

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:

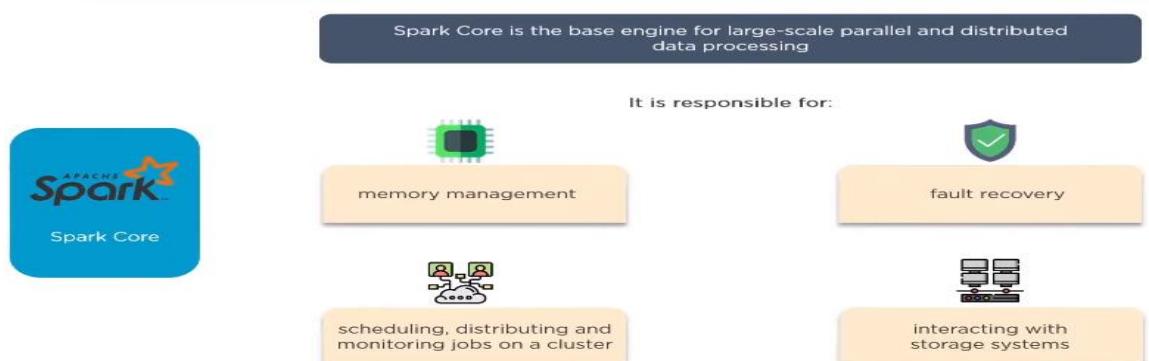


Apache Spark Core

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:

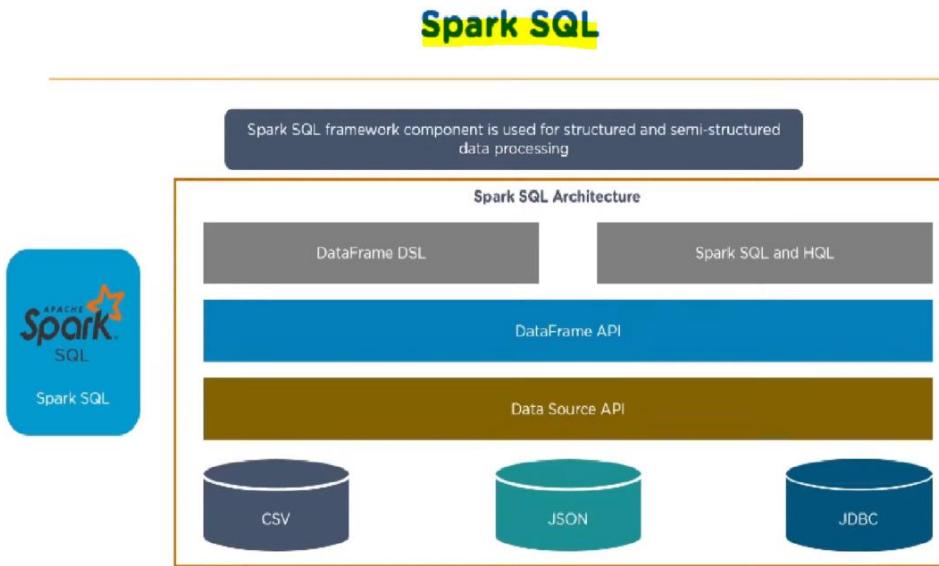
- It contains the basic functionality of spark. (Task scheduling, memory management, fault recovery, interacting with storage systems).
- Home to API that defines RDDs.

Spark Core



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.

Spark Streaming

Spark Streaming is a lightweight API that allows developers to perform batch processing and real-time streaming of data with ease

Provides secure, reliable, and fast processing of live data streams

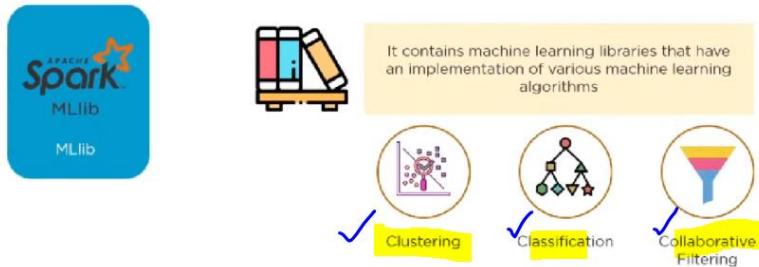


4. Mlib Machine Learning: MLlib in spark is a scalable Machine learning library that contains various machine learning algorithms. The motive behind MLlib creation is to make the implementation of machine learning simple. It contains machine learning libraries and the implementation of various algorithms.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

Spark MLlib

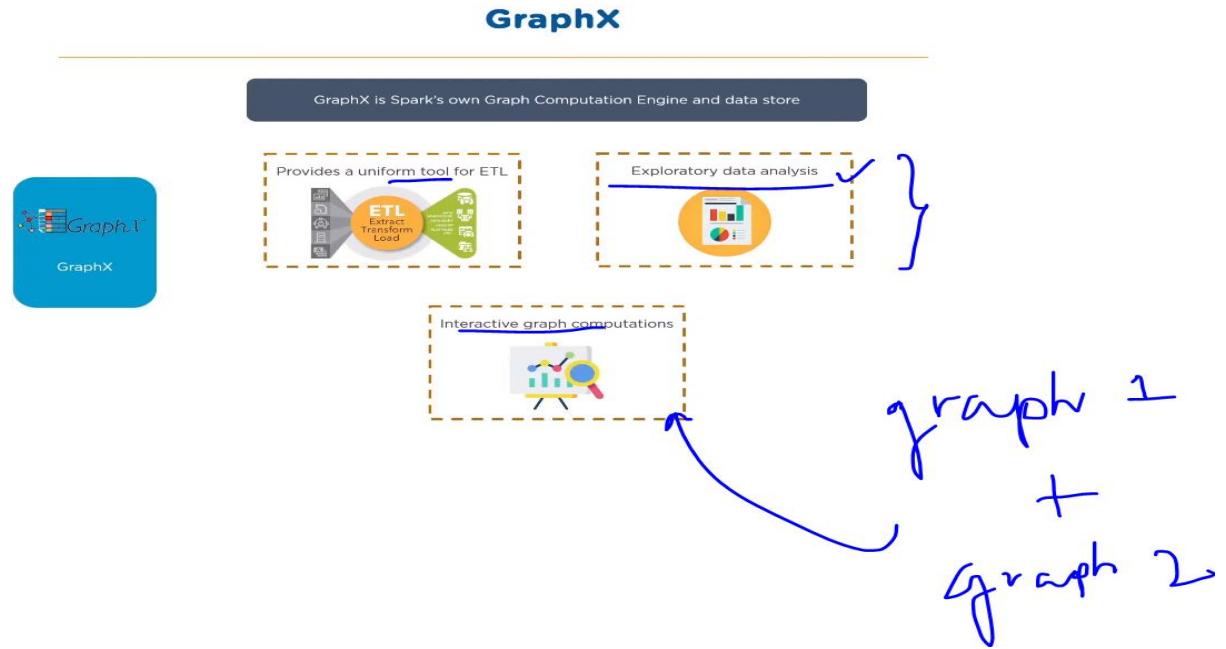
MLlib is a low-level machine learning library that is simple to use, is scalable, and compatible with various programming languages

