**Module1. Explore Core Data Concepts**

Structured data is data that adheres to a fixed schema, so all the data has the same fields or properties.

Semi-structured data is information that has some structure, but which allows for some variation between entity instances. For example, while most customers may have an email address, some might have multiple email addresses, and some might have none at all.

Not all data is structured or even semi-structured. For example, documents, images, audio and video data, and binary files might not have a specific structure. This kind of data is referred to as unstructured data.

Some common optimized file formats you might see include **Avro, ORC,** and**Parquet**:

* ***Avro*** is a row-based format. It was created by Apache. Each record contains a header that describes the structure of the data in the record. This header is stored as JSON. The data is stored as binary information. An application uses the information in the header to parse the binary data and extract the fields it contains. Avro is a good format for compressing data and minimizing storage and network bandwidth requirements.
* ***ORC*** (Optimized Row Columnar format) organizes data into columns rather than rows. It was developed by HortonWorks for optimizing read and write operations in Apache Hive (Hive is a data warehouse system that supports fast data summarization and querying over large datasets). An ORC file contains *stripes* of data. Each stripe holds the data for a column or set of columns. A stripe contains an index into the rows in the stripe, the data for each row, and a footer that holds statistical information (count, sum, max, min, and so on) for each column.
* ***Parquet*** is another columnar data format. It was created by Cloudera and Twitter. A Parquet file contains row groups. Data for each column is stored together in the same row group. Each row group contains one or more chunks of data. A Parquet file includes metadata that describes the set of rows found in each chunk. An application can use this metadata to quickly locate the correct chunk for a given set of rows and retrieve the data in the specified columns for these rows. Parquet specializes in storing and processing nested data types efficiently. It supports very efficient compression and encoding schemes.

There are four common types of non-relational database commonly in use.

**Key-value databases** in which each record consists of a unique key and an associated value.

**Document databases**, which are a specific form of key-value database in which the value is a JSON document.

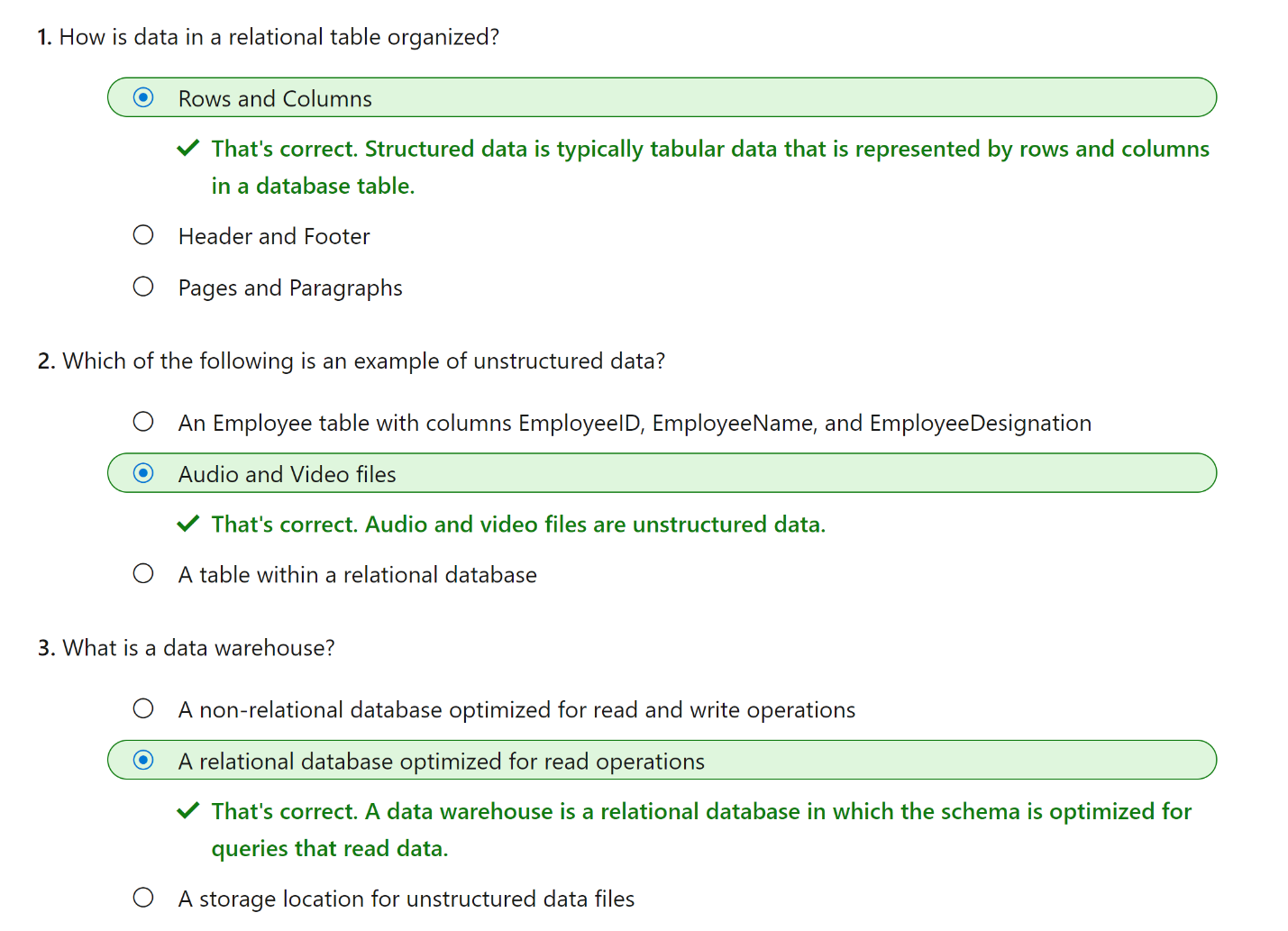
**Column family databases**, which store tabular data comprising rows and columns, but you can divide the columns into groups known as column-families. Each column family holds a set of columns that are logically related together.

**Graph databases**, which store entities as nodes with links to define relationships between them.

OLTP systems enforce transactions that support so-called ACID semantics:

* **Atomicity** – each transaction is treated as a single unit, which succeeds completely or fails completely.
* **Consistency** – transactions can only take the data in the database from one valid state to another.
* **Isolation** – concurrent transactions cannot interfere with one another, and must result in a consistent database state.
* **Durability** – when a transaction has been committed, it will remain committed.
* *Data lakes* are common in large-scale data analytical processing scenarios, where a large volume of file-based data must be collected and analyzed.
* *Data warehouses* are an established way to store data in a relational schema that is optimized for read operations – primarily queries to support reporting and data visualization.

An OLAP model is an aggregated type of data storage that is optimized for analytical workloads. Data aggregations are across dimensions at different levels, enabling you to drill up/down to view aggregations at multiple hierarchical levels



Module2 - **Explore data roles and services**

The three key job roles that deal with data in most organizations are:

* **Database administrators** manage databases, assigning permissions to users, storing backup copies of data and restore data in the event of a failure.
* **Data engineers** manage infrastructure and processes for data integration across the organization, applying data cleaning routines, identifying data governance rules, and implementing pipelines to transfer and transform data between systems.
* **Data analysts** explore and analyze data to create visualizations and charts that enable organizations to make informed decisions.

Azure Data Factory is an Azure service that enables you to define and schedule data pipelines to transfer and transform data. You can integrate your pipelines with other Azure services, enabling you to ingest data from cloud data stores, process the data using cloud-based compute, and persist the results in another data store.

Azure Synapse Analytics is a comprehensive, unified data analytics solution that provides a single service interface for multiple analytical capabilities, including:

* **Pipelines** - based on the same technology as Azure Data Factory.
* **SQL** - a highly scalable SQL database engine, optimized for data warehouse workloads.
* **Apache Spark** - an open-source distributed data processing system that supports multiple programming languages and APIs, including Java, Scala, Python, and SQL.
* **Azure Synapse Data Explorer** - a high-performance data analytics solution that is optimized for real-time querying of log and telemetry data using Kusto Query Language (KQL).

Azure Databricks is an Azure-integrated version of the popular Databricks platform, which combines the Apache Spark data processing platform with SQL database semantics and an integrated management interface to enable large-scale data analytics.

Azure HDInsight is an Azure service that provides Azure-hosted clusters for popular Apache open-source big data processing technologies, including:

