1. Explore Core Data Concepts:

Data is a collection of facts such as numbers, descriptions, and observations used in decision making.

\* OLAP vs OLTP:-

To support fast processing, the data in a transactional system is often divided into small pieces.

Splitting tables out into separate groups of columns like this is called normalized.

Normalization can enable a transactional system to use cache & process InMemory. But it can also make queries complex (joins).

Analytical systems are concerned with capturing raw data, and using it to generate insights.

\* Four Stages of Analytical system:-

Data Ingestion ==> Data Transformation/Cleaning => Querying/Analytical Processing => Visualization

\* Transactional workload VS Analytical workload:

A transaction is a sequence of operations that are atomic. Means all operationsin the sequence should be complete successfully, if any operation fails, all operations should be revert back to previous stage. (Ex. bank transactions).

A transactional database must adhere to the ACID (Atomicity, Consistency, Isolation, Durability) properties.

1. Atomicity: Each transaction is treated as a single unit, which either succeeds completely, or fails completely.
2. Consistency: A transaction can only take the data in the database from one valid state to another. (If fund is credited, it should be debited from another account or a record which describe frome where this fund is coming externally ).
3. Isolation: The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized. ie. transferred funds in one account or the other, but not in both, nor in neither.
4. Durability: Once a transaction has been committed, it will remain committed even if there's a system failure or power crash.

Lock vs eventual consistency:-

Many systems implement relational consistency and isolation by applying locks to data when it is updated. The lock prevents another process from reading the data until the lock is released. The lock is only released when the transaction commits or rolls back. Extensive locking can lead to poor performance, while applications wait for locks to be released.

In Distributed dB, locks may be retained for a very long time, especially if there's a network failure between databases at a critical time. To counter this problem, distributive DB implements "eventual consistency." In this form of consistency, as an application writes data, each change is recorded by one server and then propagated to the other servers in the distributed database system asynchronously.

\* Batch vs Stream processing:

Newly arriving data elements are collected into a group. The whole group is then processed at a future time as a batch. Any Problems with data, errors, and program crashes that occur during batch jobs bring the whole process to a halt. The input data must be carefully checked before the job can be run again. Even minor data errors, such as typographical errors in dates, can prevent a batch job from running.

In stream processing, each new piece of data is processed when it arrives. For example, data ingestion is inherently a streaming process.

*Explore Roles & Responsibility:*

\* Common Roles: Database Administrators, Data Engineers, Data Analysts.

DBA Tools : Azure data stdio, SQL Server Management Stdio, Azure portal.

Data Engineering Tools:

*Describe RDBMS Concepts:*

**Index** : Create index on a column to search it fast. However indexes aren't free. An index might consume additional storage space, and each time you insert, update, or delete data in a table, the indexes for that table must be maintained. This additional work can slow down insert, update, and delete operations, and incur additional processing charges. Some relational database management systems also support 'clustered indexes'. A clustered index physically reorganizes a table by the index key (Grouping through Index Key.)

**View :** A view is a virtual table based on the result set of a query.

*Describe Non-RDBMS Concepts:*

\* Semi Structured Data: JSON, Avro, OCR, Parquet

\* Transparent VS Opaque:

DBMS just sees the value as an unstructured block. Only the application understands how the data in the value is structured and what fields it contains.

In Transparent, DBMS understands how the fields in the data are organized. (Ex. RDBMS )

\* NoSQL: Key-Value, Document DB, Column family dB, Graph DB

**Key-Value** : data is oragined with Key and sorted with Key ASC. This is very fast for ingestion.

**Document DB** : Document store entire data of an entity (ie. MongoDB) The focus of a document database is its query capabilities.

**Column DB** : Whole document is devided into column family (ie. one column for name, one is for address info). ie Cassandra.

**Graph DB:**

*Explore Concept of Data Analytics*

\* Data Ingestion :

\* Data Processing :

**ETL :** Suitable for systems that only require simple models, with little dependency between items.

**ELT :** suitable for constructing complex models that depend on multiple items in the database, often using periodic batch processing. ELT is a scalable approach that is suitable for the cloud.

\* Data Analytics :

**Descriptive Analytics** : What has happened.

**Diagnostic Analytics** : Why things happened. This generally occurs in three steps:

Identify anomalies in the data. These may be unexpected changes in a metric or a particular market.

Collect data that's related to these anomalies.

Use statistical techniques to discover relationships and trends that explain these anomalies.

**Predictive Analytics :** What will happen in the future.

**Prescriptive Analytics :** What action should be taken to acheive target or goal.

**Cognitive Analytics :** draw inference from existing data and patterns. (a self-learning feedback loop)

**2.Explore Relational Data in Azure:**

*Explore Relational Data Services in Azure*

\* Pass : You specify resources and Azure will automatically VM, network & other devices for you.

\* Azure data services falls is PaSS CATEGORY.