MLlib eases the deployment and development of scalable machine learning algorithms



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.

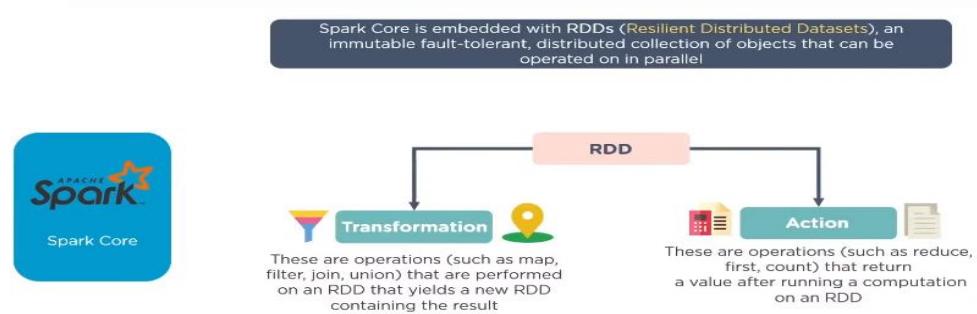
- Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects,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

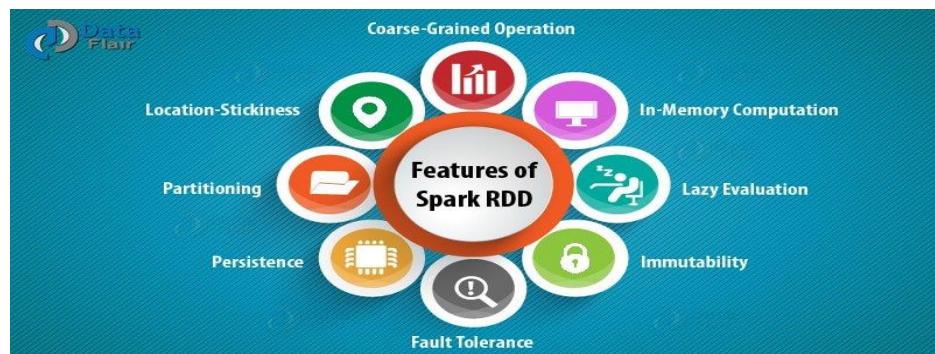
There are two ways to create RDDs – parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

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

Resilient Distributed Dataset



features of Apache Spark RDD are:

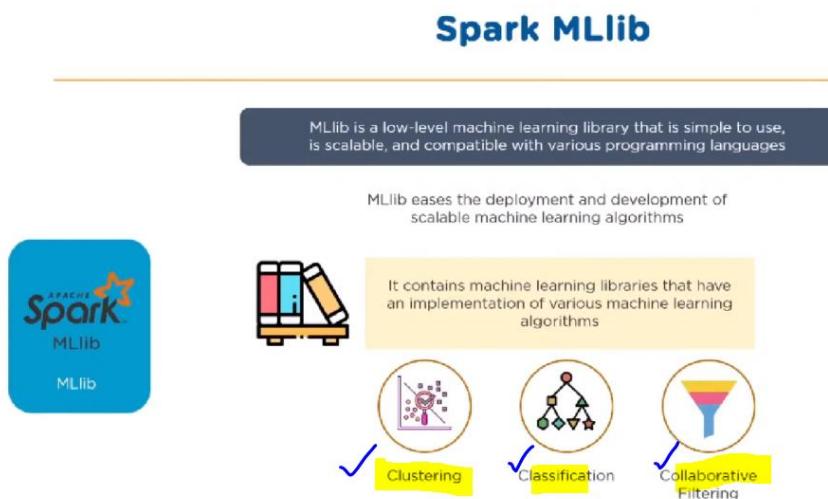


QUESTION 3. Explain various Mlib algorithms Spark is supporting ?

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:

