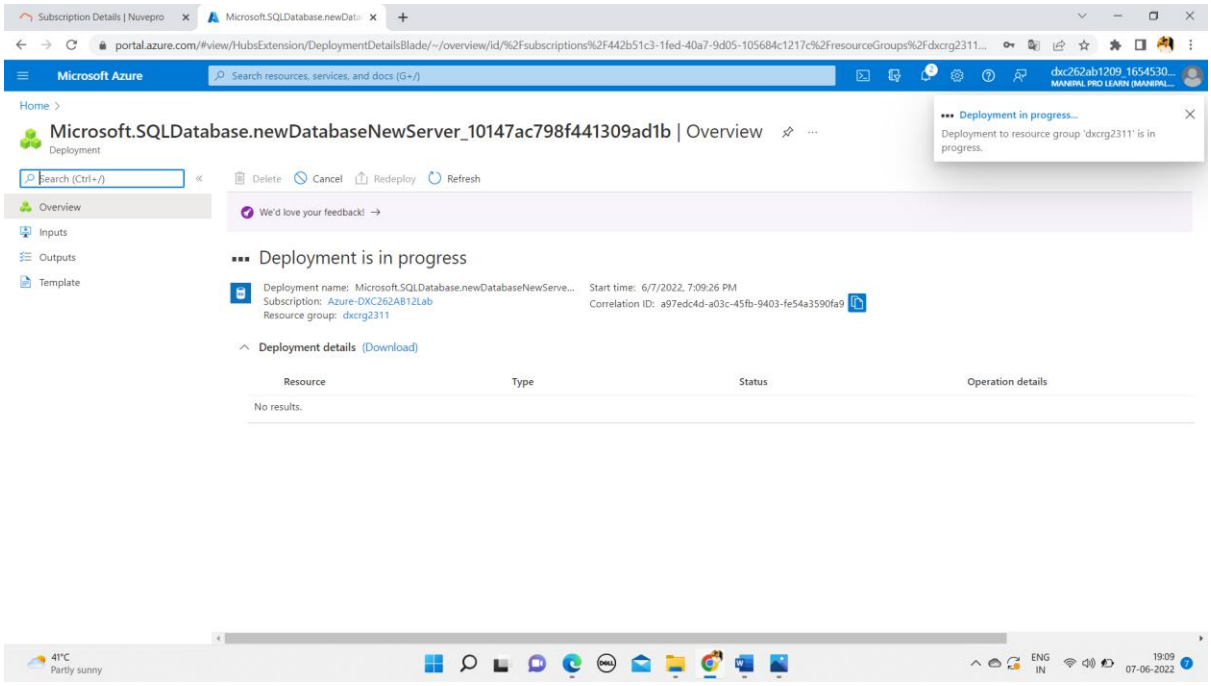


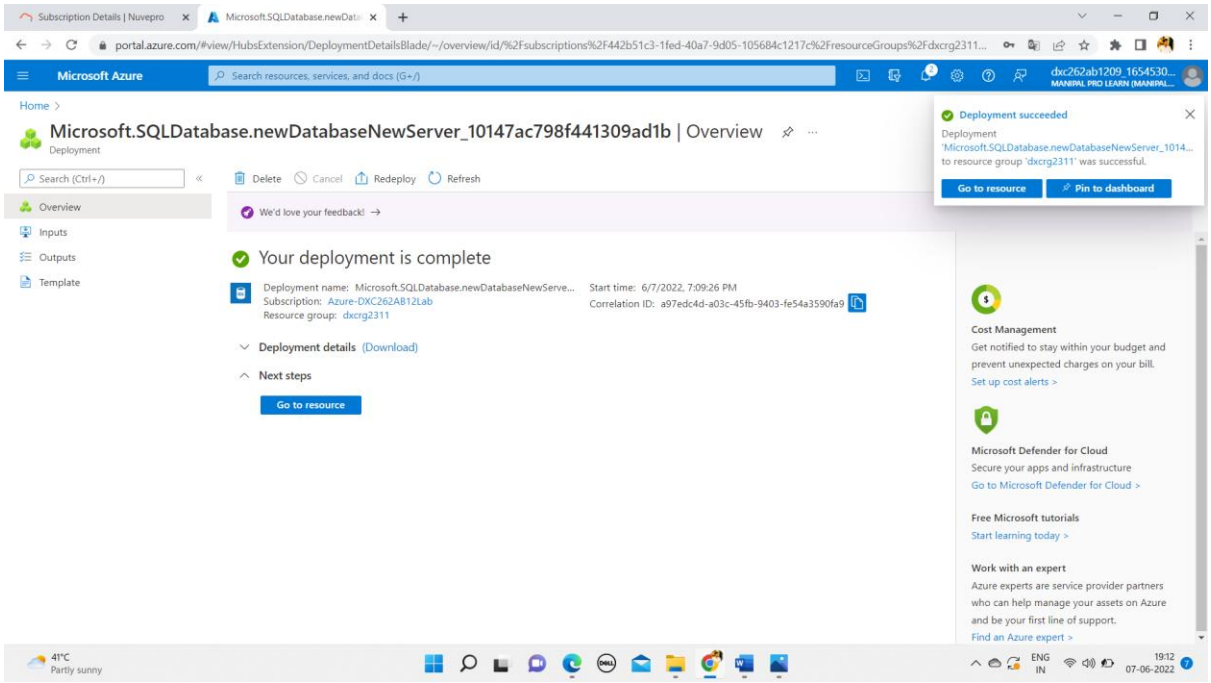


