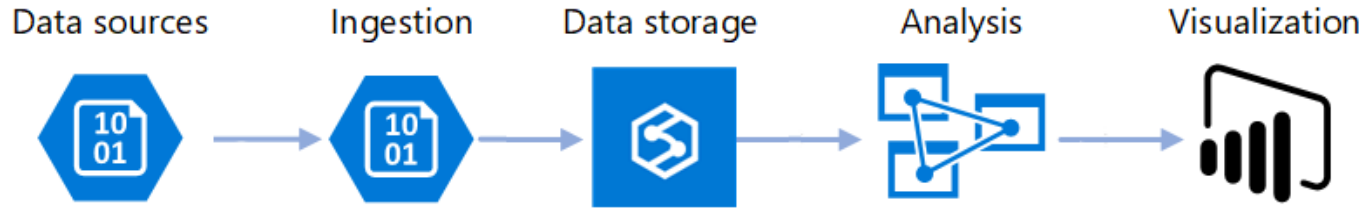
DP900 Final Notes – Part 4

Describe an analytics workload on Azure (25—30%)



Describe common elements of large-scale analytics

• Describe considerations for data ingestion and processing

If your organization is small, and works with limited data sources, then you might not require a data integration service at all. If, however, your organization works with big data, or is a traditional relational data warehousing organization, you might benefit from a data integration solution. Consider the following points:

Big data organizations rely on technologies for handling large amounts of diverse data. For them, Azure Data Factory provides a means to create and run pipelines in the cloud. These pipelines can access both cloud and on-premises data services. These pipelines typically work with technologies such as Azure Synapse Analytics, Azure Blobs, Azure Data Lake, Azure HDInsight, Azure Databricks, and Azure Machine Learning.

Relational data warehousing organizations typically rely on technologies such as SQL Server. SSIS is often used to create SSIS packages. For such organizations, Azure Data Factory provides the ability to run SSIS packages on Azure, letting them access both cloud and on-premises data services.

• Describe options for analytical data stores

These options provide various database models that are optimized for different types of tasks:

1. Key/value databases hold a single serialized object for each key value. They're good for storing large volumes of data where you want to get one item for a given key value and you don't have to query based on other properties of the item.
2. Document databases are key/value databases in which the values are *documents*. A "document" in this context is a collection of named fields and values. The database typically stores the data in a format such as XML, YAML, JSON, or BSON, but may use plain text. Document databases can query on non-key fields and define secondary indexes to make querying more efficient. This makes a document database more suitable for applications that need to retrieve data based on criteria more complex than the value of the document key. For example, you could query on fields such as product ID, customer ID, or customer name.
3. Column store databases are key/value data stores that store each column separately on disk. A *wide column store* database is a type of column store database that stores *column families*, not just single columns. For example, a census database might have a column family for a person's name (first, middle, last), a family for the person's address, and a family for the person's profile information (date of birth, gender). The database can store each column family in a separate partition, while keeping all the data for one person related to the same key. An application can read a single column family without reading through all of the data for an entity.
4. Graph databases store information as a collection of objects and relationships. A graph database can efficiently perform queries that traverse the network of objects and the relationships between them. For example, the objects might be employees in a human resources database, and you might want to facilitate queries such as "find all employees who directly or indirectly work for Scott."
5. Telemetry and time series databases are an append-only collection of objects. Telemetry databases efficiently index data in a variety of column stores and in-memory structures, making them the optimal choice for storing and analyzing vast quantities of telemetry and time series data.

General capabilities

Capability	SQL Database	Azure Synapse SQL pool	Azure Synapse Spark pool	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Is managed service	Yes	Yes	Yes	Yes	Yes [1]	Yes [1]	Yes	Yes
Primary database model	Relational (column store format when using columnstore indexes)	Relational tables with column storage	Wide column store	Relational (column store), telemetry, and time series store	Wide column store	Hive/In-Memory	Tabular semantic models	Document store, graph, key-value store, wide column store
SQL language support	Yes	Yes	Yes	Yes	Yes (using Phoenix JDBC driver)	Yes	No	Yes
Optimized for speed serving layer	Yes [2]	Yes [3]	Yes	Yes	Yes	Yes	No	Yes

[1] With manual configuration and scaling.
[2] Using memory-optimized tables and hash or nonclustered indexes.
[3] Supported as an Azure Stream Analytics output.

Scalability capabilities

Capability	SQL Database	Azure Synapse SQL pool	Azure Synapse Spark pool	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Redundant regional servers for high availability	Yes	No	No	Yes	Yes	No	Yes	Yes
Supports query scale-out	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic scalability (scale up)	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Supports in-memory caching of data	Yes	Yes	Yes	Yes	No	Yes	Yes	No