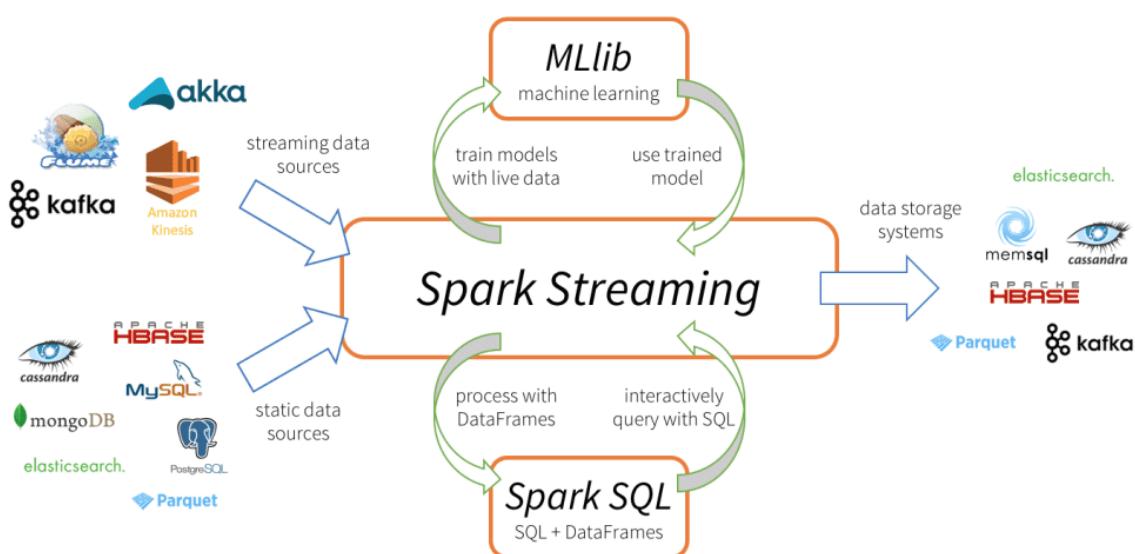
1. **Integrated:** Apache Spark SQL mixes SQL queries with Spark programs. With the help of Spark SQL, we can query structured data as a distributed dataset (RDD). We can run SQL queries alongside complex analytic algorithms using tight integration property of Spark SQL.
2. **Unified Data Access:** Using 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.

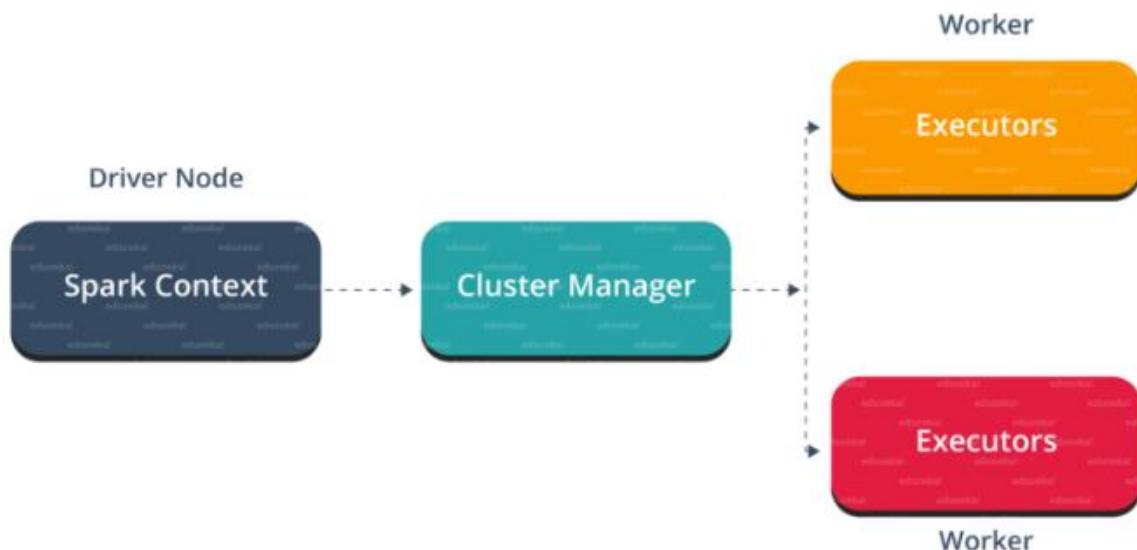
QUESTION 6. Explain SPARK architecture? what is Master - Slave architecture ?

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:

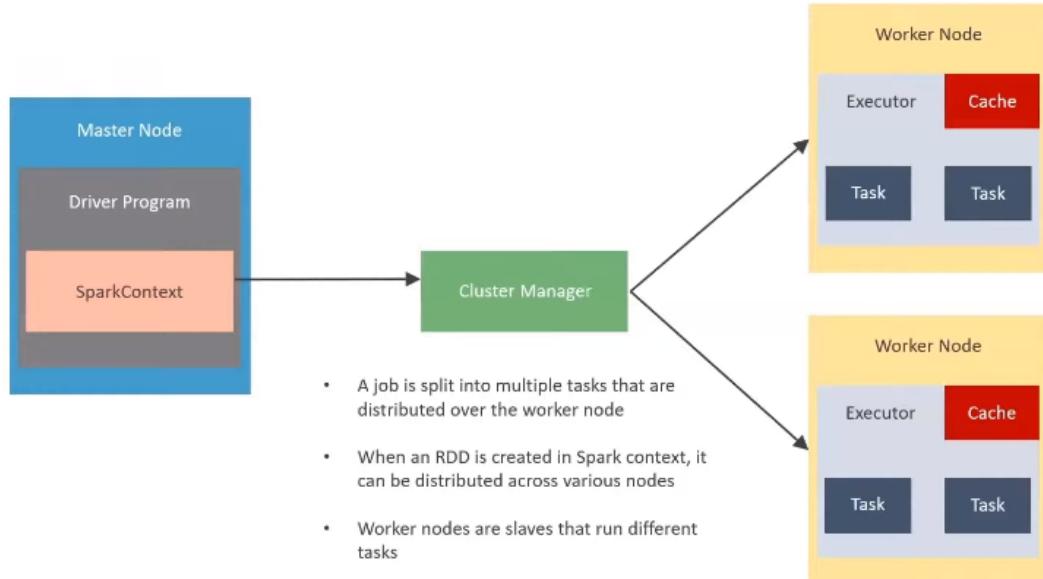
- Resilient Distributed Dataset (RDD)
- Directed Acyclic Graph (DAG)

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:

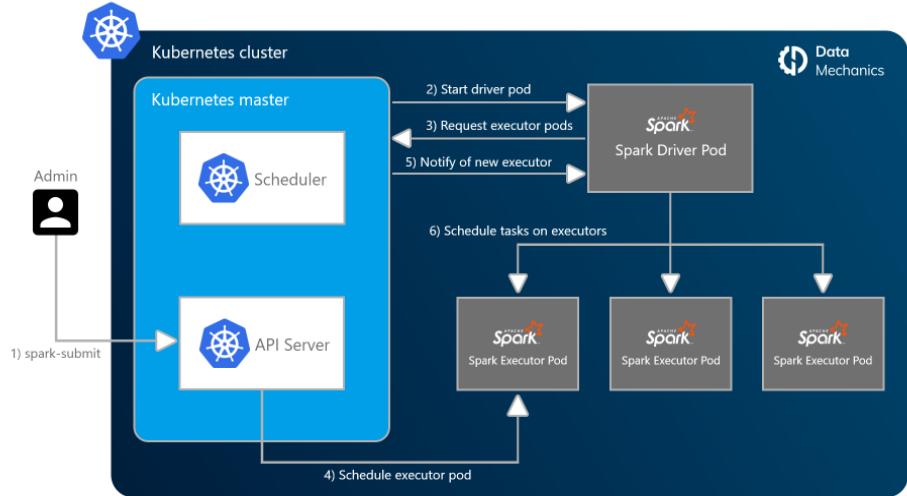
1. Standalone cluster manager
2. Mesos
3. Hadoop Yarn
4. Kubernetes



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.

Apache Spark on Kubernetes Architecture



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.

The screenshot shows the Azure portal interface for managing Azure Cosmos DB accounts. The top navigation bar includes 'Subscription Details | Nuvepro', 'Azure Cosmos DB - Microsoft Az...', and a search bar. Below the header, there's a 'Microsoft Azure' logo and a search bar with placeholder text 'Search resources, services, and docs (G+ /)'. A user profile icon is visible in the top right corner. The main content area is titled 'Azure Cosmos DB' and shows a message: 'No Azure Cosmos DB accounts to display'. It provides a brief description: 'Create a globally distributed, multi-model, fully managed database using API of your choice. Or try it for free, up to 20K RU/s, for 30 days with unlimited renewal.' Below this is a large blue 'Create Azure Cosmos DB account' button. At the bottom of the page, there's a status bar with weather information (41°C, Partly sunny), system icons, and a timestamp (07-06-2022 18:23).

Click on create bottom.

The screenshot shows the 'Select API option' blade for creating a new Azure Cosmos DB account. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Select API option' and asks 'Which API best suits your workload?'. It states that Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. Below this, a note says 'To start, select the API to create a new account. The API selection cannot be changed after account creation.' There are five API options listed in cards: 1. **Core (SQL) - Recommended**: Describes it as the core or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java. Includes 'Create' and 'Learn more' buttons. 2. **Azure Cosmos DB API for MongoDB**: Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB. Includes 'Create' and 'Learn more' buttons. 3. **Cassandra**: Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB. Includes 'Create' and 'Learn more' buttons. 4. **Azure Table**: Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB, but do not want to re-write your application to use the SQL API. Includes 'Create' and 'Learn more' buttons. 5. **Gremlin (Graph)**: Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data. Includes 'Create' and 'Learn more' buttons. At the bottom of the page, there's a status bar with weather information (41°C, Partly sunny), system icons, and a timestamp (07-06-2022 18:25).

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.

Subscription Details | Nuvepro > Create Azure Cosmos DB Account - Core (SQL)

Microsoft Azure > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Core (SQL) ...

Basics Global Distribution Networking Backup Policy Encryption Tags Review + create

Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. Try it for free, for 30 days with unlimited renewals. Go to production starting at \$24/month per database, multiple containers included. Learn more

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure-DXC262AB12Lab

Resource Group * (New) dxcrg231

Create new

Instance Details

Account Name * dxccosmosdb23177

Location * (US) East US

Capacity mode Provisioned throughput Serverless

Learn more about capacity mode

Review + create Previous Next: Global Distribution

Subscription Details | Nuvepro > Create Azure Cosmos DB Account - Core (SQL)

Microsoft Azure > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Core (SQL) ...

Validation Success

Basics Global Distribution Networking Backup Policy Encryption Tags Review + create

Creation Time

Estimated Account Creation Time (in minutes) 2

The estimated creation time is calculated based on the location you have selected

Basics

Subscription Azure-DXC262AB12Lab
Resource Group (new) dxcrg231
Location East US
Account Name (new) dxccosmosdb23177
API Core (SQL)
Capacity mode Serverless
Availability Zones Disable

Backup Policy

Backup policy Periodic

Create Previous Next Download a template for automation

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**.

Subscription Details | Nuvepro > Microsoft.Azure.CosmosDB-20220607182946 | Overview

Microsoft Azure > Search resources, services, and docs (G+)

Microsoft.Azure.CosmosDB-20220607182946 | Overview

Deployment

Search (Ctrl+ /) Delete Cancel Redeploy Refresh

We'd love your feedback! →

... Deployment is in progress

Deployment name: Microsoft.Azure.CosmosDB-20220607182946
Subscription: Azure-DXC262AB12Lab
Resource group: dxcrg231

Start time: 6/7/2022, 6:29:58 PM
Correlation ID: 431c4a03-2554-4bd2-a7e6-dbc4ef6cb841

Deployment details (Download)

Resource	Type	Status	Operation details
No results.			

41°C Partly sunny ENG IN 07-06-2022 18:30

The screenshot shows the Microsoft Azure Deployment blade for 'Microsoft.Azure.CosmosDB-20220607182946'. The main message is 'Your deployment is complete'. Deployment details include name: Microsoft.Azure.CosmosDB-20220607182946, subscription: Azure-DXC262AB12Lab, resource group: dxcrg231. A 'Go to resource' button is present. The blade includes links for Cost Management, Microsoft Defender for Cloud, and Free Microsoft tutorials. The system tray at the bottom shows the date as 07-06-2022 and time as 18:33.

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

The screenshot shows the Azure Cosmos DB account page for 'dxccosmosdb23177'. The left sidebar lists options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Notifications, Data Explorer, Features, Default consistency, Backup & Restore, Firewall and virtual networks, Private Endpoint Connections, CORS, Dedicated Gateway, Keys, and Advisor Recommendations. The main area displays 'Essentials' information: Status (Online), Resource group (dxcrg231), Subscription (Azure-DXC262AB12Lab), Subscription ID (442b51c3-1fed-40a7-9d05-105684c1217c), Backup policy (Periodic), Read Locations (East US), Write Locations (East US), URI (https://dxccosmosdb23177.documents.azure.com:443), and Capacity mode (Serverless). It also shows 'Containers' (empty), 'Monitoring' (with a 24 hours filter selected), and 'Requests' (empty). The system tray at the bottom shows the date as 07-06-2022 and time as 18:36.