Now click on review+create.

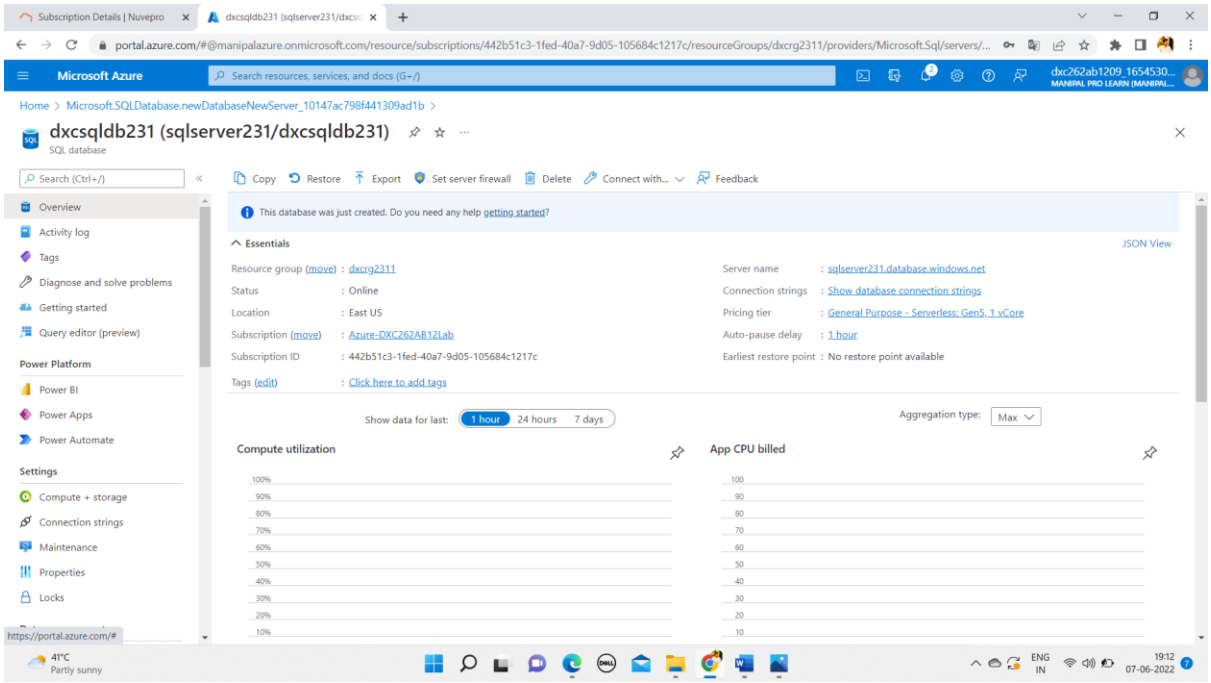


Click on create.