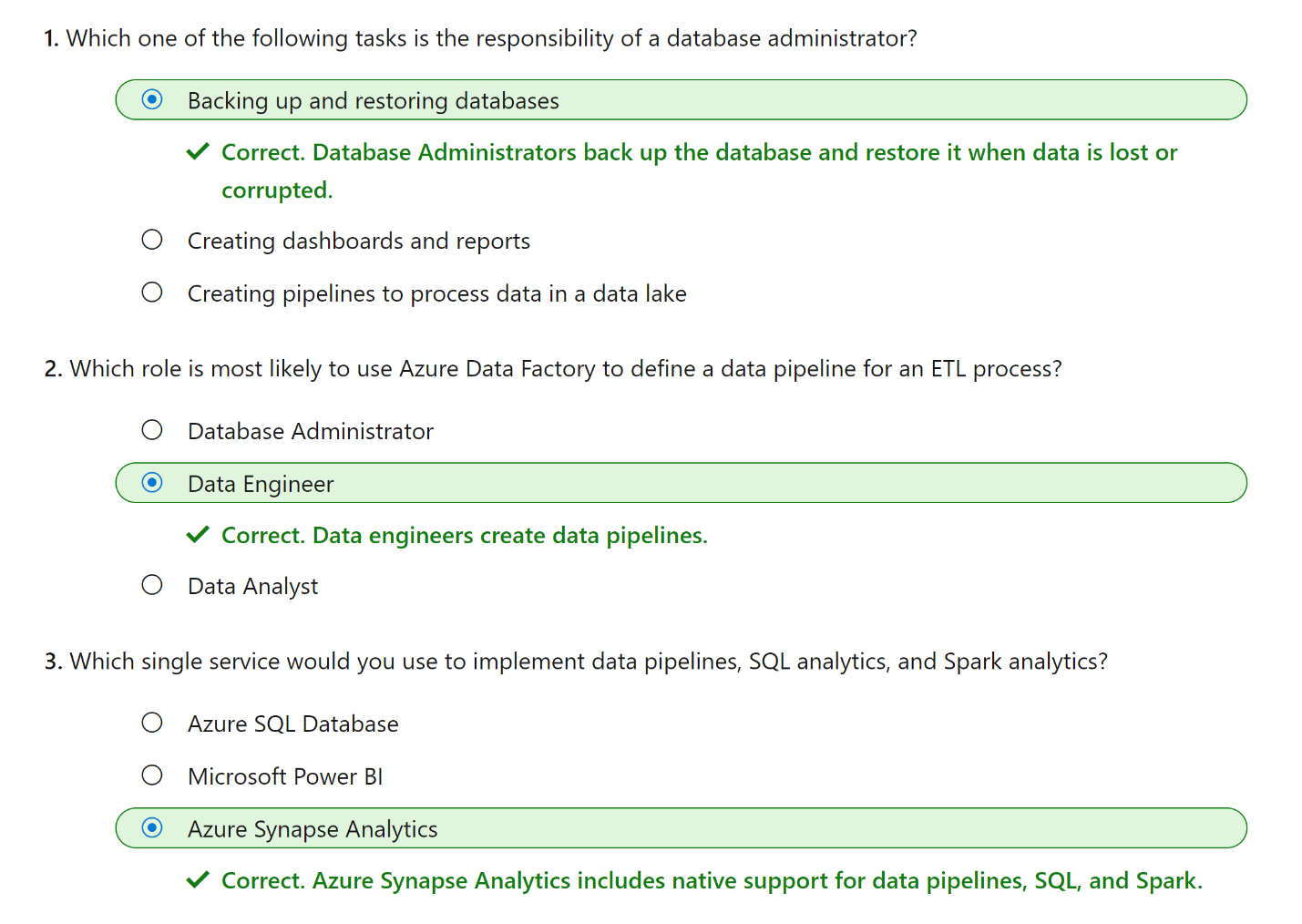
* **Apache Spark** - a distributed data processing system that supports multiple programming languages and APIs, including Java, Scala, Python, and SQL.
* **Apache Hadoop** - a distributed system that uses *MapReduce* jobs to process large volumes of data efficiently across multiple cluster nodes. MapReduce jobs can be written in Java or abstracted by interfaces such as Apache Hive - a SQL-based API that runs on Hadoop.
* **Apache HBase** - an open-source system for large-scale NoSQL data storage and querying.
* **Apache Kafka** - a message broker for data stream processing.
* **Apache Storm** - an open-source system for real-time data processing through a topology of *spouts* and *bolts*.

Azure Stream Analytics is a real-time stream processing engine that captures a stream of data from an input, applies a query to extract and manipulate data from the input stream, and writes the results to an output for analysis or further processing.

Azure Data Explorer is a standalone service that offers the same high-performance querying of log and telemetry data as the Azure Synapse Data Explorer runtime in Azure Synapse Analytics.

Microsoft Purview provides a solution for enterprise-wide data governance and discoverability. You can use Microsoft Purview to create a map of your data and track data lineage across multiple data sources and systems, enabling you to find trustworthy data for analysis and reporting.

Microsoft Power BI is a platform for analytical data modeling and reporting that data analysts can use to create and share interactive data visualizations. Power BI reports can be created by using the Power BI Desktop application, and then published and delivered through web-based reports and apps in the Power BI service, as well as in the Power BI mobile app.



**Module3 – Explore Relational database services in Azure**

Azure SQL is a collective term for a family of Microsoft SQL Server based database services in Azure. Specific Azure SQL services include:

* **SQL Server on Azure Virtual Machines (VMs)** - A virtual machine running in Azure with an installation of SQL Server. The use of a VM makes this option an infrastructure-as-a-service (IaaS) solution that virtualizes hardware infrastructure for compute, storage, and networking in Azure; making it a great option for "lift and shift" migration of existing on-premises SQL Server installations to the cloud.
* **Azure SQL Managed Instance** - A platform-as-a-service (PaaS) option that provides near-100% compatibility with on-premises SQL Server instances while abstracting the underlying hardware and operating system. The service includes automated software update management, backups, and other maintenance tasks, reducing the administrative burden of supporting a database server instance.
* **Azure SQL Database** - A fully managed, highly scalable PaaS database service that is designed for the cloud. This service includes the core database-level capabilities of on-premises SQL Server, and is a good option when you need to create a new application in the cloud.
* **Azure SQL Edge** - A SQL engine that is optimized for Internet-of-things (IoT) scenarios that need to work with streaming time-series data.

SQL Managed Instance automates backups, software patching, database monitoring, and other general tasks, but you have full control over security and resource allocation for your databases.

All communications are encrypted and signed using certificates. To check the trustworthiness of communicating parties, managed instances constantly verify these certificates through certificate revocation lists. If the certificates are revoked, the managed instance closes the connections to protect the data.

MariaDB is a newer database management system, created by the original developers of MySQL. The database engine has since been rewritten and optimized to improve performance. MariaDB offers compatibility with Oracle Database (another popular commercial database management system). One notable feature of MariaDB is its built-in support for temporal data. A table can hold several versions of data, enabling an application to query the data as it appeared at some point in the past.

PostgreSQL is a hybrid relational-object database. You can store data in relational tables, but a PostgreSQL database also enables you to store custom data types, with their own non-relational properties. The database management system is extensible; you can add code modules to the database, which can be run by queries

Azure Database for MySQL has two deployment options: Single Server and Flexible Server. Azure Database for MySQL Flexible Server is a fully managed database as a service offering, with predictable performance and dynamic scalability. Flexible Server provides more granular control and flexibility over database management functions and configuration settings. Single servers are best for existing applications already using single server.

You get the following features with Azure Database for MySQL:

* High availability features built-in.
* Predictable performance.
* Easy scaling that responds quickly to demand.
* Secure data, both at rest and in motion.
* Automatic backups and point-in-time restore for the last 35 days.
* Enterprise-level security and compliance with legislation.

The system uses pay-as-you-go pricing so you only pay for what you use.

Azure Database for MySQL servers provides monitoring functionality to add alerts, and to view metrics and logs.

Azure Database for MariaDB delivers:

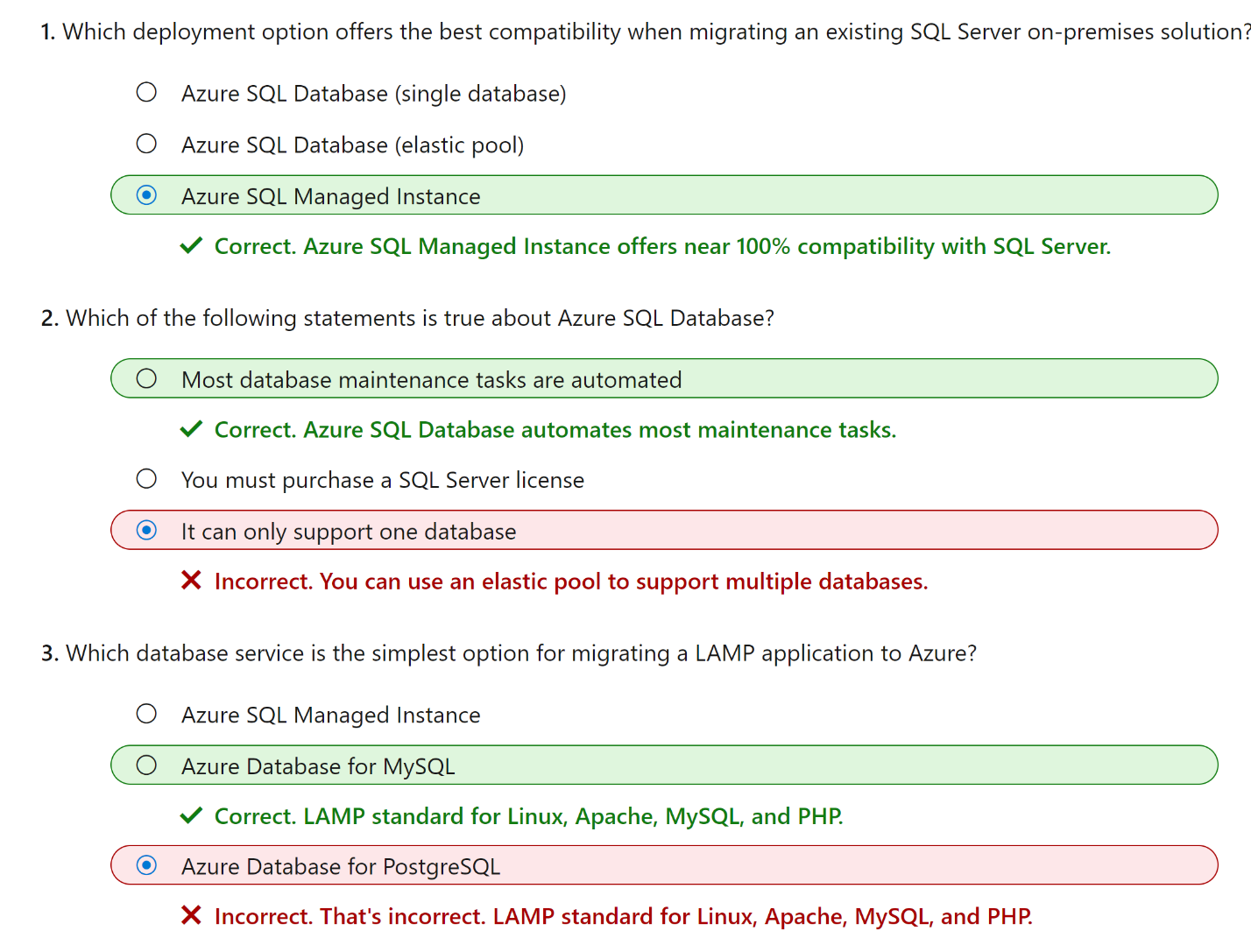
* Built-in high availability with no additional cost.
* Predictable performance, using inclusive pay-as-you-go pricing.
* Scaling as needed within seconds.
* Secured protection of sensitive data at rest and in motion.
* Automatic backups and point-in-time-restore for up to 35 days.
* Enterprise-grade security and compliance.

Some features of on-premises PostgreSQL databases aren't available in Azure Database for PostgreSQL. These features are mostly concerned with the extensions that users can add to a database to perform specialized tasks, such as writing stored procedures in various programming languages (other than pgsql, which is available), and interacting directly with the operating system. Azure Database for PostgreSQL has three deployment options: Single Server, Flexible Server, and Hyperscale.