QUESTION 9. Explain with screenshots & 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.

The screenshot shows the Microsoft Azure portal with the URL <https://portal.azure.com/#/resource/subscriptions/442b51c3-1fed-40a7-9d05-105684c1217c/resourcegroups/dxcrg231/providers/Microsoft.DocumentDB/data...>. The browser tab is titled "dxccosmosdb23177 - Microsoft". The main content area is titled "Welcome to C" and "Globally distributed, multi-model data". On the left, the "Data Explorer" section is selected, showing options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Notifications, and Data Explorer. The "Settings" section includes Default consistency, Backup & Restore, Firewall and virtual networks, Private Endpoint Connections, CORS, Dedicated Gateway, Keys, and Advisor Recommendations. The weather icon shows 41°C Partly sunny. In the center, there are two cards: "Launch quick start" and "New Container". The "New Container" card has a form with fields: "Database id" (radio buttons for "Create new" (selected) and "Use existing" with input "sports"), "Container id" (input "playername"), "Partition key" (input "/players"), and "Unique keys" (checkbox). Below the form, there's an "Analytical store" section with "On" (radio button selected) and "Off" (radio button), and a note about Azure Synapse Link. A "Enable" button is at the bottom right of the form. The status bar at the bottom right shows ENG IN, 18:40, and 07-06-2022.