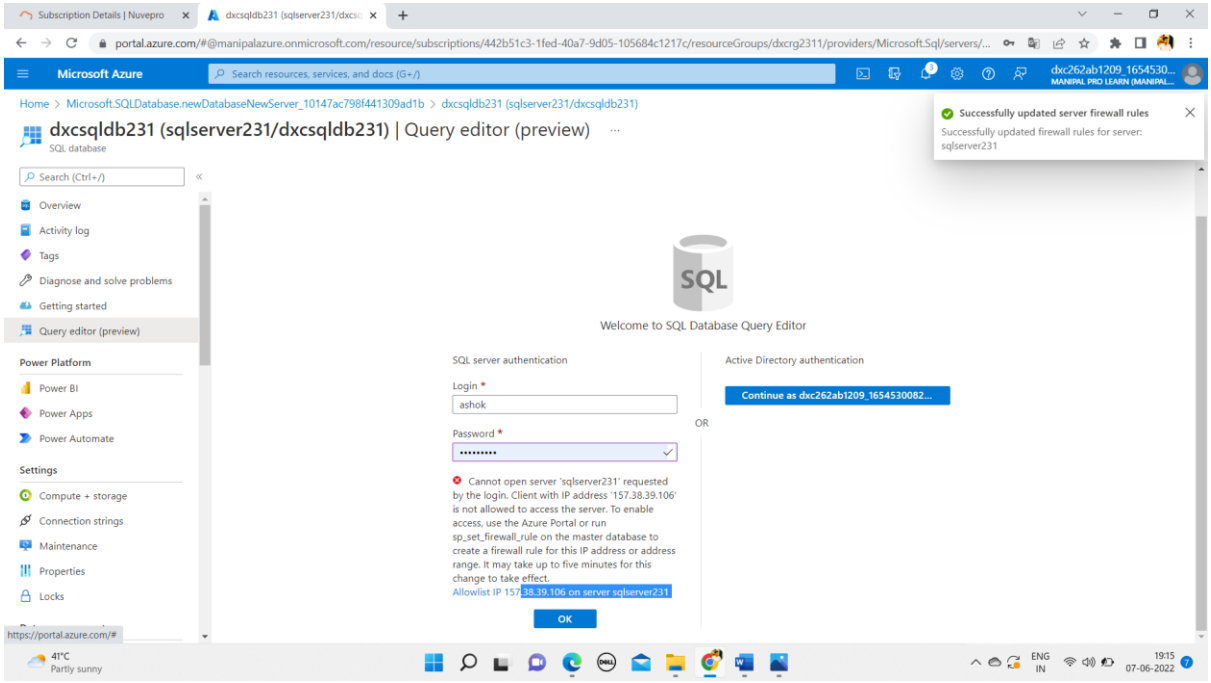




Click on go to resources



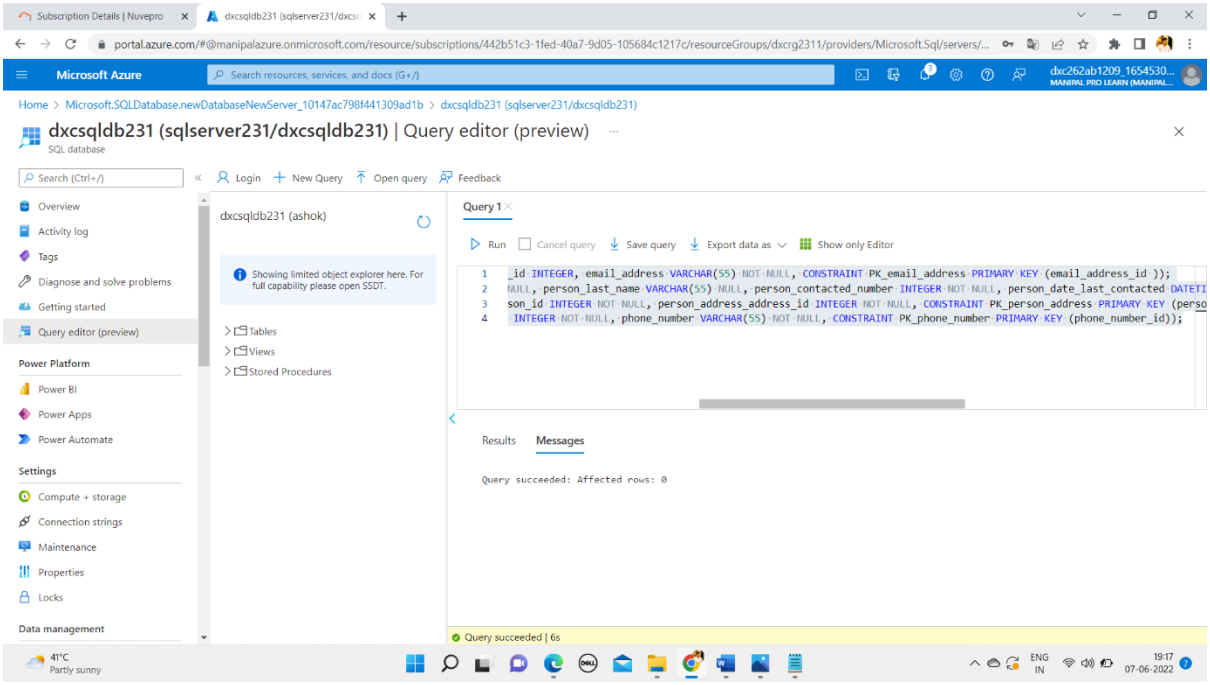
Click on query editor (preview) and fill login details of server admin and allow address to access the server.



Write and run these queries for create tables.

```
CREATE TABLE address( address_id INTEGER NOT NULL, address_building_number VARCHAR(55) NOT NULL, address_street VARCHAR(55) NOT NULL, address_locality VARCHAR(55), address_city VARCHAR(55) NOT NULL, address_zip_postal VARCHAR(55) NOT NULL, address_state_province_county VARCHAR(55) NOT NULL, address_country VARCHAR(55) NOT NULL , CONSTRAINT PK_Address PRIMARY KEY (address_id) );
```

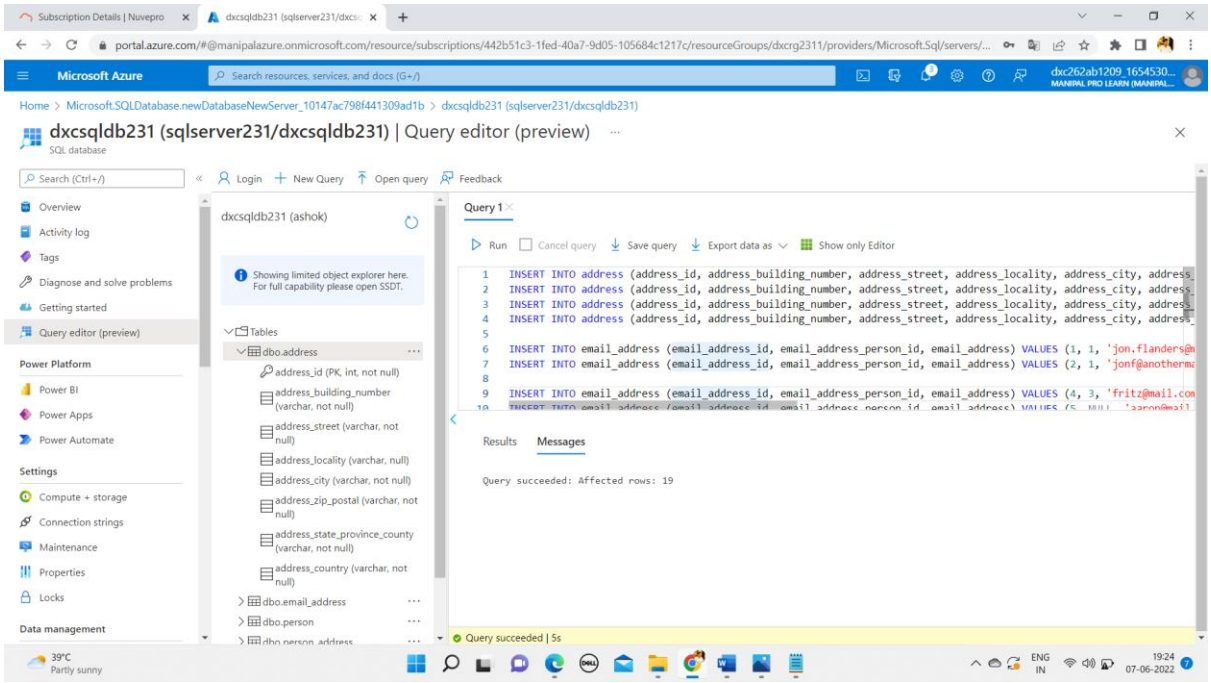
```
CREATE TABLE email_address( email_address_id INTEGER NOT NULL, email_address_p
erson_id INTEGER, email_address VARCHAR(55) NOT NULL, CONSTRAINT PK_email_addr
ess PRIMARY KEY (email_address_id ));
CREATE TABLE person( person_id INTEGER NOT NULL, person_first_name VARCHAR(55)
 NOT NULL, person_last_name VARCHAR(55) NULL, person_contacted_number INTEGER
NOT NULL, person_date_last_contacted DATETIME NOT NULL,person_date_added DATET
IME NOT NULL, CONSTRAINT PK_person PRIMARY KEY (person_id));
CREATE TABLE person_address( person_address_id INTEGER NOT NULL, person_addres
s_person_id INTEGER NOT NULL, person_address_address_id INTEGER NOT NULL, CONS
TRAINT PK_person_address PRIMARY KEY (person_address_id));
CREATE TABLE phone_number( phone_number_id INTEGER NOT NULL, phone_number_pers
on_id INTEGER NOT NULL, phone_number VARCHAR(55) NOT NULL, CONSTRAINT PK_phone
_number PRIMARY KEY (phone_number_id));
```