**\* SQL server on Azure VM:** 'lift-&-Shift', easily migrate on-premise SQL server DB to cloud.

**\* Azure SQL DB**: Scaling up or out will take effect without restarting the SQL database. 3 options are available.

- **Single DB :** Quickly setup & run a single DB. You can also specify a serverless configuration. In this configuration, Microsoft creates its own server, which might be shared by a number of databases belonging to other Azure subscribers. Microsoft ensures the privacy of your database. Your database automatically scales and resources are allocated or deallocated as required.

- **Elastic Pool :** Pool of servers which shares same resources. This model is useful if you have databases with resource requirements that vary over time, and can help you to reduce costs. Elastic Pool enables you to use the resources available in the pool, and then release the resources once processing has completed.

- **Managed Instance:** A fully controllable instance of SQL Server in the cloud. You can install multiple databases on the same instance. You have complete control over this instance, much as you would for an on-premises server. The Managed instance service automates backups, software patching, database monitoring, and other general tasks.

*Azure RDBMS Offerings:*

MySQL :

MariaDB :

Azure SQL:

PostGreSQL : Single server, HyperScale(distributed via partition/shrading key).

*Provising & Deploying RDBMS*

\* Provision RDBMS on Azure

\* Configuring Services related to RDBMS

- Configure connectivity to virtual network & on-premise servers.

- Configure private endpoints.( )

- Configure Authentication (ADDS)

Querying RDBMS in Azure

\* DML Commands: SIDU ( SELECT, INSERT, DELETE, UPDATE )

\* DDL Commands: CARD ( CREATE, ALTER, RENAME, DROP )

Running Query - sqlcmd : a cmd utility to run sql queries from cloud shell or from local cmd.

- Azure data Stdio : Cross platform GUI tool

- SQL Server Management Stdio :

- SQl Server data tool in Visual Stdio

**3.Explore Non-Relational Data in Azure:**

*Non Relational Offerring in Azure*

**\* Azure Table storage:**

- Key Value based => unique keys with 'array of Values'.

- Azure Table Storage have no concept of relationships, stored procedures, secondary indexes, or foreign keys.

- You can group rows based on common properties index via a partiton key. (partition + row = clustered Index)

- Partition key => Row key => ARRAY OF VALUES.

- Items in the same partition are AUTO INDEXED & stored in row key order.

**\* Point Query VS Range Query**

- Point Query retrieve a single row; Range query retrieve a contiguous block of rows.

- A table can have 252 columns(arrays of value), a single column can hold data to 64 KB.

**\* Blob storage:** binary object. For unsuctured data.

**Block Blob :** To store large, blob that donot change frequently. (50,000 blocks of upto 100 MB = 4.7 TB MAX)

**Page Blob** : Random read/write. (512 byte pages, max 8 Tb) => AZure use this for VM storage.

**Append Blobs :** Max 195 Gb, average size is 4 MB. Only Append, no update or delete.

- In Azure , you create blobs inside containers, and you can organize blobs in a hierarchy of folders.

- Blob storage provides three access tiers, which help to balance access latency and storage cost: Hot, Cold & Archieve

You can create lifecycle management policies for blobs in a storage account. A lifecycle management policy can automatically move a blob from Hot to Cool, to the Archive tier, as it ages and is used less frequently (policy is based on the number of days since modification). A lifecycle management policy can also arrange to delete outdated blobs.

Other features available with Azure Blob storage include:

* Versioning: You can maintain and restore earlier versions of a blob.
* Soft delete: Recover a blob that has been removed or overwritten, by accident or otherwise.
* Snapshots: A snapshot is a read-only version of a blob at a particular point in time.
* Change Feed: The change feed for a blob provides an ordered, read-only, record of the updates made to a blob. You can use the change feed to monitor these changes, and perform operations:

**\* File Storage:** File Storage enables you to create files shares in the cloud, and access these file shares from anywhere with internet connection. It works on SMB 3.0 MAX size of a single file is 1 TB, it supports 2000 concurent connections to a single file.

- Tools for file storage: AzCopy, Azure File Synce Copy, Azure Portal.

**NOTE:** Don't use Azure File Storage for files that can be written by multiple concurrent processes simultaneously. Multiple writers require careful synchronization, otherwise the changes made by one process can be overwritten by another.

**\* Cosmos DB:** A Multi-model NoSQL database management system. Cosmos DB manages data as a partitioned set of documents. A document can hold up to 2 MB of data.

- Documents in a Cosmos DB are organized into containers. The documents in a container are grouped together into partitions. A partition holds a set of documents that share a common partition key.

- Cosmos DB & Table Storage both uses partiton ID. Unlike Azure Table storage, documents in a Cosmos DB partition aren't sorted by ID. Instead, Cosmos DB maintains a separate index. This index contains not only the document IDs, but also tracks the value of every other field in each document. This index is created and maintained automatically.

- Cosmos DB automatically allocates space in a container for your partitions, and each partition can grow up to 10 GB in size.

**NOTE:** The primary purpose of the Table, MongoDB, Cassandra, and Gremlin APIs is to support existing applications. If you are building a new application and database, you should use the SQL API.