Security capabilities

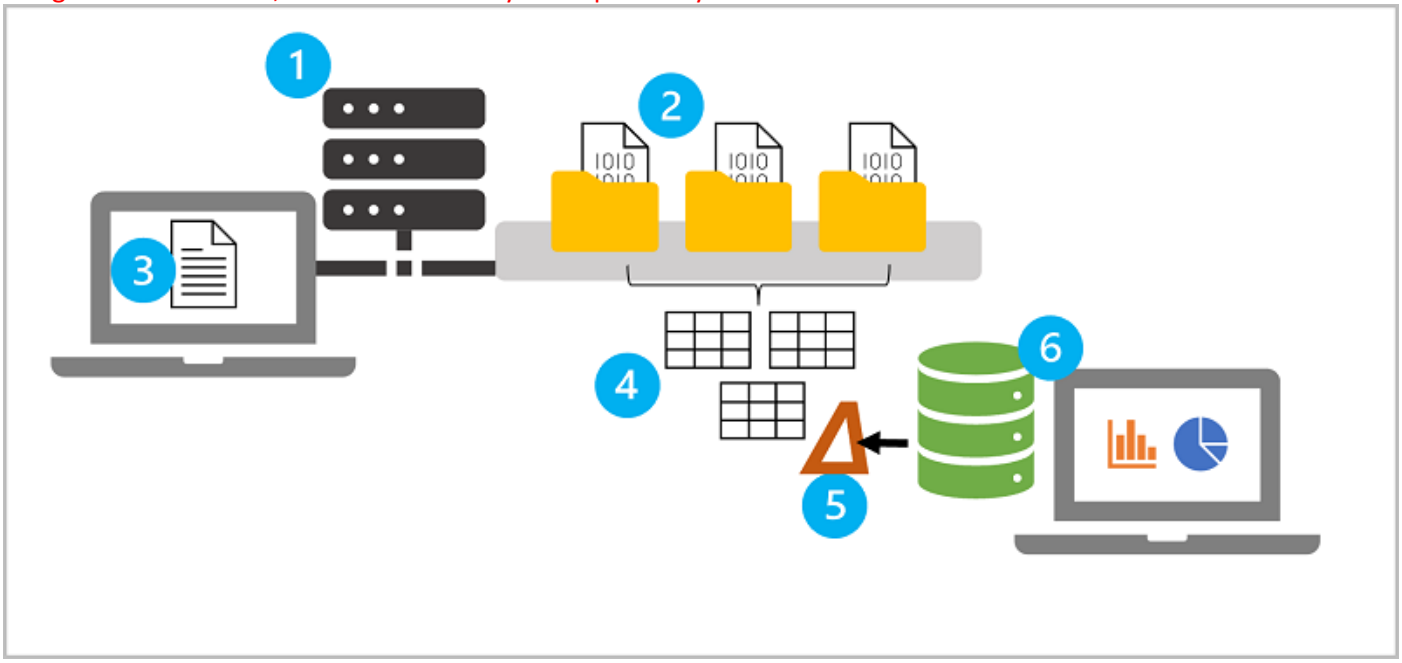
Capability	SQL Database	Azure Synapse	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Authentication	SQL / Azure Active Directory (Azure AD)	SQL / Azure AD	Azure AD	local / Azure AD [1]	local / Azure AD [1]	Azure AD	database users / Azure AD via access control (IAM)
Data encryption at rest	Yes [2]	Yes [2]	Yes	Yes [1]	Yes [1]	Yes	Yes
Row-level security	Yes	Yes [3]	Yes	Yes [1]	Yes [1]	Yes	No
Supports firewalls	Yes	Yes	Yes	Yes [4]	Yes [4]	Yes	Yes
Dynamic data masking	Yes	Yes	Yes	Yes [1]	Yes	No	No

• Describe Azure services for data warehousing, including Azure Synapse Analytics, Azure Databricks, Azure HDInsight, and Azure Data Factory

Azure Synapse Analytics
(formally known as SQL Data Warehousing)

Azure Databricks
Azure Databricks is a cloud service that provides a scalable platform for data analytics using Apache Spark.
Azure Databricks is hosted on the Microsoft Azure cloud platform, and integrated with Azure services such as Azure Active Directory, Azure Storage, Azure Synapse Analytics, and Azure Machine Learning. Organizations can apply their existing capabilities with the Databricks platform, and build fully integrated data analytics solutions that work with cloud infrastructure used by other enterprise applications.
Azure Databricks is a comprehensive platform that offers many data processing capabilities. While you can use the service to support any workload that requires scalable data processing, Azure Databricks is optimized for three specific types of data workload and associated user personas: Data Science and Engineering, Machine Learning and SQL (Premium only).

Azure Databricks is an amalgamation of multiple technologies that enable you to work with data at scale. Before using Azure Databricks, there are some key concepts that you should understand.



1. **Apache Spark clusters** - Spark is a distributed data processing solution that makes use of *clusters* to scale processing across multiple compute *nodes*. Each Spark cluster has a *driver* node to coordinate processing jobs, and one or more *worker* nodes on which the processing occurs. This distributed model enables each node to operate on a subset of the job in parallel; reducing the overall time for the job to complete. To learn more about clusters in Azure Databricks, see Clusters in the Azure Databricks documentation.
2. **Databricks File System (DBFS)** - While each cluster node has its own local file system (on which operating system and other node-specific files are stored), the nodes in a cluster have access to a shared, distributed file system in which they can access and operate on data files. The *Databricks File System* (DBFS) enables you to mount cloud storage and use it to work with and persist file-based data. To learn more about DBFS, see Databricks File System (DBFS) in the Azure Databricks documentation.
3. **Notebooks** - One of the most common ways for data analysts, data scientists, data engineers, and developers to work with Spark is to write code in *notebooks*. Notebooks provide an interactive environment in which you can combine text and graphics in *Markdown* format with cells containing code that you run interactively in the notebook session. To learn more about notebooks, see Notebooks in the Azure Databricks documentation.
4. **Hive metastore** - *Hive* is an open source technology used to define a relational abstraction layer of tables over file-based data. The tables can then be queried using SQL syntax. The table definitions and details of the file system locations on which they're based is stored in the metastore for a Spark cluster. A *Hive*

metastore is created for each cluster when it's created, but you can configure a cluster to use an existing external metastore if necessary - see Metastores in the Azure Databricks documentation for more details.

5. **Delta Lake** - *Delta Lake* builds on the relational table schema abstraction over files in the data lake to add support for SQL semantics commonly found in relational database systems. Capabilities provided by Delta Lake include transaction logging, data type constraints, and the ability to incorporate streaming data into a relational table. To learn more about Delta Lake, see Delta Lake Guide in the Azure Databricks documentation.
6. **SQL Warehouses** - *SQL Warehouses* are relational compute resources with endpoints that enable client applications to connect to an Azure Databricks workspace and use SQL to work with data in tables. The results of SQL queries can be used to create data visualizations and dashboards to support business analytics and decision making. SQL Warehouses are only available in *premium* tier Azure Databricks workspaces. To learn more about SQL Warehouses, see SQL Warehouses in the Azure Databricks documentation.