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"  
}
```

The screenshot shows the Microsoft Azure Data Explorer interface for a Cosmos DB account named 'dxcosmosdb23177'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Notifications, and Data Explorer. Under Data Explorer, there are sections for Settings, Features, Default consistency, Backup & Restore, Firewall and virtual networks, Private Endpoint Connections, CORS, Dedicated Gateway, Keys, and Advisor Recommendations. The main area displays the SQL API interface for the 'playername' container. A search bar at the top right contains the query 'SELECT * FROM c'. The results pane shows a list of items with IDs ranging from 01 to 08, each corresponding to a player name: KL Rahul, MS Dhoni, Hardik Pandya, Rishabh Pant, Virat Kohli, Rohit Sharma, Ravindra Jadeja, and Dinesh Karthik. Below the results, a note states 'Notebooks is currently not available. We are working on it.'

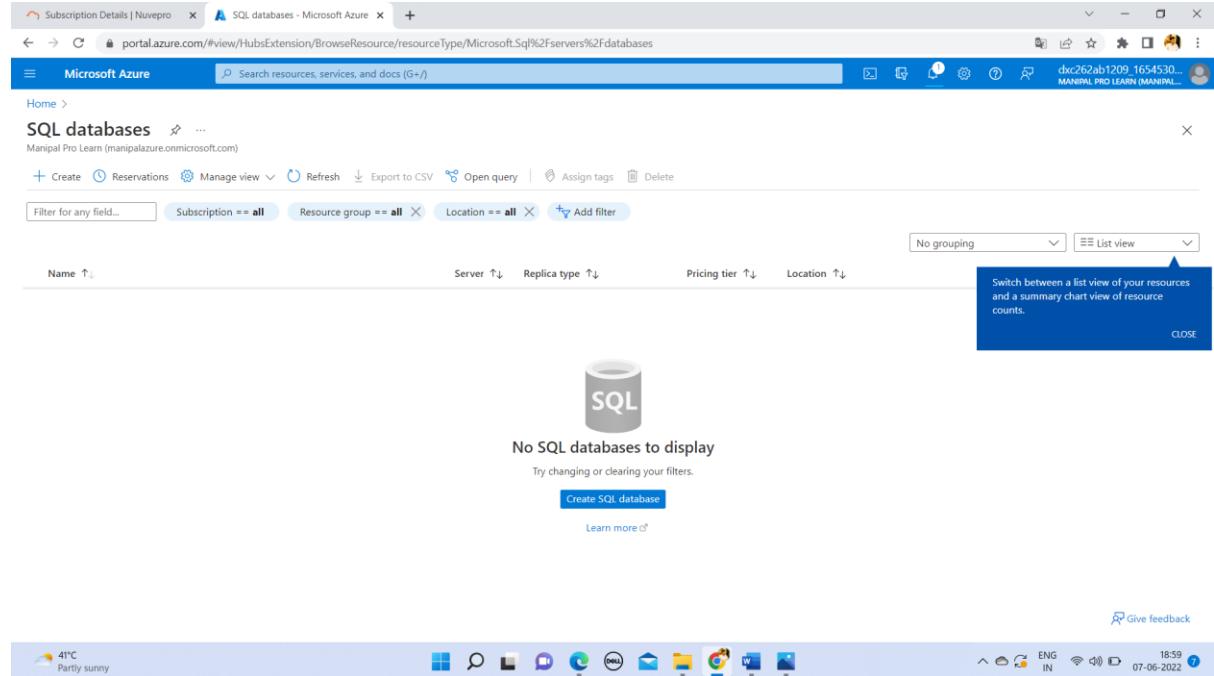
Now click on save bottom.

This screenshot is identical to the previous one, showing the Microsoft Azure Data Explorer interface for the 'dxcosmosdb23177' account. The main difference is the message 'Successfully created new item for container playername' displayed at the bottom of the results pane. The SQL query 'SELECT * FROM c' is still present in the search bar, and the list of player names remains the same.

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.

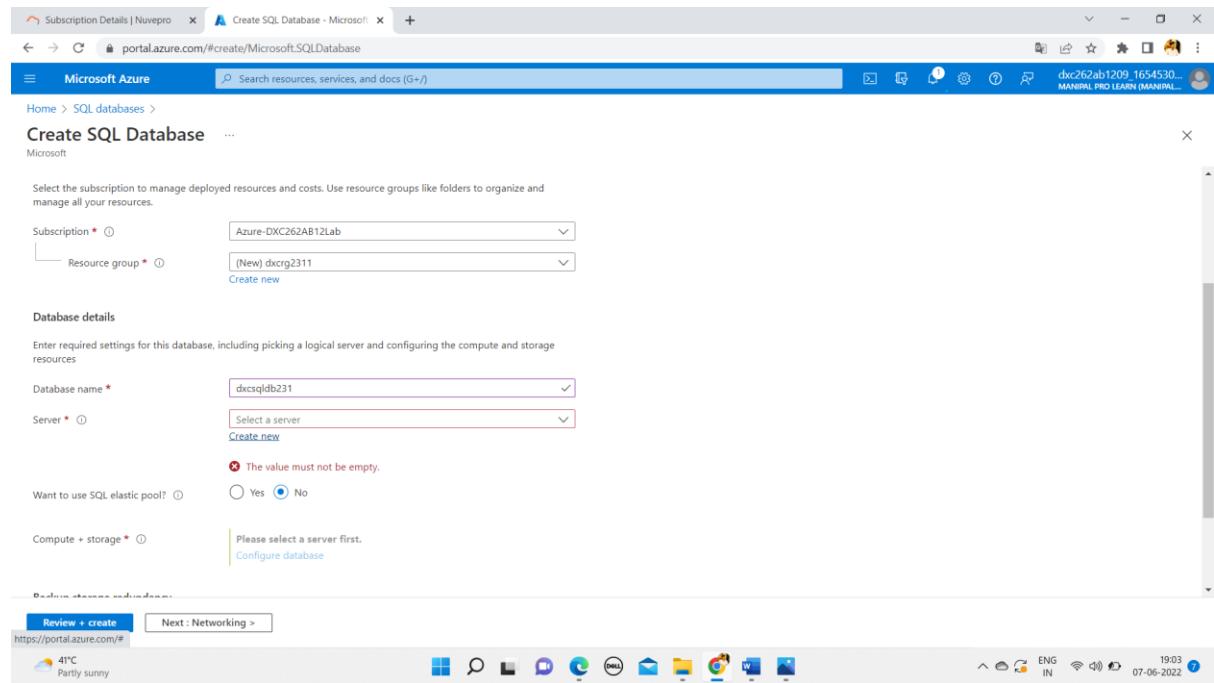
QUESTION 10. Explain with screenshots & step how to create Azure SQL Db & also explain how to insert data into Azure SQL D?

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



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Subscription Details | Nuvapro' and 'SQL databases - Microsoft Azure'. The main search bar says 'Search resources, services, and docs (G+)'. Below the search bar, there's a 'Microsoft Azure' logo and a user profile. The main content area is titled 'SQL databases' and shows a large 'SQL' icon. A message 'No SQL databases to display' is centered, with a note 'Try changing or clearing your filters.' and a blue 'Create SQL database' button. To the right, a tooltip says 'Switch between a list view of your resources and a summary chart view of resource counts.' The bottom of the screen shows a Windows taskbar with various pinned icons and the system tray indicating the date as 07-06-2022 and time as 18:59.

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



The screenshot shows the 'Create SQL Database' wizard in the Microsoft Azure portal. Step 1: 'Select subscription' shows 'Subscription' set to 'Azure-DXC262AB12Lab' and 'Resource group' set to '(New) dxcrg2311'. Step 2: 'Database details' shows 'Database name' set to 'dxcsql2b21', 'Server' dropdown set to 'Select a server', and 'Compute + storage' dropdown set to 'Please select a server first.'. Step 3: 'Review + create' shows 'Review + create' and 'Next : Networking >' buttons. The bottom of the screen shows a Windows taskbar with various pinned icons and the system tray indicating the date as 07-06-2022 and time as 19:03.

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

The screenshot shows the 'Create SQL Database Server' configuration page in Microsoft Azure. The server name is set to 'sqlserver231' and the location is '(US) East US'. The authentication method is set to 'Use SQL authentication'. The server admin login is 'ashok' and the password is '*****'. The confirm password is also '*****'. At the bottom, there is an 'OK' button.

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

The screenshot shows the 'Configure - Microsoft Azure' configuration page. Under 'Service and compute tier', the 'General Purpose (Scalable compute and storage options)' service tier is selected. Under 'Compute tier', the 'Serverless' option is selected. Under 'Compute Hardware', 'Gen5' hardware configuration is chosen. The 'Max vCores' slider is set to 1. A cost summary table is shown on the right:

Gens - General Purpose (GP_S, GenS, I)	Cost per GB (in -)	x 1.3
Max storage selected (in GB)	---	---
ESTIMATED STORAGE COST / MONTH	---	---
COMPUTE COST / VCORE / SECOND	---	---

NOTES:
1. Serverless databases are billed in vCores based on a combination of CPU and memory utilization. Learn more about serverless billing.

At the bottom, there is an 'Apply' button.

Subscription Details | Nuvepro > Configure - Microsoft Azure > portal.azure.com/#create/Microsoft.SQLDatabase

Microsoft Azure Search resources, services, and docs (G+)

Home > SQL databases > Create SQL Database > Configure ...

Max vCores: 1