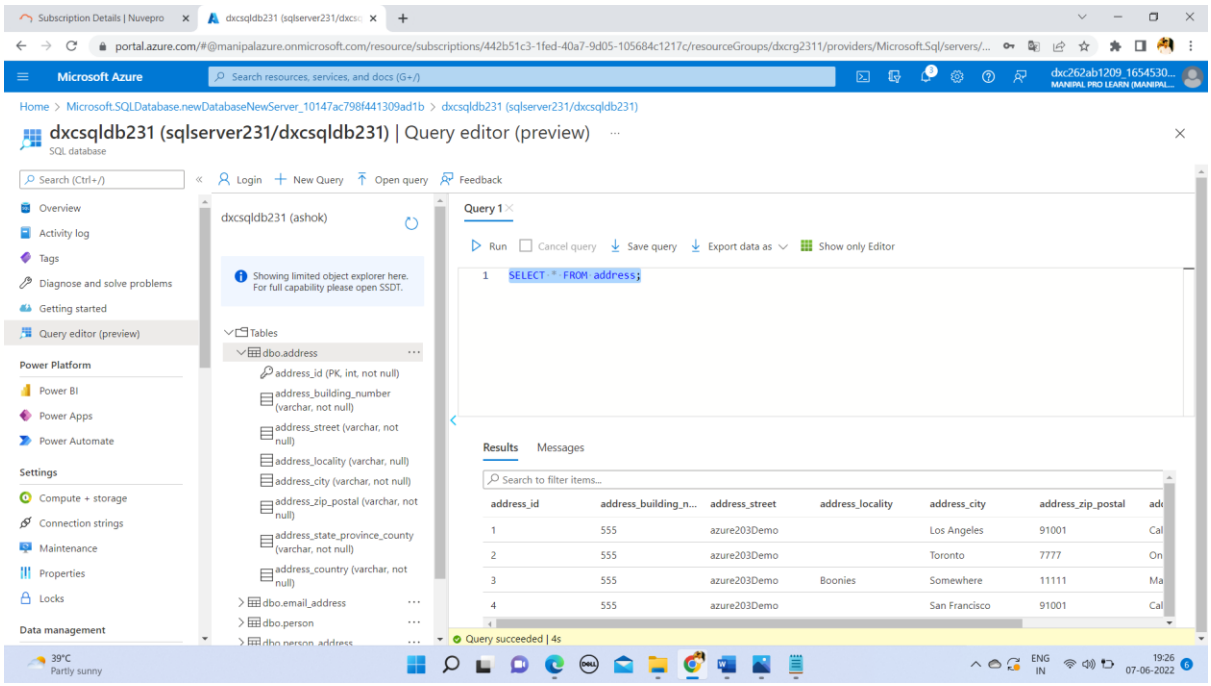
Queries for inserting data into tables

```
INSERT INTO address (address_id, address_building_number, address_street, addr
ess_locality, address_city, address_zip_postal, address_state_province_county,
 address_country) VALUES (4, '555', 'azure203Demo', NULL, 'San Francisco', '91
001', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, addr
ess_locality, address_city, address_zip_postal, address_state_province_county,
 address_country) VALUES (1, '555', 'azure203Demo', NULL, 'Los Angeles', '9100
1', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, addr
ess_locality, address_city, address_zip_postal, address_state_province_county,
 address_country) VALUES (2, '555', 'azure203Demo', NULL, 'Toronto', '7777', '
Ontario', 'Canada');
INSERT INTO address (address_id, address_building_number, address_street, addr
ess_locality, address_city, address_zip_postal, address_state_province_county,
```