Azure HDInsight

Azure HDInsight is a managed, full-spectrum, open-source analytics service in the cloud for enterprises. With HDInsight, you can use open-source frameworks such as, Apache Spark, Apache Hive, LLAP, Apache Kafka, Hadoop and more, in your Azure environment.

What is HDInsight and the Hadoop technology stack?

Azure HDInsight is a full-spectrum, managed cluster platform which simplifies running big data frameworks in large volume and velocity using Apache Spark, Apache Hive, LLAP, Apache Kafka, Apache Hadoop, and more in your Azure environment.

Why should I use Azure HDInsight?

Capability	**Description**
Cloud native	Azure HDInsight enables you to create optimized clusters for Spark, Interactive query (LLAP), Kafka, HBase and Hadoop on Azure. HDInsight also provides an end-to-end SLA on all your production workloads.
Low-cost and scalable	HDInsight enables you to scale workloads up or down. You can reduce costs by creating clusters on demand and paying only for what you use. You can also build data pipelines to operationalize your jobs. Decoupled compute and storage provide better performance and flexibility.
Secure and compliant	HDInsight enables you to protect your enterprise data assets with Azure Virtual Network, encryption, and integration with Azure Active Directory. HDInsight also meets the most popular industry and government compliance standards.
Monitoring	Azure HDInsight integrates with Azure Monitor logs to provide a single interface with which you can monitor all your clusters.
Global availability	HDInsight is available in more regions than any other big data analytics offering. Azure HDInsight is also available in Azure Government, China, and Germany, which allows you to meet your enterprise needs in key sovereign areas.
Productivity	Azure HDInsight enables you to use rich productive tools for Hadoop and Spark with your preferred development environments. These development environments include Visual Studio, VSCode, Eclipse, and IntelliJ for Scala, Python, Java, and .NET support.
Extensibility	You can extend the HDInsight clusters with installed components (Hue, Presto, and so on) by using script actions, by adding edge nodes, or by integrating with other big data certified applications. HDInsight enables seamless integration with the most popular big data solutions with a one-click deployment.

What is big data?

Big data is collected in escalating volumes, at higher velocities, and in a greater variety of formats than ever before. It can be historical (meaning stored) or real time (meaning streamed from the source). See Scenarios for using HDInsight to learn about the most common use cases for big data.

Cluster types in HDInsight

HDInsight includes specific cluster types and cluster customization capabilities, such as the capability to add components, utilities, and languages. HDInsight offers the following cluster types:

Cluster Type	Description	Get Started
Apache Hadoop	A framework that uses HDFS, YARN resource management, and a simple MapReduce programming model to process and analyze batch data in parallel.	Create an Apache Hadoop cluster
Apache Spark	An open-source, parallel-processing framework that supports in-memory processing to boost the performance of big-data analysis applications. See What is Apache Spark in HDInsight?.	Create an Apache Spark cluster
Apache HBase	A NoSQL database built on Hadoop that provides random access and strong consistency for large amounts of unstructured and semi-structured data--potentially billions of rows times millions of columns. See What is HBase on HDInsight?	Create an Apache HBase cluster
Apache Interactive Query	In-memory caching for interactive and faster Hive queries. See Use Interactive Query in HDInsight.	Create an Interactive Query cluster
Apache Kafka	An open-source platform that's used for building streaming data pipelines and applications. Kafka also provides message-queue functionality that allows you to publish and subscribe to data streams. See Introduction to Apache Kafka on HDInsight.	Create an Apache Kafka cluster

Scenarios for using HDInsight

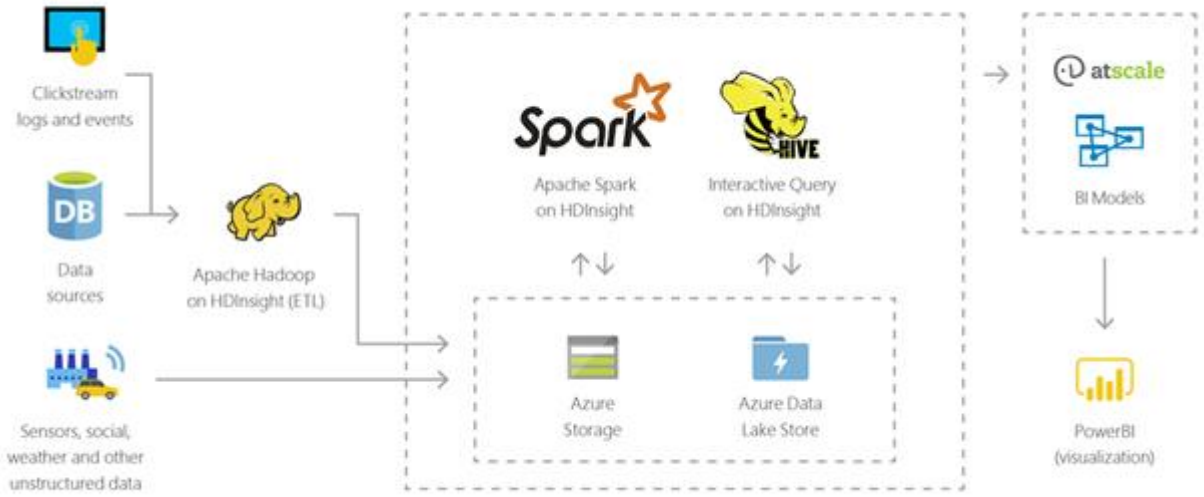
Azure HDInsight can be used for various scenarios in big data processing. It can be historical data (data that's already collected and stored) or real-time data (data that's directly streamed from the source). The scenarios for processing such data can be summarized in the following categories:

Batch processing (ETL)

Extract, transform, and load (ETL) is a process where unstructured or structured data is extracted from heterogeneous data sources. It's then transformed into a structured format and loaded into a data store. You can use the transformed data for data science or data warehousing.