The single-server deployment option for PostgreSQL provides similar benefits as Azure Database for MySQL. You choose from three pricing tiers: Basic, General Purpose, and Memory Optimized. Each tier supports different numbers of CPUs, memory, and storage sizes; you select one based on the load you expect to support.

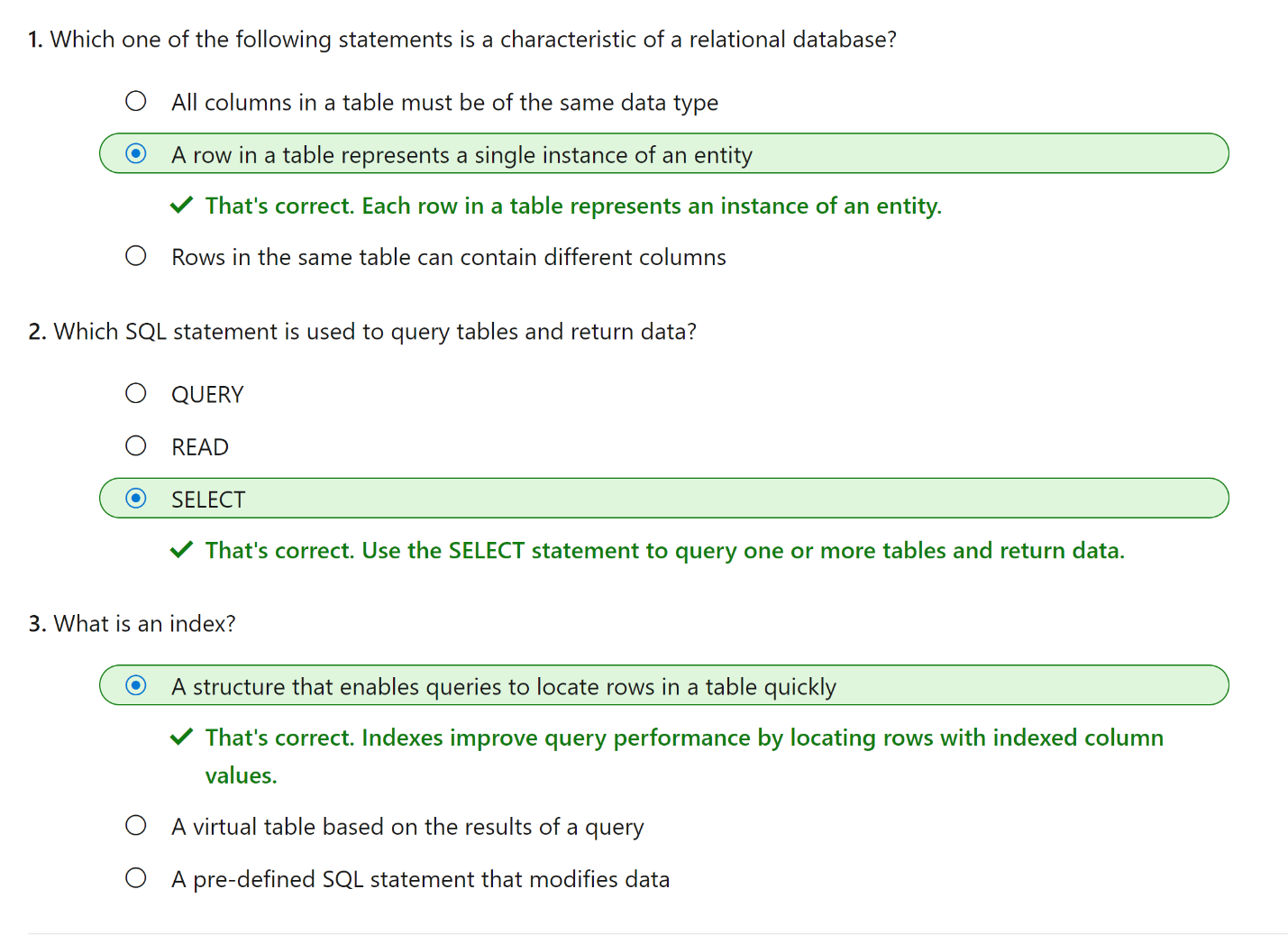
The flexible-server deployment option for PostgreSQL is a fully managed database service. It provides more control and server configuration customizations, and has better cost optimization controls.

Hyperscale (Citus) is a deployment option that scales queries across multiple server nodes to support large database loads. Your database is split across nodes. Data is split into chunks based on the value of a partition key or sharding key. Consider using this deployment option for the largest database PostgreSQL deployments in the Azure Cloud

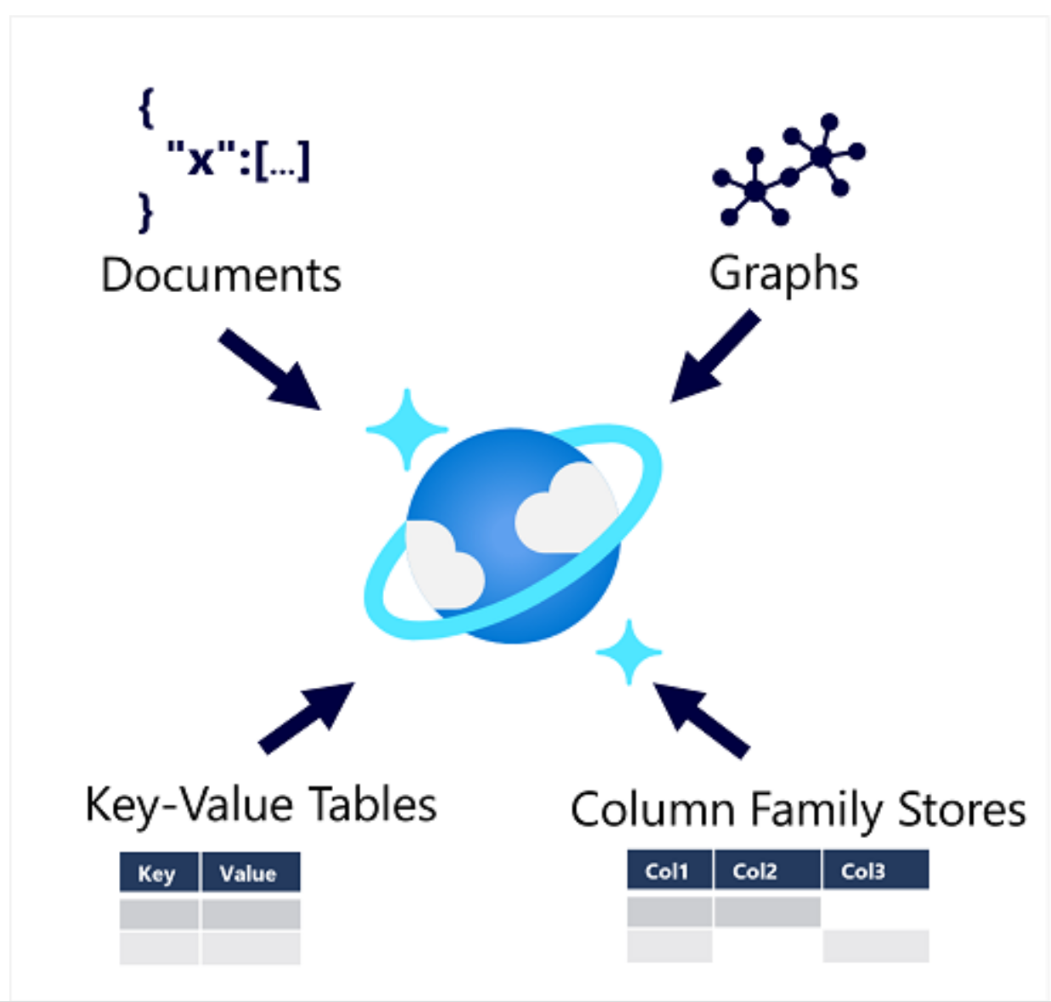


# Module4 - Explore fundamental relational data concepts

Normalization is a term used by database professionals for a schema design process that minimizes data duplication and enforces data integrity.



# Module5 - Explore fundamentals of Azure Cosmos DB



Cosmos DB uses indexes and partitioning to provide fast read and write performance and can scale to massive volumes of data. You can enable multi-region writes, adding the Azure regions of your choice to your Cosmos DB account so that globally distributed users can each work with data in their local replica

Cosmos DB is a highly scalable database management system. Cosmos DB automatically allocates space in a container for your partitions, and each partition can grow up to 10 GB in size. Indexes are created and maintained automatically. There's virtually no administrative overhead.

Cosmos DB is highly suitable for the following scenarios:

* *IoT and telematics*. These systems typically ingest large amounts of data in frequent bursts of activity. Cosmos DB can accept and store this information quickly. The data can then be used by analytics services, such as Azure Machine Learning, Azure HDInsight, and Power BI. Additionally, you can process the data in real-time using Azure Functions that are triggered as data arrives in the database.
* *Retail and marketing*. Microsoft uses Cosmos DB for its own e-commerce platforms that run as part of Windows Store and Xbox Live. It's also used in the retail industry for storing catalog data and for event sourcing in order processing pipelines.
* *Gaming*. The database tier is a crucial component of gaming applications. Modern games perform graphical processing on mobile/console clients, but rely on the cloud to deliver customized and personalized content like in-game stats, social media integration, and high-score leaderboards. Games often require single-millisecond latencies for reads and write to provide an engaging in-game experience. A game database needs to be fast and be able to handle massive spikes in request rates during new game launches and feature updates.
* *Web and mobile applications*. Azure Cosmos DB is commonly used within web and mobile applications, and is well suited for modeling social interactions, integrating with third-party services, and for building rich personalized experiences. The Cosmos DB SDKs can be used to build rich iOS and Android applications using the popular Xamarin framework.