Min vCores: 0.5 vCores

Auto-pause delay: Disabled

Data max size (GB): 1

Log space allocated: 307.2 MB

Would you like to make this database zone redundant? No

Apply

Now click on review+create.

Subscription Details | Nuvepro > Create SQL Database - Microsoft Azure > portal.azure.com/#create/Microsoft.SQLDatabase

Microsoft Azure Search resources, services, and docs (G+)

Home > SQL databases > Create SQL Database > Review + create

Basics Networking Security Additional settings Tags Review + create

Product details

SQL database by Microsoft Terms of use | Privacy policy

Estimated cost
Storage cost -- / month + Compute cost -- / vCore / second
View pricing details

Terms
By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see [Azure Marketplace Terms](#).

Basics

Subscription	Azure-DXC262AB12Lab
Resource group	dxcrg2311
Region	East US
Database name	dxcsqldb231
Server	(new) sqlserver231
Authentication method	SQL authentication
Server admin login	ashok

Create < Previous Download a template for automation

Click on create.

The screenshot shows the Microsoft Azure portal interface. The main title bar reads "Subscription Details | Nuvepro" and "Microsoft.SQLDatabase.newData...". The address bar shows the URL "portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/-/overview/id%2fsubscriptions%2f442b51c3-1fed-40a7-9d05-105684c1217c%2fresourceGroups%2fdxcrg2311...". The top navigation bar includes icons for Home, Search, and various service links. A user profile icon in the top right corner shows "MANUAL PRO LEARN (MANUAL)" and a notification badge with the number "1". A modal window titled "... Deployment in progress..." is open, stating "Deployment to resource group 'dxcrg2311' is in progress." Below the modal, the main content area shows the "Overview" tab selected for a resource named "Microsoft.SQLDatabase.newDatabaseNewServer_10147ac798f441309ad1b". It displays deployment details: Deployment name: Microsoft.SQLDatabase.newDatabaseNewServer_10147ac798f441309ad1b, Start time: 6/7/2022, 7:09:26 PM, Subscription: Azure-DXC262AB12Lab, Correlation ID: a97edc4d-a03c-45fb-9403-fe54a3590fa5, Resource group: dxcrg2311. A "Deployment details" section is present, but it says "No results.".

This screenshot shows the Microsoft Azure portal after a deployment has completed. The interface is identical to the previous one, with the same title bar, address bar, and user profile. The modal window now displays a green checkmark icon and the message "Deployment succeeded". It also states "'Microsoft.SQLDatabase.newDatabaseNewServer_10147ac798f441309ad1b' to resource group 'dxcrg2311' was successful." Below the modal, the "Overview" tab for the same resource shows a green checkmark icon and the message "Your deployment is complete". It lists the deployment details again. A "Next steps" section is visible, containing a "Go to resource" button. To the right of the main content area, there are promotional cards for "Cost Management", "Microsoft Defender for Cloud", "Free Microsoft tutorials", and "Work with an expert". The system tray at the bottom shows the date and time as 07-06-2022 19:09.

Click on go to resources

The screenshot shows the Azure portal interface for managing databases. On the left, a sidebar lists various database management options like Overview, Activity log, Tags, and Query editor (preview). The main content area displays the properties of the 'dxcsqlldb231' database. Key details include:

- Resource group:** dxcrg2311
- Status:** Online
- Location:** East US
- Subscription:** Azure-DXC262AB12Lab
- Subscription ID:** 442b51c3-1fed-40a7-9d05-105684c1217c
- Tags:** Click here to add tags
- Compute utilization:** A chart showing CPU usage over the last 1 hour.

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

The screenshot shows the 'Query editor (preview)' page for the 'dxcsqlldb231' database. The top right corner shows a success message: "Successfully updated server firewall rules". The main area is titled "Welcome to SQL Database Query Editor" and contains a SQL authentication dialog box. The dialog box fields are:

- Login:** ashok
- Password:** [redacted]
- OK** button

Below the dialog, there is a note about opening the server from a specific IP address:

Cannot open server 'sqlserver231' requested by the login. Client with IP address '157.39.39.106' is not allowed to access the server. To enable access, use the Azure Portal or run sp_set_firewall_rule on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect.
Allowlist IP 157.39.39.106 on server sqlserver231

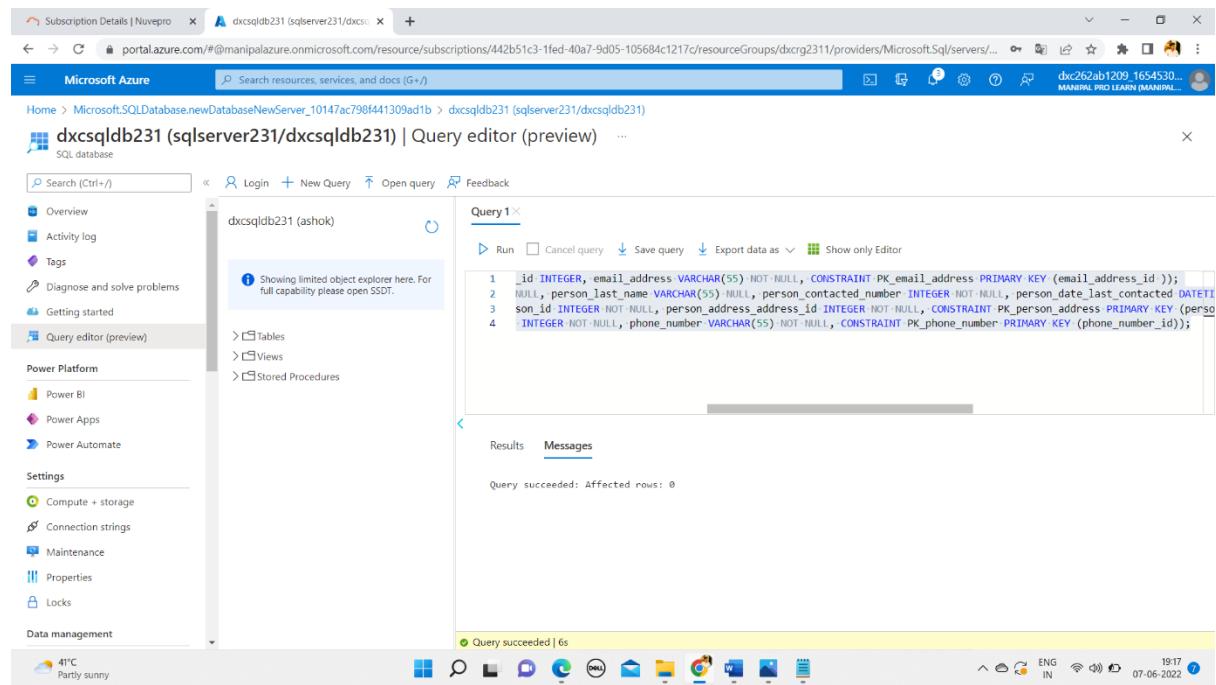
Write and run these queries for create tables.