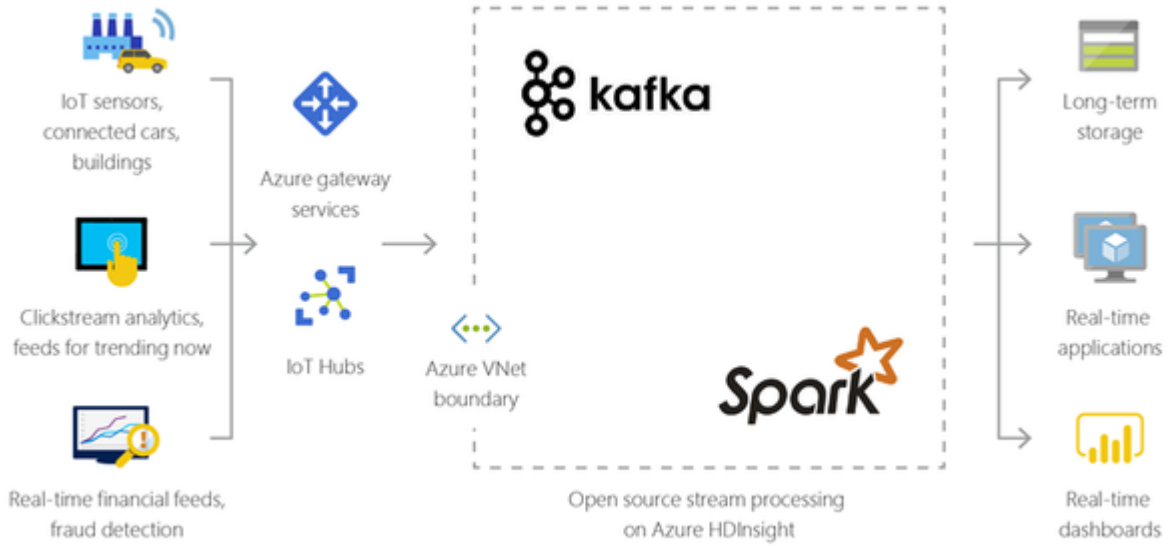
Data warehousing

You can use HDInsight to perform interactive queries at petabyte scales over structured or unstructured data in any format. You can also build models connecting them to BI tools.



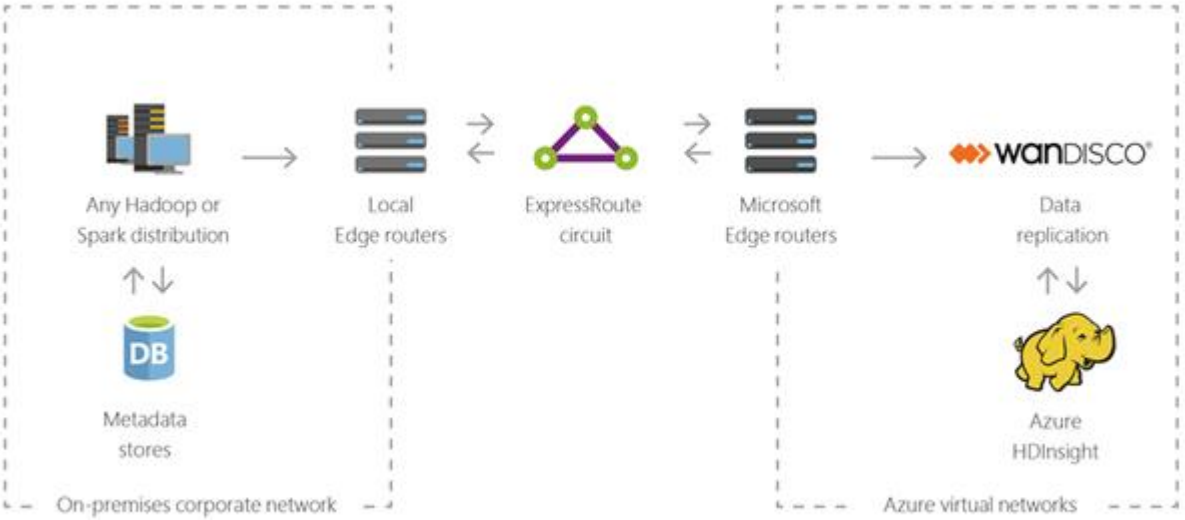
Internet of Things (IoT)

You can use HDInsight to process streaming data that's received in real time from different kinds of devices. For more information, read this blog post from Azure that announces the public preview of Apache Kafka on HDInsight with Azure Managed disks.



Hybrid

You can use HDInsight to extend your existing on-premises big data infrastructure to Azure to apply the advanced analytics capabilities of the cloud.



Open-source components in HDInsight

Azure HDInsight enables you to create clusters with open-source frameworks such as Spark, Hive, LLAP, Kafka, Hadoop and HBase. These clusters, by default, come with other open-source components that are included on the cluster such as Apache Ambari, Avro, Apache Hive3, HCatalog, Apache Hadoop MapReduce, Apache Hadoop YARN, Apache Phoenix, Apache Pig, Apache Sqoop, Apache Tez, Apache Oozie, and Apache ZooKeeper.

Programming languages in HDInsight

HDInsight clusters, including Spark, HBase, Kafka, Hadoop, and others, support many programming languages. Some programming languages aren't installed by default. For libraries, modules, or packages that aren't installed by default, use a script action to install the component.