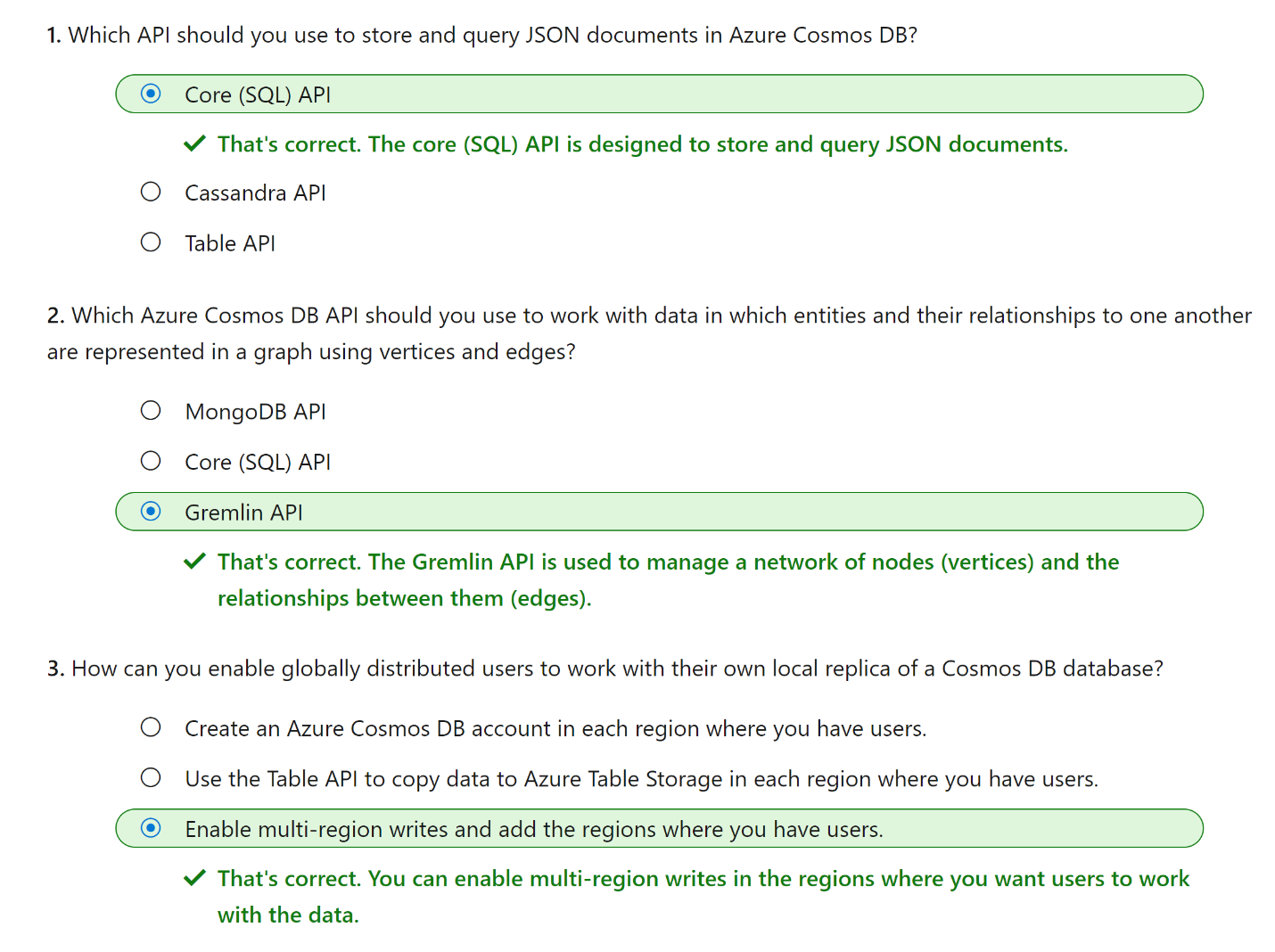
The native API in Cosmos DB manages data in JSON document format, and despite being a NoSQL data storage solution, uses SQL syntax to work with the data.

MongoDB is a popular open source database in which data is stored in Binary JSON (BSON) format. The Azure Cosmos DB MongoDB API enables developers to use MongoDB client libraries and code to work with data in Azure Cosmos DB.

The Table API is used to work with data in key-value tables, similar to Azure Table Storage. The Azure Cosmos DB Table API offers greater scalability and performance than Azure Table Storage.

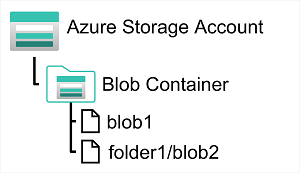
The Cassandra API is compatible with Apache Cassandra, which is a popular open source database that uses a column-family storage structure. Column families are tables, similar to those in a relational database, with the exception that it's not mandatory for every row to have the same columns.

The Gremlin API is used with data in a graph structure; in which entities are defined as vertices that form nodes in connected graph. Nodes are connected by edges that represent relationships



# Module6 - Explore Azure Storage for non-relational data

Azure Blob Storage is a service that enables you to store massive amounts of unstructured data as binary large objects, or blobs, in the cloud. Blobs are an efficient way to store data files in a format that is optimized for cloud-based storage, and applications can read and write them by using the Azure blob storage API.



In an Azure storage account, you store blobs in containers. A container provides a convenient way of grouping related blobs together. You control who can read and write blobs inside a container at the container level.

Azure Blob Storage supports three different types of blob:

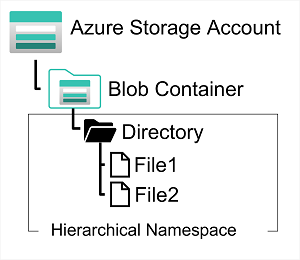
* **Block blobs**. A block blob is handled as a set of blocks. Each block can vary in size, up to 100 MB. A block blob can contain up to 50,000 blocks, giving a maximum size of over 4.7 TB. The block is the smallest amount of data that can be read or written as an individual unit. Block blobs are best used to store discrete, large, binary objects that change infrequently.
* **Page blobs**. A page blob is organized as a collection of fixed size 512-byte pages. A page blob is optimized to support random read and write operations; you can fetch and store data for a single page if necessary. A page blob can hold up to 8 TB of data. Azure uses page blobs to implement virtual disk storage for virtual machines.
* **Append blobs**. An append blob is a block blob optimized to support append operations. You can only add blocks to the end of an append blob; updating or deleting existing blocks isn't supported. Each block can vary in size, up to 4 MB. The maximum size of an append blob is just over 195 GB.

Blob storage provides three access tiers, which help to balance access latency and storage cost:

* The *Hot* tier is the default. You use this tier for blobs that are accessed frequently. The blob data is stored on high-performance media.
* The *Cool* tier has lower performance and incurs reduced storage charges compared to the Hot tier. Use the Cool tier for data that is accessed infrequently. It's common for newly created blobs to be accessed frequently initially, but less so as time passes. In these situations, you can create the blob in the Hot tier, but migrate it to the Cool tier later. You can migrate a blob from the Cool tier back to the Hot tier.
* The *Archive* tier provides the lowest storage cost, but with increased latency. The Archive tier is intended for historical data that mustn't be lost, but is required only rarely. Blobs in the Archive tier are effectively stored in an offline state. Typical reading latency for the Hot and Cool tiers is a few milliseconds, but for the Archive tier, it can take hours for the data to become available. To retrieve a blob from the Archive tier, you must change the access tier to Hot or Cool. The blob will then be rehydrated. You can read the blob only when the rehydration process is complete.

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, and then 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.

Azure Data Lake Store (Gen1) is a separate service for hierarchical data storage for analytical data lakes, often used by so-called big data analytical solutions that work with structured, semi-structured, and unstructured data stored in files. Azure Data Lake Storage Gen**2** is a newer version of this service that is integrated into Azure Storage; enabling you to take advantage of the scalability of blob storage and the cost-control of storage tiers, combined with the hierarchical file system capabilities and compatibility with major analytics systems of Azure Data Lake Store.

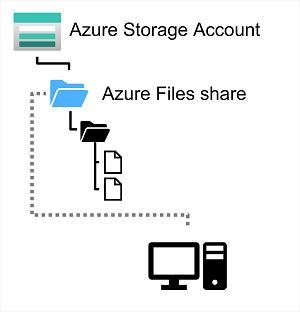


Systems like Hadoop in Azure HDInsight, Azure Databricks, and Azure Synapse Analytics can mount a distributed file system hosted in Azure Data Lake Store Gen2 and use it to process huge volumes of data.

To create an Azure Data Lake Store Gen2 files system, you must enable the **Hierarchical Namespace** option of an Azure Storage account.

Many on-premises systems comprising a network of in-house computers make use of file shares. A file share enables you to store a file on one computer, and grant access to that file to users and applications running on other computers. This strategy can work well for computers in the same local area network, but doesn't scale well as the number of users increases, or if users are located at different sites.

Azure Files is essentially a way to create cloud-based network shares, such as you typically find in on-premises organizations to make documents and other files available to multiple users. By hosting file shares in Azure, organizations can eliminate hardware costs and maintenance overhead, and benefit from high availability and scalable cloud storage for files.



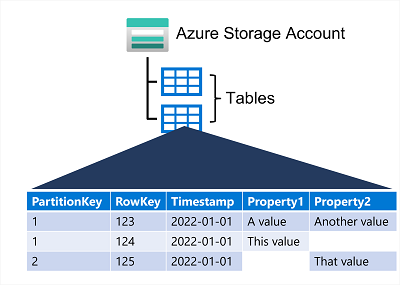
You create Azure File storage in a storage account. Azure Files enables you to share up to 100 TB of data in a single storage account. This data can be distributed across any number of file shares in the account. The maximum size of a single file is 1 TB, but you can set quotas to limit the size of each share below this figure. Currently, Azure File Storage supports up to 2000 concurrent connections per shared file.

Azure File Storage offers two performance tiers. The *Standard* tier uses hard disk-based hardware in a datacenter, and the *Premium* tier uses solid-state disks. The *Premium* tier offers greater throughput, but is charged at a higher rate.

Azure Files supports two common network file sharing protocols:

* *Server Message Block* (SMB) file sharing is commonly used across multiple operating systems (Windows, Linux, macOS).
* *Network File System* (NFS) shares are used by some Linux and macOS versions. To create an NFS share, you must use a premium tier storage account and create and configure a virtual network through which access to the share can be controlled.

Azure Table Storage is a NoSQL storage solution that makes use of tables containing key/value data items. Each item is represented by a row that contains columns for the data fields that need to be stored.