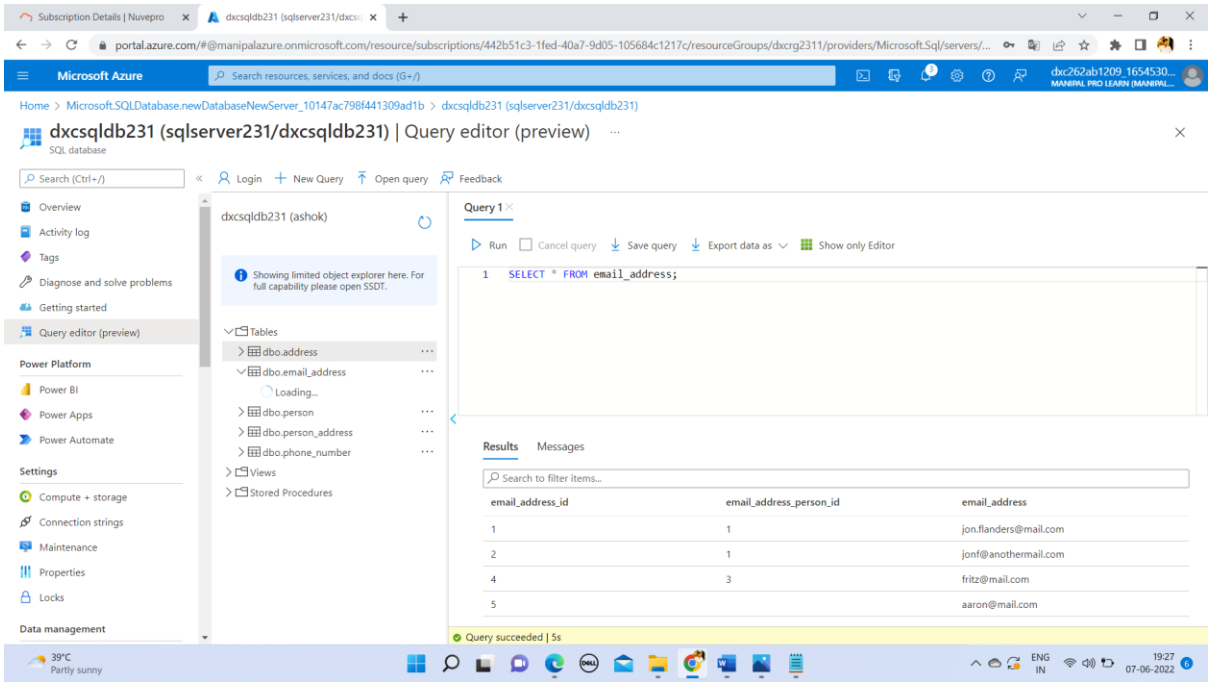
```
 address_country) VALUES (3, '555', 'azure203Demo', 'Boonies', 'Somewhere', '1
1111', 'Maine', 'US');

INSERT INTO email_address (email_address_id, email_address_person_id, email_ad
dress) VALUES (1, 1, 'jon.flanders@mail.com');
INSERT INTO email_address (email_address_id, email_address_person_id, email_ad
dress) VALUES (2, 1, 'jonf@anothermail.com');

INSERT INTO email_address (email_address_id, email_address_person_id, email_ad
dress) VALUES (4, 3, 'fritz@mail.com');
INSERT INTO email_address (email_address_id, email_address_person_id, email_ad
dress) VALUES (5, NULL, 'aaron@mail.com');

INSERT INTO person (person_id, person_first_name, person_last_name, person_con
tacted_number,person_date_last_contacted,person_date_added ) VALUES (1, 'Jon',
 'Flanders', 5,'2013-09-14 11:43:31','2013-01-14 11:43:31');
INSERT INTO person (person_id, person_first_name, person_last_name, person_con
tacted_number,person_date_last_contacted,person_date_added) VALUES (2, 'Shanno
n', 'Ahern', 0,'2013-08-14 11:43:31','2013-02-14 11:43:31');
INSERT INTO person (person_id, person_first_name, person_last_name, person_con
tacted_number,person_date_last_contacted,person_date_added) VALUES (3, 'Fritz'
, 'Onion', 1,'2013-07-14 11:43:31','2013-03-14 11:43:31');

INSERT INTO person_address (person_address_id, person_address_person_id, perso
n_address_address_id) VALUES (1, 1, 1);
INSERT INTO person_address (person_address_id, person_address_person_id, perso
n_address_address_id) VALUES (3, 2, 1);
INSERT INTO person_address (person_address_id, person_address_person_id, perso
n_address_address_id) VALUES (4, 2, 2);
INSERT INTO person_address (person_address_id, person_address_person_id, perso
n_address_address_id) VALUES (5, 3, 3);

INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_numbe
r) VALUES (1, 1, '555-1212');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_numbe
r) VALUES (2, 2, '555-1213');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_numbe
r) VALUES (3, 3, '555-1214');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_numbe
r) VALUES (4, 3, '555-1215');
```