Programming language	**Information**
Default programming language support	By default, HDInsight clusters support: Java Python .NET Go

Programming language	Information
Java virtual machine (JVM) languages	Many languages other than Java can run on a Java virtual machine (JVM). However, if you run some of these languages, you might have to install more components on the cluster. The following JVM-based languages are supported on HDInsight clusters: Clojure Jython (Python for Java) Scala
Hadoop-specific languages	HDInsight clusters support the following languages that are specific to the Hadoop technology stack: Pig Latin for Pig jobs HiveQL for Hive jobs and SparkSQL

Development tools for HDInsight

You can use HDInsight development tools, including IntelliJ, Eclipse, Visual Studio Code, and Visual Studio, to author and submit HDInsight data query and job with seamless integration with Azure.

Azure toolkit for IntelliJ10

Azure toolkit for Eclipse6

Azure HDInsight tools for VS Code13

Azure data lake tools for Visual Studio9

Business intelligence on HDInsight

Familiar business intelligence (BI) tools retrieve, analyze, and report data that is integrated with HDInsight by using either the Power Query add-in or the Microsoft Hive ODBC Driver:

Apache Spark BI using data visualization tools with Azure HDInsight

Visualize Apache Hive data with Microsoft Power BI in Azure HDInsight

Visualize Interactive Query Hive data with Power BI in Azure HDInsight

Connect Excel to Apache Hadoop with Power Query (requires Windows)

Connect Excel to Apache Hadoop with the Microsoft Hive ODBC Driver (requires Windows)

In-region data residency

Spark, Hadoop, and LLAP don't store customer data, so these services automatically satisfy in-region data residency requirements specified in the Trust Center.

Kafka and HBase do store customer data. This data is automatically stored by Kafka and HBase in a single region, so this service satisfies in-region data residency requirements specified in the Trust Center.

Familiar business intelligence (BI) tools retrieve, analyze, and report data that is integrated with HDInsight by using either the Power Query add-in or the Microsoft Hive ODBC Driver.

Azure Data Factory

Azure Data Factory is comprised of the components described in the following table.

Pipelines: A logical grouping of activities that perform a specific unit of work. These activities together perform a task. The advantage of using a pipeline is that you can more easily manage the activities as a set instead of as individual items.

Activities: A single processing step in a pipeline. Azure Data Factory supports three types of activity: data movement, data transformation, and control activities.

Datasets: Represent data structures within your data stores. These point to (or reference) the data that you want to use in your activities as either inputs or outputs.

Linked services: Define the required connection information needed for Azure Data Factory to connect to external resources, such as a data source. Azure Data Factory uses these for two purposes: to represent a **data store** or a **compute resource**.

Data flows: Enable your data engineers to develop data transformation logic without needing to write code. Data flows are run as activities within Azure Data Factory pipelines that use scaled-out Apache Spark clusters.

Integration runtimes: Azure Data Factory uses the compute infrastructure to provide the following data integration capabilities across different network environments: data flow, data movement, activity dispatch, and SSIS package execution. In Azure Data Factory, an integration runtime provides the bridge between the activity and linked services.

Describe consideration for real-time data analytics

• Describe the difference between batch and streaming data

STREAM PROCESSING

Input volume = small batches
Output volume = small batches
Input Type = very dynamic
Concurrency = very high
Transformations = window aggregation
Latency = very low
Type of job = fast running
Uptime = always running
Memory consumption = low
Message ingestion = Azure event hub, Apache Kafta, HDInsight with Kafta
Processing engines = Azure stream analytics, Azure Databricks with Spark streming, HDInsight with Spart streaming or Storm

BATCH PROCESSING

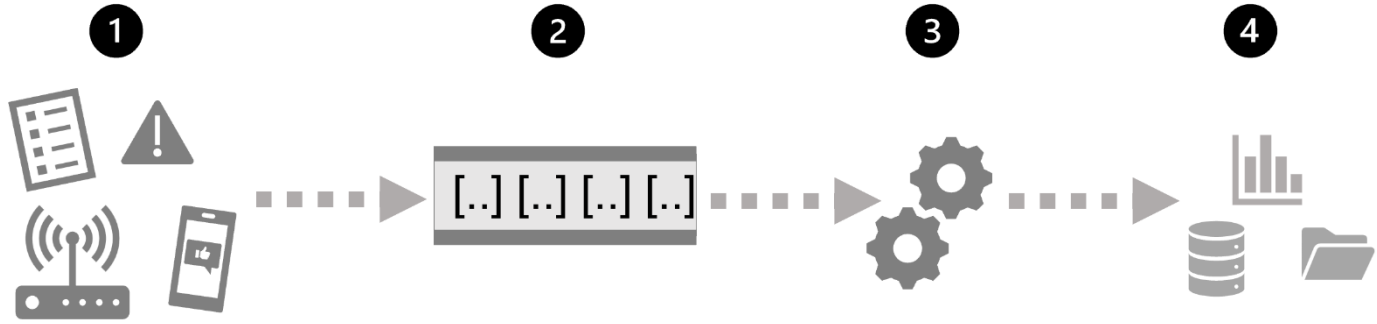
Input volume = large batches
Output volume = small batches, large batches, structured data
Input Type = almost static
Concurrency = very low
Transformations = complex transformations
Latency = high
Type of job = long running
Uptime = scheduled runs
Memory consumption = very high
Message ingestion = N/A
Processing engines = Azure Data Factory with Mapping Data Flows, Azure Data Factory with Wrangling Data Flows, Azure Synapse Analytics, Azure Data Lake Analytics, Azure Databricks, HDInsight with Hive, Spark or MapReduce

• Describe technologies for real-time analytics including Azure Stream Analytics, Azure Synapse Data Explorer, and Spark structured streaming

Azure Stream Analytics

A general architecture for stream processing

At its simplest, a high-level architecture for stream processing looks like this:



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.

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.

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.

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.

Real-time analytics in Azure

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.