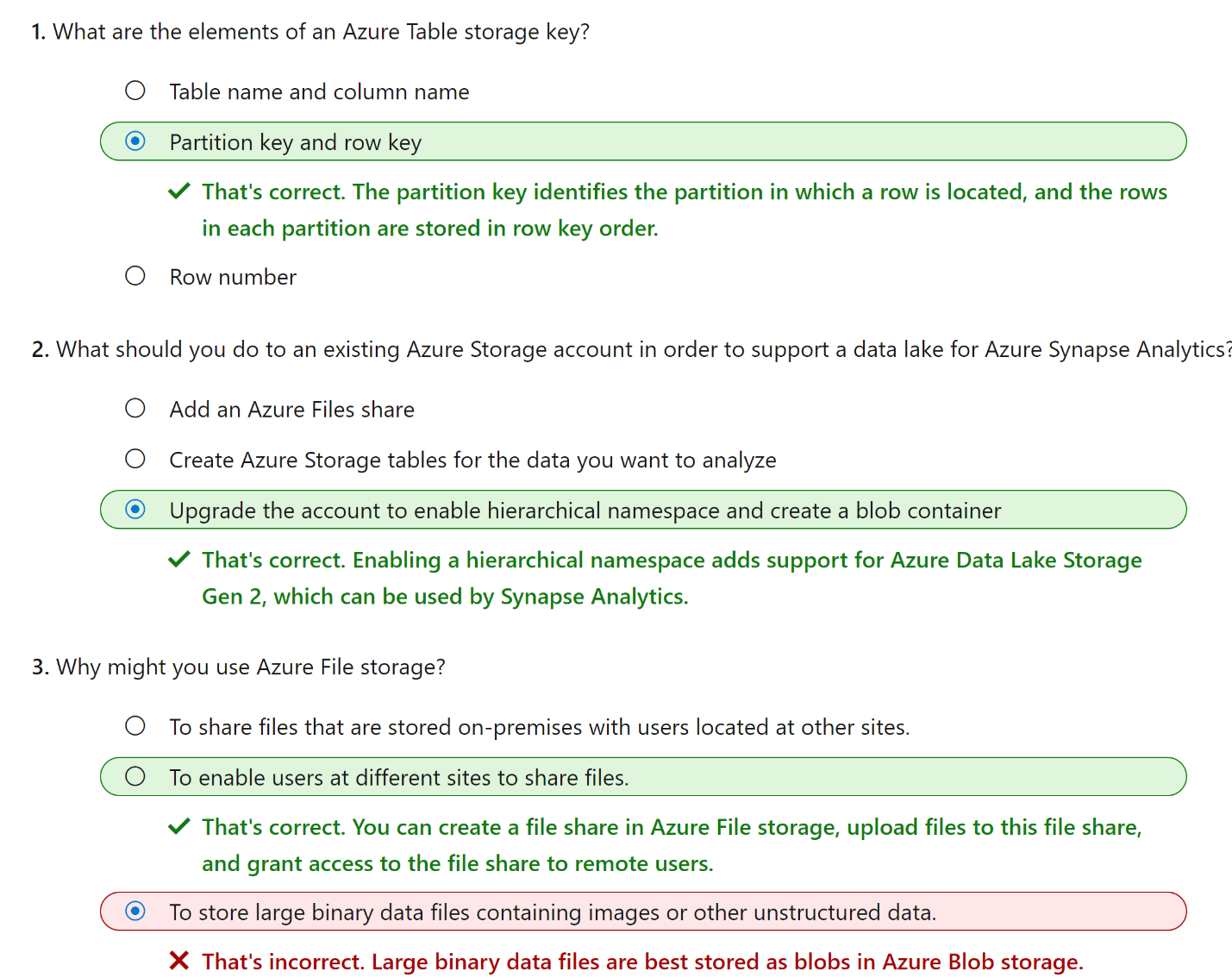


All rows in a table must have a unique key (composed of a partition key and a row key), and when you modify data in a table, a timestamp column records the date and time the modification was made; but other than that, the columns in each row can vary. Azure Table Storage tables have no concept of foreign keys, relationships, stored procedures, views, or other objects you might find in a relational database. Data in Azure Table storage is usually denormalized, with each row holding the entire data for a logical entity.

To help ensure fast access, Azure Table Storage splits a table into partitions. Partitioning is a mechanism for grouping related rows, based on a common property or partition key. Rows that share the same partition key will be stored together. Partitioning not only helps to organize data, it can also improve scalability and performance in the following ways:

* Partitions are independent from each other, and can grow or shrink as rows are added to, or removed from, a partition. A table can contain any number of partitions.
* When you search for data, you can include the partition key in the search criteria. This helps to narrow down the volume of data to be examined, and improves performance by reducing the amount of I/O (input and output operations, or *reads* and *writes*) needed to locate the data.

The key in an Azure Table Storage table comprises two elements; the partition key that identifies the partition containing the row, and a row key that is unique to each row in the same partition. Items in the same partition are stored in row key order. If an application adds a new row to a table, Azure ensures that the row is placed in the correct position in the table. This scheme enables an application to quickly perform *point* queries that identify a single row, and *range* queries that fetch a contiguous block of rows in a partition.



# Module7 - Explore fundamentals of real-time analytics

There are two general ways to process data:

* *Batch processing*, in which multiple data records are collected and stored before being processed together in a single operation.
* *Stream processing*, in which a source of data is constantly monitored and processed in real time as new data events occur.

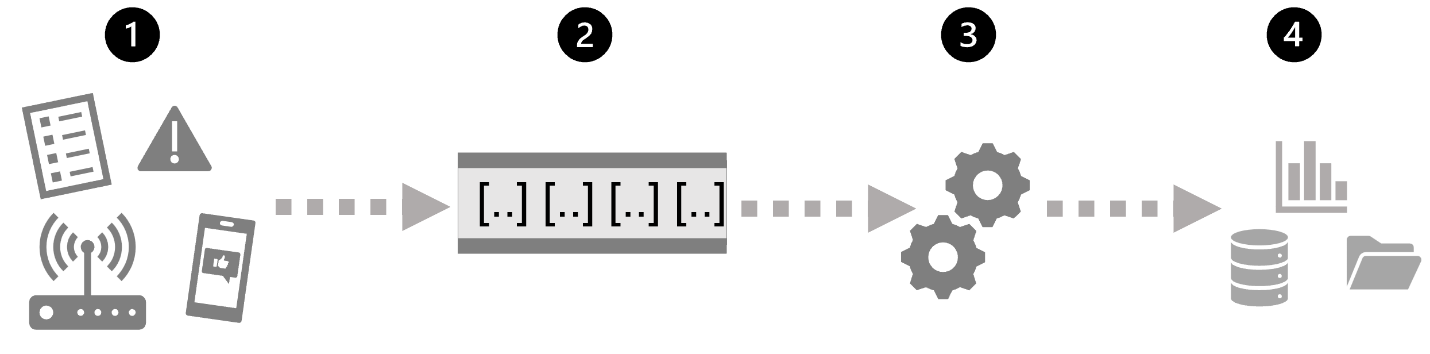
In batch processing, newly arriving data elements are collected and stored, and the whole group is processed together as a batch. Exactly when each group is processed can be determined in a number of ways. For example, you can process data based on a scheduled time interval (for example, every hour), or it could be triggered when a certain amount of data has arrived, or as the result of some other event.

In stream processing, each new piece of data is processed when it arrives. Unlike batch processing, there's no waiting until the next batch processing interval - data is processed as individual units in real-time rather than being processed a batch at a time. Stream data processing is beneficial in scenarios where new, dynamic data is generated on a continual basis.

Apart from the way in which batch processing and streaming processing handle data, there are other differences:

* *Data scope*: Batch processing can process all the data in the dataset. Stream processing typically only has access to the most recent data received, or within a rolling time window (the last 30 seconds, for example).
* *Data size*: Batch processing is suitable for handling large datasets efficiently. Stream processing is intended for individual records or *micro batches* consisting of few records.
* *Performance*: *Latency* is the time taken for the data to be received and processed. The latency for batch processing is typically a few hours. Stream processing typically occurs immediately, with latency in the order of seconds or milliseconds.
* *Analysis*: You typically use batch processing to perform complex analytics. Stream processing is used for simple response functions, aggregates, or calculations such as rolling averages.

A high-level architecture for stream processing looks like this:



1. An event generates some data. This might be a signal being emitted by a sensor, a social media message being posted, a log file entry being written, or any other occurrence that results in some digital data.
2. The generated data is captured in a streaming *source* for processing. In simple cases, the source may be a folder in a cloud data store or a table in a database. In more robust streaming solutions, the source may be a "queue" that encapsulates logic to ensure that event data is processed in order and that each event is processed only once.
3. The event data is processed, often by a perpetual query that operates on the event data to select data for specific types of events, project data values, or aggregate data values over temporal (time-based) periods (or *windows*) - for example, by counting the number of sensor emissions per minute.
4. The results of the stream processing operation are written to an output (or *sink*), which may be a file, a database table, a real-time visual dashboard, or another queue for further processing by a subsequent downstream query

Microsoft Azure supports multiple technologies that you can use to implement real-time analytics of streaming data, including:

* **Azure Stream Analytics**: A platform-as-a-service (PaaS) solution that you can use to define *streaming jobs* that ingest data from a streaming source, apply a perpetual query, and write the results to an output.
* **Spark Structured Streaming**: An open-source library that enables you to develop complex streaming solutions on Apache Spark based services, including **Azure Synapse Analytics**, **Azure Databricks**, and **Azure HDInsight**.
* **Azure Data Explorer**: A high-performance database and analytics service that is optimized for ingesting and querying batch or streaming data with a time-series element, and which can be used as a standalone Azure service or as an **Azure Synapse Data Explorer** runtime in an Azure Synapse Analytics workspace.

### Sources for stream processing

The following services are commonly used to ingest data for stream processing on Azure:

* **Azure Event Hubs**: A data ingestion service that you can use to manage queues of event data, ensuring that each event is processed in order, exactly once.
* **Azure IoT Hub**: A data ingestion service that is similar to Azure Event Hubs, but which is optimized for managing event data from Internet-of-things (IoT) devices.
* **Azure Data Lake Store Gen 2**: A highly scalable storage service that is often used in batch processing scenarios, but which can also be used as a source of streaming data.
* **Apache Kafka**: An open-source data ingestion solution that is commonly used together with Apache Spark. You can use Azure HDInsight to create a Kafka cluster.