Run `SELECT * FROM address;` to see **address** table.



Run `SELECT * FROM email_address;` to see **email_address** table.


Subscription Details | Nuvepro
dxcsqldb231 (sqlserver231/dxcs
portal.azure.com/#@manipalazure.onmicrosoft.com/resource/subscriptions/442b51c3-1fed-40a7-9d05-105684c1217c/resourceGroups/dxcrg2311/providers/Microsoft.Sql/servers/...
Microsoft Azure
Search resources, services, and docs (G+/)
dxc262ab1209_1654530...
MANIPAL PRO LEARN (MANIPAL...
Home > Microsoft.SQLDatabase.newDatabaseNewServer_10147ac798f441309ad1b > dxcsqldb231 (sqlserver231/dxcsqldb231)
dxcsqldb231 (sqlserver231/dxcsqldb231) | Query editor (preview)
SQL database
Search (Ctrl+/)
Login
New Query
Open query
Feedback
Overview
Activity log
Tags
Diagnose and solve problems
Getting started
Query editor (preview)
Power Platform
Power BI
Power Apps
Power Automate
Settings
Compute + storage
Connection strings
Maintenance
Properties
Locks
Data management
dxcsqldb231 (ashok)
Showing limited object explorer here. For full capability please open SSDT.
Tables
dbo.address
dbo.email_address
Loading...
dbo.person
dbo.person_address
dbo.phone_number
Views
Stored Procedures
Query 1
Run
Cancel query
Save query
Export data as
Show only Editor
1 SELECT * FROM email_address;
Results
Messages
Search to filter items...
email_address_id
email_address_person_id
email_address
1
1
jon.flanders@mail.com
2
1
jonf@anothermail.com
4
3
fritz@mail.com
5
aaron@mail.com
Query succeeded | 5s
39°C
Partly sunny
ENG
IN
19:27
07-06-2022

Thank you!

ASHOK KUMAR