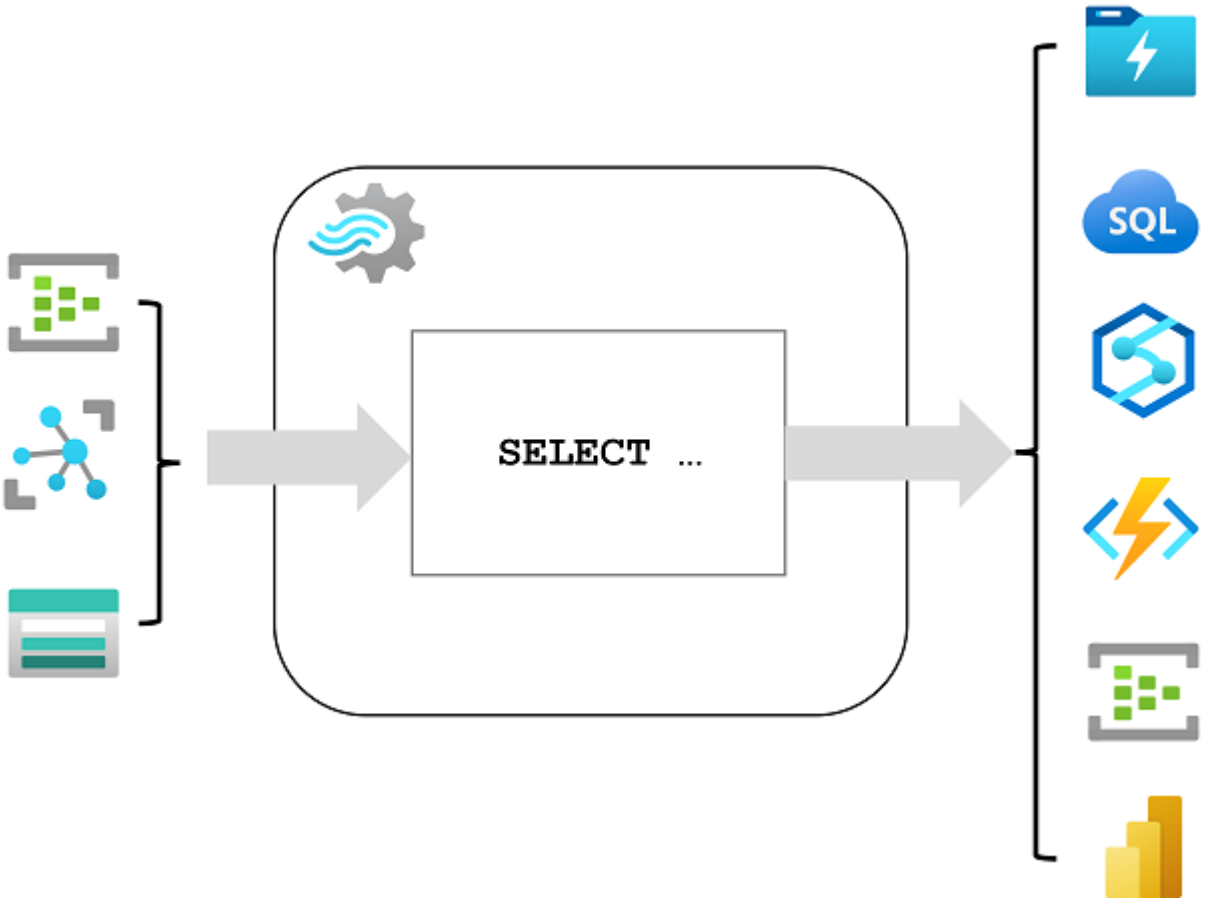
Explore Azure Stream Analytics

Azure Stream Analytics is a service for complex event processing and analysis of streaming data. Stream Analytics is used to:

Ingest data from an *input*, such as an Azure event hub, Azure IoT Hub, or Azure Storage blob container.

Process the data by using a *query* to select, project, and aggregate data values.

Write the results to an *output*, such as Azure Data Lake Gen 2, Azure SQL Database, Azure Synapse Analytics, Azure Functions, Azure event hub, Microsoft Power BI, or others.



Once started, a Stream Analytics query will run perpetually, processing new data as it arrives in the input and storing results in the output.

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.

Azure Stream Analytics jobs and clusters

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.

Explore Apache Spark on Microsoft Azure

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.

Spark Structured Streaming

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.

Note

For more information about Spark Structured Streaming, see the **Spark Structured Streaming programming guide.**