### Sinks for stream processing

The output from stream processing is often sent to the following services:

* **Azure Event Hubs**: Used to queue the processed data for further downstream processing.
* **Azure Data Lake Store Gen 2** or **Azure blob storage**: Used to persist the processed results as a file.
* **Azure SQL Database** or **Azure Synapse Analytics**, or **Azure Databricks**: Used to persist the processed results in a database table for querying and analysis.
* **Microsoft Power BI**: Used to generate real time data visualizations in reports and dashboards.

Azure Stream Analytics is a great technology choice when you need to continually capture data from a streaming source, filter or aggregate it, and send the results to a data store or downstream process for analysis and reporting.

The easiest way to use Azure Stream Analytics is to create a Stream Analytics job in an Azure subscription, configure its input(s) and output(s), and define the query that the job will use to process the data. The query is expressed using structured query language (SQL) syntax, and can incorporate static reference data from multiple data sources to supply lookup values that can be combined with the streaming data ingested from an input.

If your stream process requirements are complex or resource-intensive, you can create a Stream Analysis cluster, which uses the same underlying processing engine as a Stream Analytics job, but in a dedicated tenant (so your processing is not affected by other customers) and with configurable scalability that enables you to define the right balance of throughput and cost for your specific scenario.

Apache Spark is a distributed processing framework for large scale data analytics. You can use Spark on Microsoft Azure in the following services:

* Azure Synapse Analytics
* Azure Databricks
* Azure HDInsight

Spark can be used to run code (usually written in Python, Scala, or Java) in parallel across multiple cluster nodes, enabling it to process very large volumes of data efficiently. Spark can be used for both batch processing and stream processing.

To process streaming data on Spark, you can use the Spark Structured Streaming library, which provides an application programming interface (API) for ingesting, processing, and outputting results from perpetual streams of data.

Spark Structured Streaming is built on a ubiquitous structure in Spark called a dataframe, which encapsulates a table of data. You use the Spark Structured Streaming API to read data from a real-time data source, such as a Kafka hub, a file store, or a network port, into a "boundless" dataframe that is continually populated with new data from the stream. You then define a query on the dataframe that selects, projects, or aggregates the data - often in temporal windows. The results of the query generate another dataframe, which can be persisted for analysis or further processing.

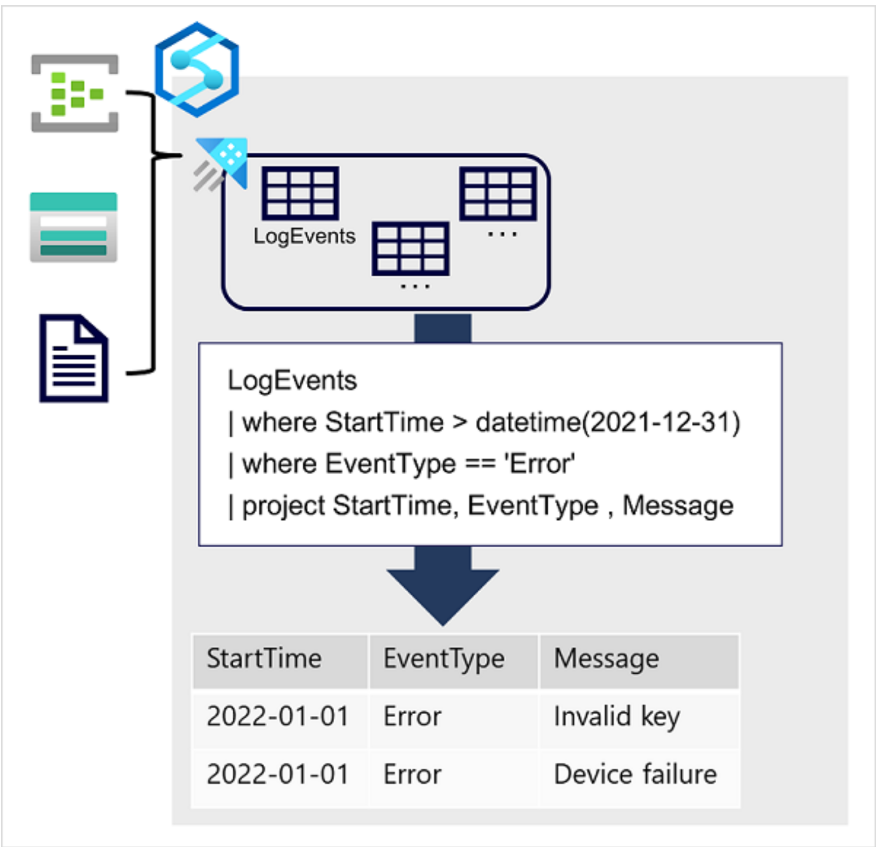
Spark Structured Streaming is a great choice for real-time analytics when you need to incorporate streaming data into a Spark based data lake or analytical data store.

Delta Lake is an open-source storage layer that adds support for transactional consistency, schema enforcement, and other common data warehousing features to data lake storage. It also unifies storage for streaming and batch data, and can be used in Spark to define relational tables for both batch and stream processing. When used for stream processing, a Delta Lake table can be used as a streaming source for queries against real-time data, or as a sink to which a stream of data is written.

The Spark runtimes in Azure Synapse Analytics and Azure Databricks include support for Delta Lake.

Delta Lake combined with Spark Structured Streaming is a good solution when you need to abstract batch and stream processed data in a data lake behind a relational schema for SQL-based querying and analysis.

Azure Data Explorer is a standalone Azure service for efficiently analyzing data. You can use the service as the output for analyzing large volumes of diverse data from data sources such as websites, applications, IoT devices, and more.

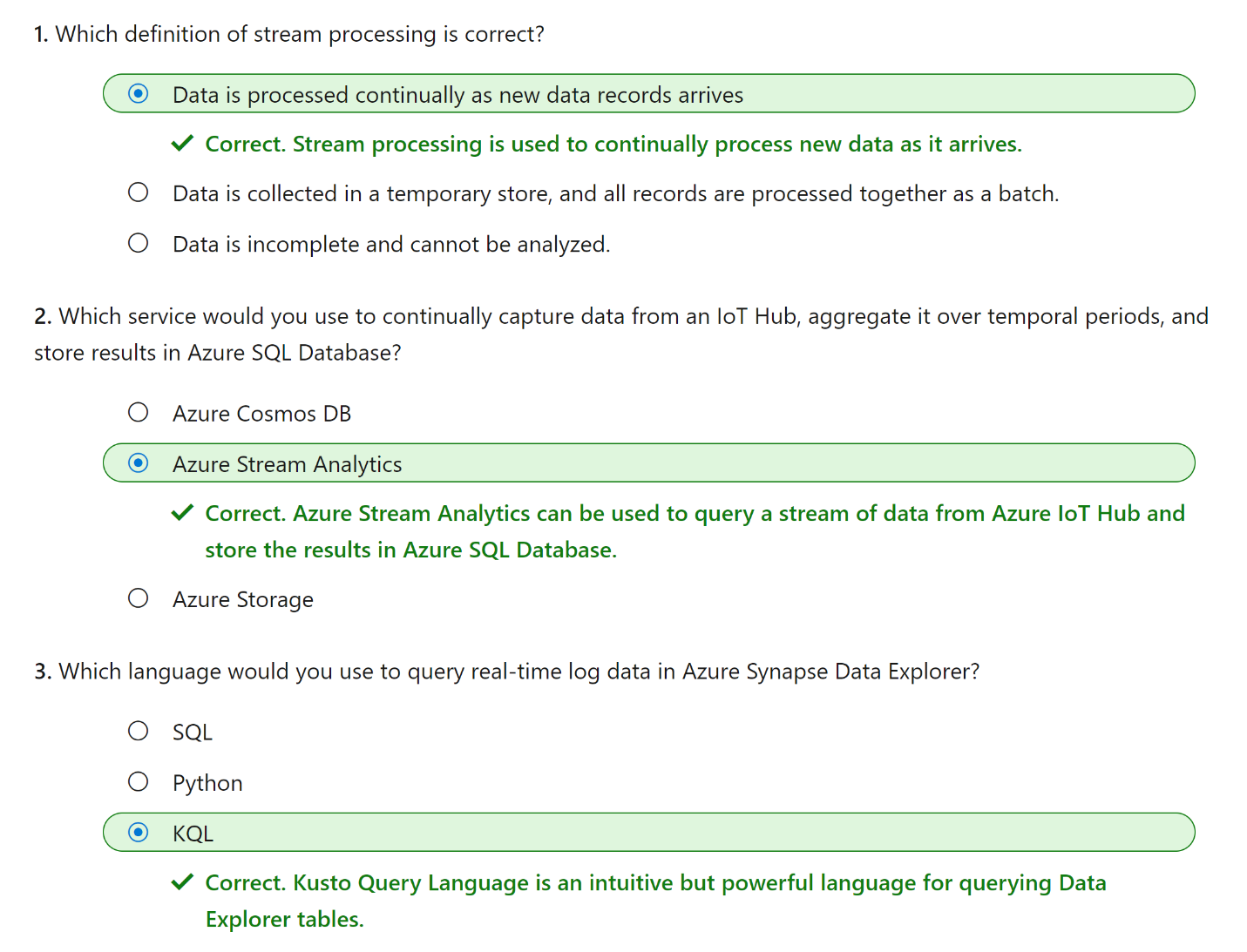


Data is ingested into Data Explorer through one or more connectors or by writing a minimal amount of code. This enables you to quickly ingest data from a wide variety of data sources, including both static and streaming sources. Data Explorer supports batching and streaming in near real time to optimize data ingestion. The ingested data is stored in tables in a Data Explorer database, where automatic indexing enables high-performance queries.