```
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_person_id INTEGER, email_address VARCHAR(55) NOT NULL, CONSTRAINT PK_email_address 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 DATETIME NOT NULL, CONSTRAINT PK_person PRIMARY KEY (person_id));
CREATE TABLE person_address( person_address_id INTEGER NOT NULL, person_address_person_id INTEGER NOT NULL, person_address_address_id INTEGER NOT NULL, CONSTRAINT PK_person_address PRIMARY KEY (person_address_id));
CREATE TABLE phone_number( phone_number_id INTEGER NOT NULL, phone_number_person_id INTEGER NOT NULL, phone_number VARCHAR(55) NOT NULL, CONSTRAINT PK_phone_number PRIMARY KEY (phone_number_id));

```



The screenshot shows the Microsoft Azure portal interface for a SQL database named 'dxcsqlldb231'. The left sidebar contains navigation links for Overview, Activity log, Tags, Diagnose and solve problems, Getting started, and Query editor (preview). The main area displays a query editor titled 'Query 1' with the following T-SQL code:

```

1 _id INTEGER, email_address VARCHAR(55) NOT NULL, CONSTRAINT PK_email_address PRIMARY KEY (email_address_id));
2 NULL, person_last_name VARCHAR(55) NULL, person_contacted_number INTEGER NOT NULL, person_date_last_contacted DATETIME
3 person_id INTEGER NOT NULL, person_address_id INTEGER NOT NULL, CONSTRAINT PK_person_address PRIMARY KEY (person_
4 _address_id);
5 _id INTEGER NOT NULL, phone_number VARCHAR(55) NOT NULL, CONSTRAINT PK_phone_number PRIMARY KEY (phone_number_id));

```

The status bar at the bottom indicates 'Query succeeded: Affected rows: 0'.

Queries for inserting data into tables

```

INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_zip_postal, address_state_province_county, address_country) VALUES (4, '555', 'azure203Demo', NULL, 'San Francisco', '91001', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_zip_postal, address_state_province_county, address_country) VALUES (1, '555', 'azure203Demo', NULL, 'Los Angeles', '91011', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, address_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, address_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');
```

The screenshot shows the Microsoft Azure portal interface with the 'Query editor (preview)' tab selected. The left sidebar lists various database management options like Overview, Activity log, Tags, etc. The main area displays a query window titled 'Query 1' containing the following SQL code:

```

1  INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_
2  INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_
3  INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_
4  INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_
5
6  INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES (1, 1, 'jon.flanders@n
7  INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES (2, 1, 'jon@notherm
8
9  INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES (4, 3, 'fritz@mail.com
10  INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES (5, null, 'ann@mail.com

```

The status bar at the bottom indicates 'Query succeeded: Affected rows: 19'.

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

The screenshot shows the Microsoft Azure portal interface with the 'Query editor (preview)' tab selected. The left sidebar lists various database management options like Overview, Activity log, Tags, etc. The main area displays a query window titled 'Query 1' containing the following SQL code:

```

1  SELECT * FROM address;

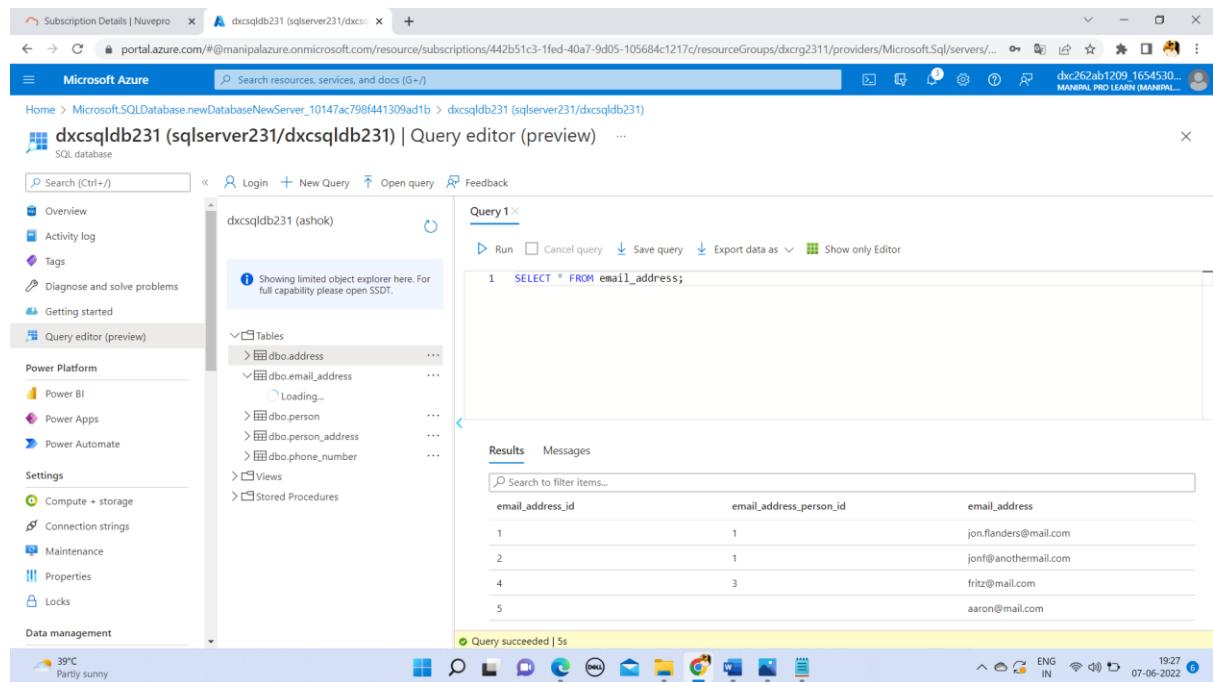
```

The results pane shows the following data:

address_id	address_building_n...	address_street	address_locality	address_city	address_zip_postal
1	555	azure203Demo		Los Angeles	91001
2	555	azure203Demo		Toronto	7777
3	555	azure203Demo	Boonies	Somewhere	11111
4	555	azure203Demo		San Francisco	91001

The status bar at the bottom indicates 'Query succeeded | 4s'.

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



Thank you!

ASHOK KUMAR