Delta Lake

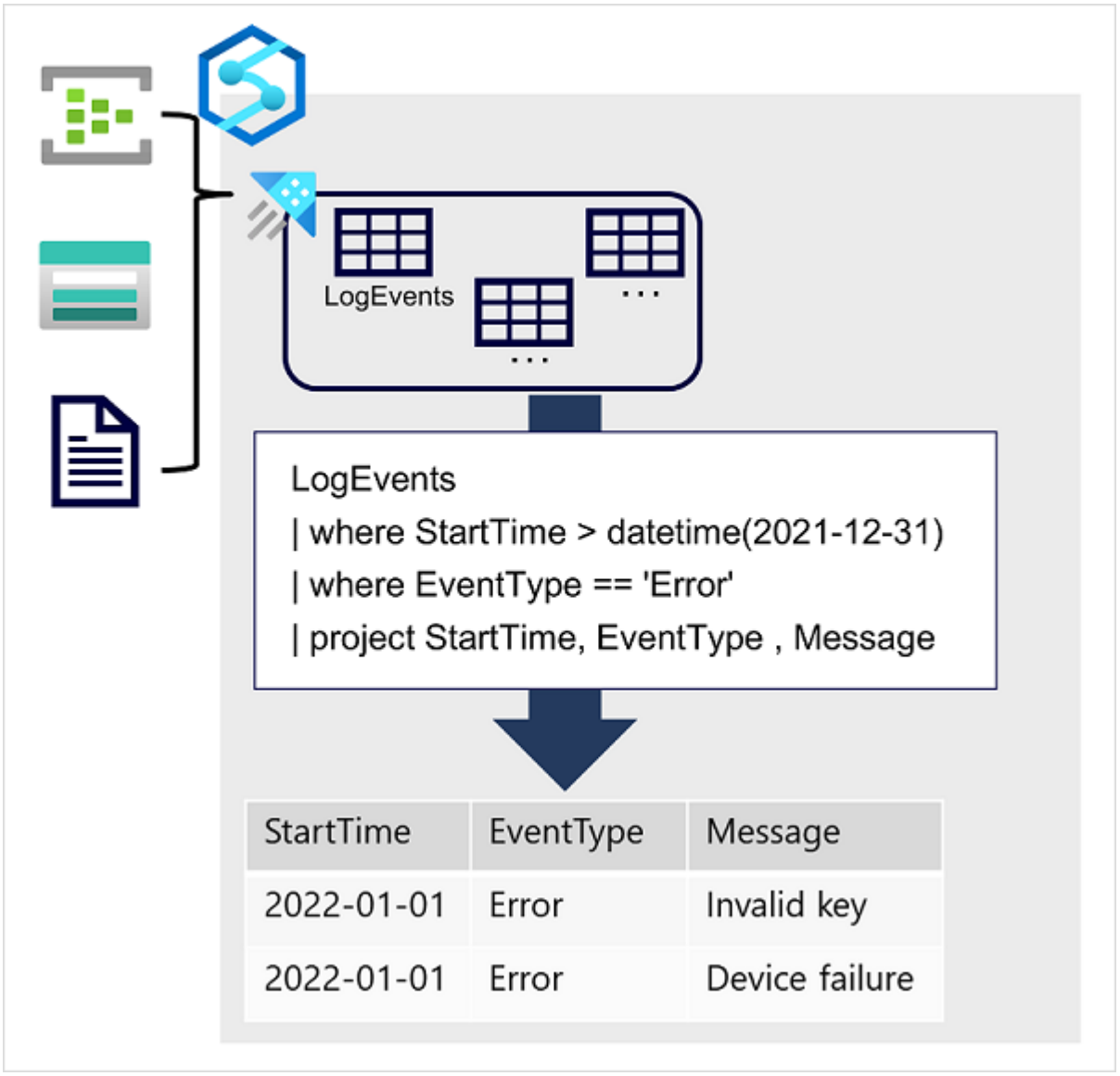
Delta Lake is an open-source storage layer that adds support for transactional consistency, schema enforcement, and other common data warehousing features to data lake storage. It also unifies storage for streaming and batch data, and can be used in Spark to define relational tables for both batch and 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.

Explore Azure Data Explorer

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. For example, by outputting Azure Stream Analytics logs to Azure Data Explorer, you can complement Stream Analytics low latency alerts handling with Data Explorer's deep investigation capabilities. The service is also encapsulated as a runtime in Azure Synapse Analytics, where it is referred to as Azure Synapse Data Explorer; enabling you to build and manage analytical solutions that combine SQL, Spark, and Data Explorer analytics in a single workspace.



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.

Kusto Query Language (KQL)

To query Data Explorer tables, you can use Kusto Query Language (KQL), a language that is specifically optimized for fast read performance – particularly with telemetry data that includes a timestamp attribute.
The most basic KQL query consists simply of a table name, in which case the query returns all of the data in the table. For example, the following query would return the contents of the **LogEvents** table:

KustoCopy

```
LogEvents
```

You can add clauses to a Kusto query to filter, sort, aggregate, and return (*project*) specific columns. Each clause is prefixed by a **|** character. For example, the following query returns the **StartTime**, **EventType**, and **Message** columns from the **LogEvents** table for errors that were recorded after December 31st 2021.

KustoCopy

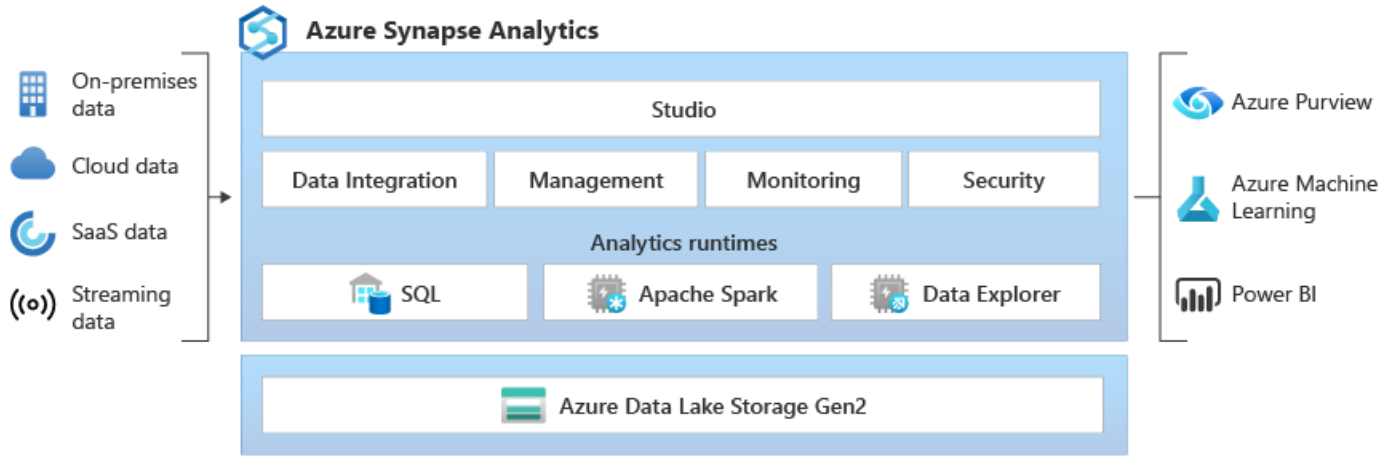
```
LogEvents
| where StartTime > datetime(2021-12-31)
```

```
| where EventType == 'Error'
| project StartTime, EventType , Message
```

Kusto query language is a versatile but intuitive language that enables data analysts to quickly gain insights from data captured and stored in a Data Explorer database.