Azure Data Explorer is a great choice of technology when you need to:

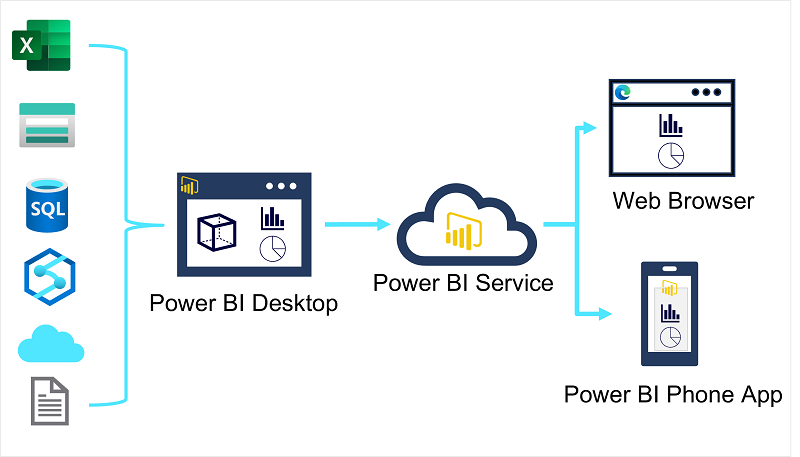
* Capture and analyze real-time or batch data that includes a time-series element; such as log telemetry or values emitted by Internet-of-things (IoT) devices.
* Explore, filter, and aggregate data quickly by using the intuitive and powerful Kusto Query Language (KQL).

Azure Synapse Data Explorer is an especially good choice when you need to perform these tasks in a centralized environment used for other kinds of analytics, such as SQL and Spark based queries.



# Module 8 - Explore fundamentals of data visualization

Microsoft Power BI is a suite of tools and services that data analysts can use to build interactive data visualizations for business users to consume.



A typical workflow for creating a data visualization solution starts with **Power BI Desktop**, a Microsoft Windows application in which you can import data from a wide range of data sources, combine and organize the data from these sources in an analytics data model, and create reports that contain interactive visualizations of the data.

After you've created data models and reports, you can publish them to the **Power BI service**; a cloud service in which reports can be published and interacted with by business users. You can also do some basic data modeling and report editing directly in the service using a web browser, but the functionality for this is limited compared to the Power BI Desktop tool. You can use the service to schedule refreshes of the data sources on which your reports are based, and to share reports with other users. You can also define dashboards and apps that combine related reports in a single, easy to consume location.

Users can consume reports, dashboards, and apps in the Power BI service through a web browser, or on mobile devices by using the **Power BI phone app**.

Analytical models enable you to structure data to support analysis. Models are based on related tables of data and define the numeric values that you want to analyze or report (known as measures) and the entities by which you want to aggregate them (known as dimensions). Conceptually, the model forms a multidimensional structure, which is commonly referred to as a cube, in which any point where the dimensions intersect represents an aggregated measure for those dimensions.)

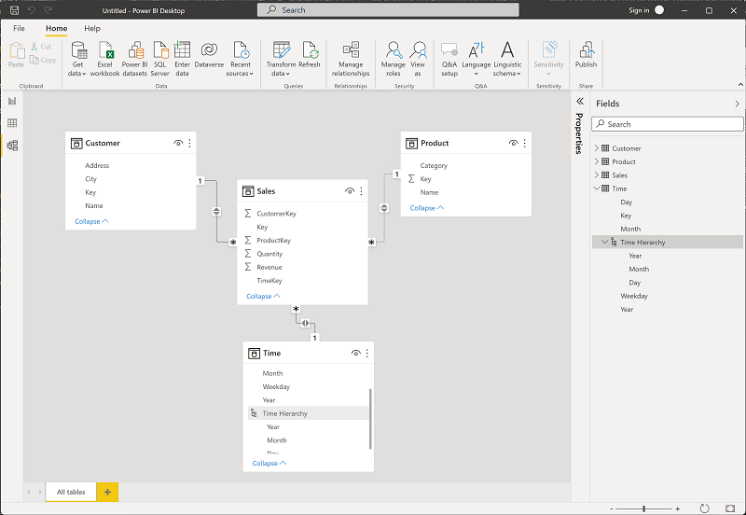
The numeric measures that will be aggregated by the various dimensions in the model are stored in Fact tables. Each row in a fact table represents a recorded event that has numeric measures associated with it.

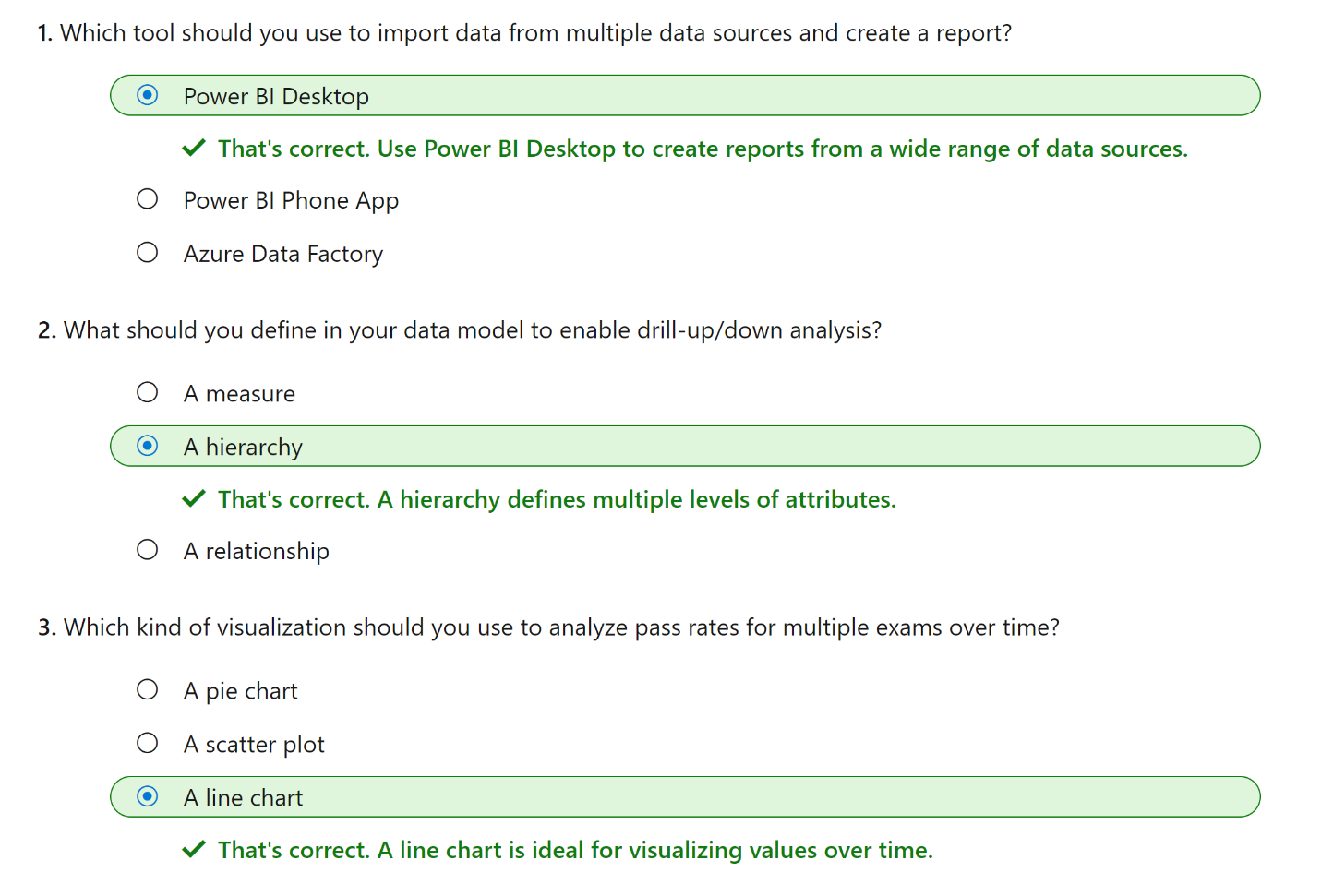
The numeric measures that will be aggregated by the various dimensions in the model are stored in Fact tables. Each row in a fact table represents a recorded event that has numeric measures associated with it.

One final thing worth considering about analytical models is the creation of attribute hierarchies that enable you to quickly drill-up or drill-down to find aggregated values at different levels in a hierarchical dimension. For example, consider the attributes in the dimension tables we’ve discussed so far. In the **Product** table, you can form a hierarchy in which each category might include multiple named products.

You can use Power BI to define an analytical model from tables of data, which can be imported from one or more data source. You can then use the data modeling interface on the **Model** tab of Power BI Desktop to define your analytical model by creating relationships between fact and dimension tables, defining hierarchies, setting data types and display formats for fields in the tables, and managing other properties of your data that help define a rich model for analysis.

1. Download the Power BI Desktop installer from <https://aka.ms/power-bi-desktop>.
2. When the file has downloaded, open it, and use the setup wizard to install Power BI Desktop on your computer. This may take a few minutes

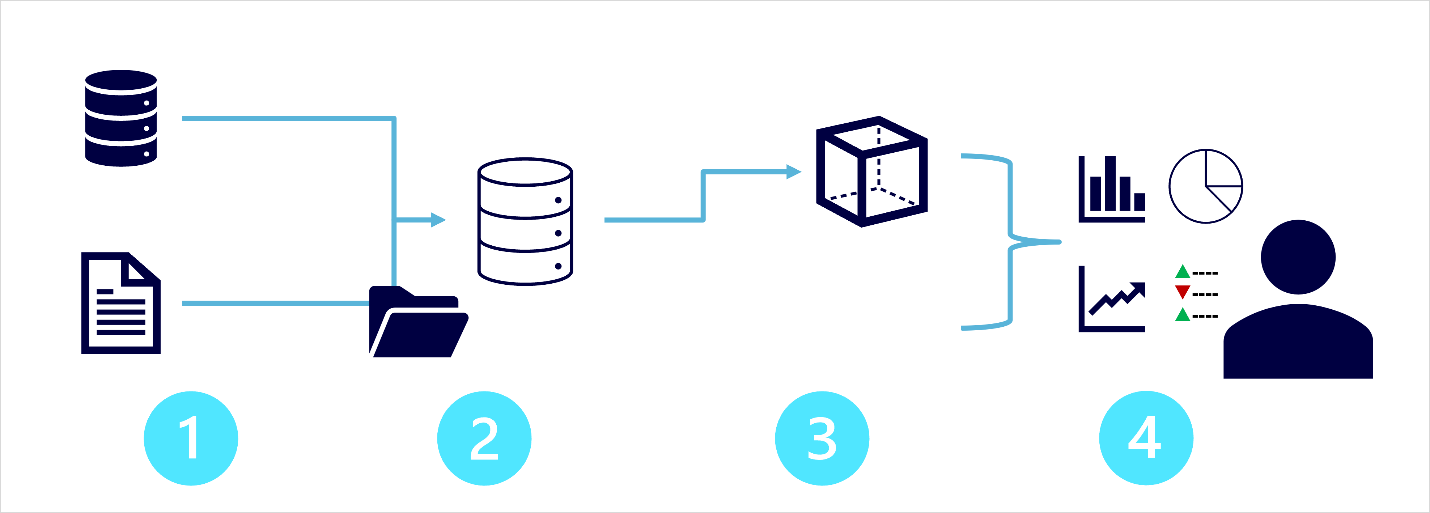




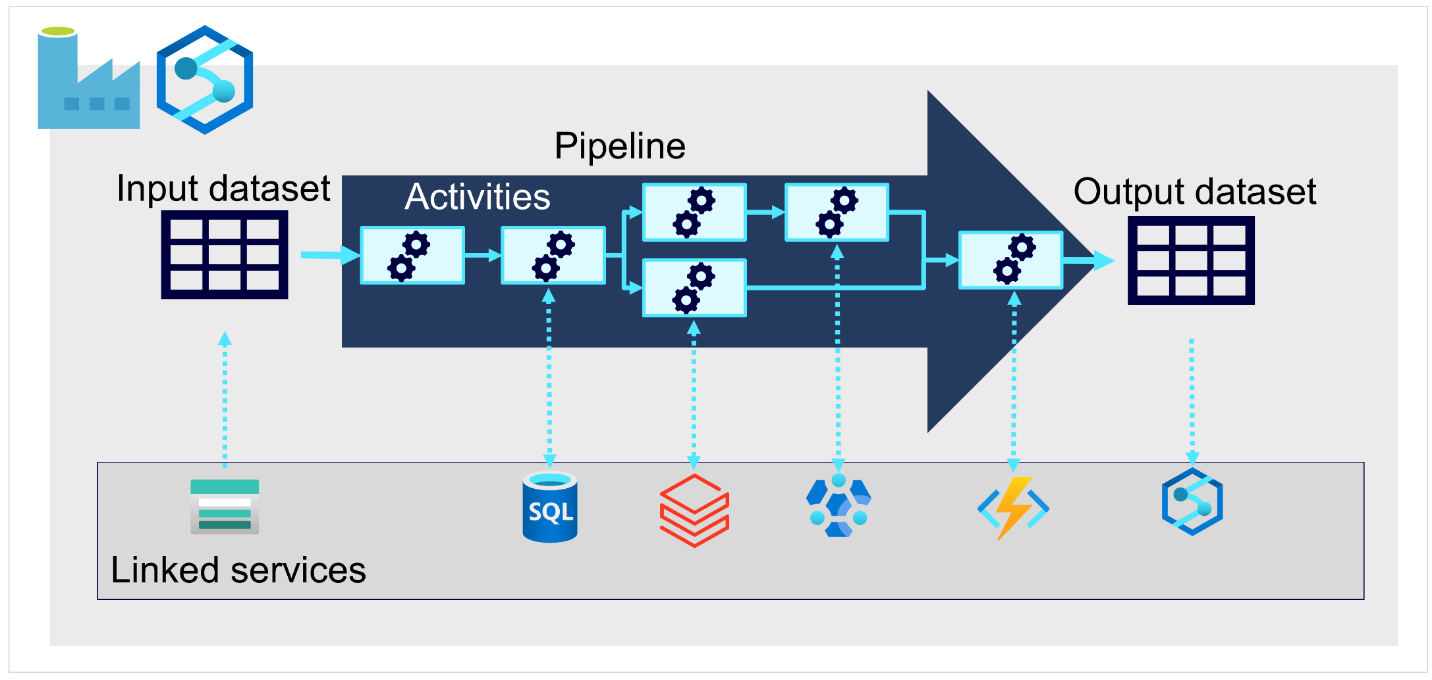
# Module 9 - Explore fundamentals of large-scale data warehousing

Large-scale data warehousing solutions combine conventional data warehousing used to support business intelligence (BI) with techniques used for so-called "big data" analytics. A conventional data warehousing solution typically involves copying data from transactional data stores into a relational database with a schema that's optimized for querying and building multidimensional models. Big Data processing solutions on the other hand, are used with large volumes of data in multiple formats, which is batch loaded or captured in real-time streams and stored in a data lake from which distributed processing engines like Apache Spark are used to process it.

Large-scale data warehousing architecture can vary, as can the specific technologies used to implement it; but in general, the following elements are included:



1. **Data ingestion and processing** – data from one or more transactional data stores, files, real-time streams, or other sources is loaded into a data lake or a relational data warehouse. The load operation usually involves an *extract, transform, and load* (ETL) or *extract, load, and transform* (ELT) process in which the data is cleaned, filtered, and restructured for analysis. In ETL processes, the data is transformed before being loaded into an analytical store, while in an ELT process the data is copied to the store and then transformed. Either way, the resulting data structure is optimized for analytical queries. The data processing is often performed by distributed systems that can process high volumes of data in parallel using multi-node clusters. Data ingestion includes both batch processing of static data and real-time processing of streaming data.
2. **Analytical data store** – data stores for large scale analytics include relational *data warehouses*, file-system based *data lakes*, and hybrid architectures that combine features of data warehouses and data lakes (sometimes called *data lakehouses* or *lake databases*). We'll discuss these in more depth later.
3. **Analytical data model** – while data analysts and data scientists can work with the data directly in the analytical data store, it’s common to create one or more data models that pre-aggregate the data to make it easier to produce reports, dashboards, and interactive visualizations. Often these data models are described as *cubes*, in which numeric data values are aggregated across one or more dimensions (for example, to determine total sales by product and region). The model encapsulates the relationships between data values and dimensional entities to support "drill-up/drill-down" analysis.
4. **Data visualization** – data analysts consume data from analytical models, and directly from analytical stores to create reports, dashboards, and other visualizations. Additionally, users in an organization who may not be technology professionals might perform self-service data analysis and reporting. The visualizations from the data show trends, comparisons, and key performance indicators (KPIs) for a business or other organization, and can take the form of printed reports, graphs and charts in documents or PowerPoint presentations, web-based dashboards, and interactive environments in which users can explore data visually.



On Azure, large-scale data ingestion is best implemented by creating pipelines that orchestrate ETL processes. You can create and run pipelines using [Azure Data Factory](https://azure.microsoft.com/services/data-factory), or you can use the same pipeline engine in [Azure Synapse Analytics](https://azure.microsoft.com/services/synapse-analytics) if you want to manage all of the components of your data warehousing solution in a unified workspace.

In either case, pipelines consist of one or more activities that operate on data. An input dataset provides the source data, and activities can be defined as a data flow that incrementally manipulates the data until an output dataset is produced. Pipelines use linked services to load and process data – enabling you to use the right technology for each step of the workflow.

A data warehouse is a relational database in which the data is stored in a schema that is optimized for data analytics rather than transactional workloads. Commonly, the data from a transactional store is transformed into a schema in which numeric values are stored in central fact tables, which are related to one or more dimension tables that represent entities by which the data can be aggregated. This kind of fact and dimension table schema is called a star schema; though it's often extended into a snowflake schema by adding additional tables related to the dimension tables to represent dimensional hierarchies (for example, product might be related to product categories). A data warehouse is a great choice when you have transactional data that can be organized into a structured schema of tables, and you want to use SQL to query them.

A data lake is a file store, usually on a distributed file system for high performance data access. Technologies like Spark or Hadoop are often used to process queries on the stored files and return data for reporting and analytics. These systems often apply a schema-on-read approach to define tabular schemas on semi-structured data files at the point where the data is read for analysis, without applying constraints when it's stored. Data lakes are great for supporting a mix of structured, semi-structured, and even unstructured data that you want to analyze without the need for schema enforcement when the data is written to the store.

You can use a hybrid approach that combines features of data lakes and data warehouses in a lake database or data lakehouse. The raw data is stored as files in a data lake, and a relational storage layer abstracts the underlying files and expose them as tables, which can be queried using SQL. SQL pools in Azure Synapse Analytics include PolyBase, which enables you to define external tables based on files in a datalake (and other sources) and query them using SQL. Synapse Analytics also supports a Lake Database approach in which you can use database templates to define the relational schema of your data warehouse, while storing the underlying data in data lake storage – separating the storage and compute for your data warehousing solution. Data lakehouses are a relatively new approach in Spark-based systems, and are enabled through technologies like Delta Lake; which adds relational storage capabilities to Spark, so you can define tables that enforce schemas and transactional consistency, support batch-loaded and streaming data sources, and provide a SQL API for querying.

there are three main services that you can use to implement a large-scale analytical store

[**Azure Synapse Analytics**](https://azure.microsoft.com/services/synapse-analytics) is a unified, end-to-end solution for large scale data analytics. It brings together multiple technologies and capabilities, enabling you to combine the data integrity and reliability of a scalable, high-performance SQL Server based relational data warehouse with the flexibility of a data lake and open-source Apache Spark. It also includes native support for log and telemetry analytics with Azure Synapse Data Explorer pools, as well as built in data pipelines for data ingestion and transformation. All Azure Synapse Analytics services can be managed through a single, interactive user interface called Azure Synapse Studio, which includes the ability to create interactive notebooks in which Spark code and markdown content can be combined. Synapse Analytics is a great choice when you want to create a single, unified analytics solution on Azure.

[**Azure Databricks**](https://azure.microsoft.com/services/databricks) is an Azure implementation of the popular Databricks platform. Databricks is a comprehensive data analytics solution built on Apache Spark, and offers native SQL capabilities as well as workload-optimized Spark clusters for data analytics and data science. Databricks provides an interactive user interface through which the system can be managed and data can be explored in interactive notebooks. Due to its common use on multiple cloud platforms, you might want to consider using Azure Databricks as your analytical store if you want to use existing expertise with the platform or if you need to operate in a multi-cloud environment or support a cloud-portable solution.

[**Azure HDInsight**](https://azure.microsoft.com/services/hdinsight) is an Azure service that supports multiple open-source data analytics cluster types. Although not as user-friendly as Azure Synapse Analytics and Azure Databricks, it can be a suitable option if your analytics solution relies on multiple open-source frameworks or if you need to migrate an existing on-premises Hadoop-based solution to the cloud.