Azure Synapse Data Explorer

Azure Synapse Data Explorer provides customers with an interactive query experience to unlock insights from log and telemetry data. To complement existing SQL and Apache Spark analytics runtime engines, the Data Explorer analytics runtime is optimized for efficient log analytics using powerful indexing technology to automatically index free-text and semi-structured data commonly found in telemetry data.



To learn more, see the following video:

What makes Azure Synapse Data Explorer unique?

Easy ingestion - Data Explorer offers built-in integrations for no-code/low-code, high-throughput data ingestion, and caching data from real-time sources. Data can be ingested from sources such as Event Hub, Kafka, Azure Data Lake, open source agents such as Fluentd/Fluent Bit, and a wide variety of cloud and on-premises data sources.

No complex data modeling - With Data Explorer, there is no need to build complex data models and no need for complex scripting to transform data before it's consumed.

No index maintenance - There is no need for maintenance tasks to optimize data for query performance and no need for index maintenance. With Data Explorer, all raw data is available immediately, allowing you to run high-performance and high-concurrency queries on your streaming and persistent data. You can use these queries to build near real-time dashboards and alerts, and connect operational analytics data with the rest of data analytics platform.

Democratizing data analytics - Data Explorer democratizes self-service, big data analytics with the intuitive Kusto Query Language (KQL) that provides the expressiveness and power of SQL with the simplicity of Excel. KQL is highly optimized for exploring raw telemetry and time series data by leveraging Data Explorer's best-in-class text indexing technology for efficient free-text and regex search, and comprehensive parsing capabilities for querying traces\text data and JSON semi-structured data including arrays and nested structures. KQL offers advanced time series support for creating, manipulating, and analyzing multiple time series with in-engine Python execution support for model scoring.

Proven technology at petabyte scale - Data Explorer is a distributed system with compute and storage that can scale independently, enabling analytics on gigabytes or petabytes of data.

Integrated - Azure Synapse Analytics provides interoperability across data between Data Explorer, Apache Spark, and SQL engines empowering data engineers, data scientists, and data analysts to easily, and securely, access and collaborate on the same data in the data lake.

When to use Azure Synapse Data Explorer?

Use Data Explorer as a data platform for building near real-time log analytics and IoT analytics solutions to:

Consolidate and correlate your logs and events data across on-premises, cloud, and third-party data sources.

Accelerate your AI Ops journey (pattern recognition, anomaly detection, forecasting, and more).

Replace infrastructure-based log search solutions to save cost and increase productivity.

Build IoT analytics solutions for your IoT data.

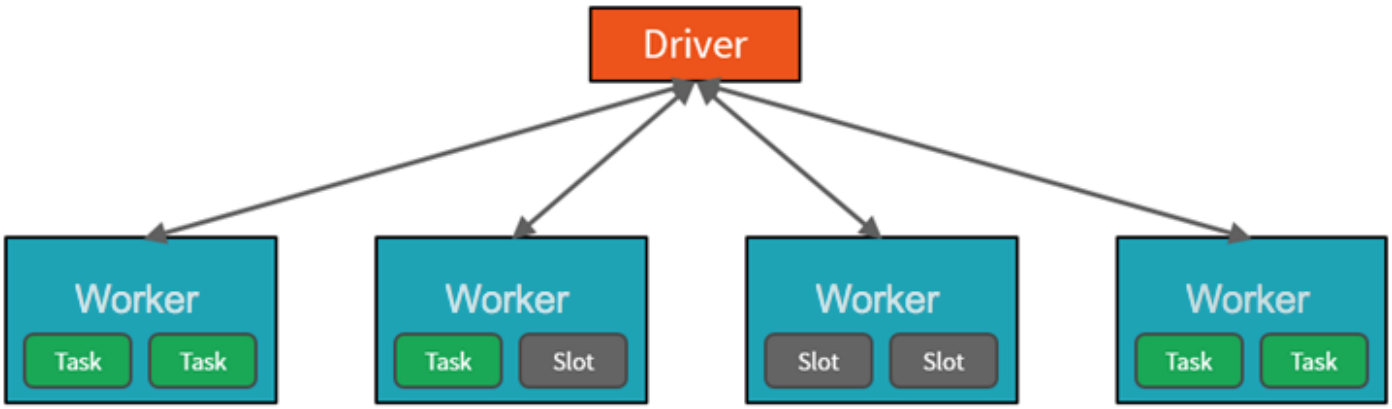
Build analytics SaaS solutions to offer services to your internal and external customers.

Spark streaming

High-level overview

From a high level, the Azure Databricks service launches and manages Apache Spark clusters within your Azure subscription. Apache Spark clusters are groups of computers that are treated as a single computer and handle the execution of commands issued from notebooks. Clusters enable processing of data to be parallelized across many computers to improve scale and performance. They consist of a Spark *driver* and *worker* nodes. The driver node sends work to the worker nodes and instructs them to pull data from a specified data source.

In Databricks, the notebook interface is typically the driver program. This driver program contains the main loop for the program and creates distributed datasets on the cluster, then applies operations to those datasets. Driver programs access Apache Spark through a *SparkSession* object regardless of deployment location.



Microsoft Azure manages the cluster, and auto-scales it as needed based on your usage and the setting used when configuring the cluster. Auto-termination can also be enabled, which allows Azure to terminate the cluster after a specified number of minutes of inactivity.

Spark jobs in detail

Work submitted to the cluster is split into as many independent jobs as needed. This is how work is distributed across the Cluster's nodes. Jobs are further subdivided into tasks. The input to a job is partitioned into one or more partitions. These partitions are the unit of work for each slot. In between tasks, partitions may need to be reorganized and shared over the network.

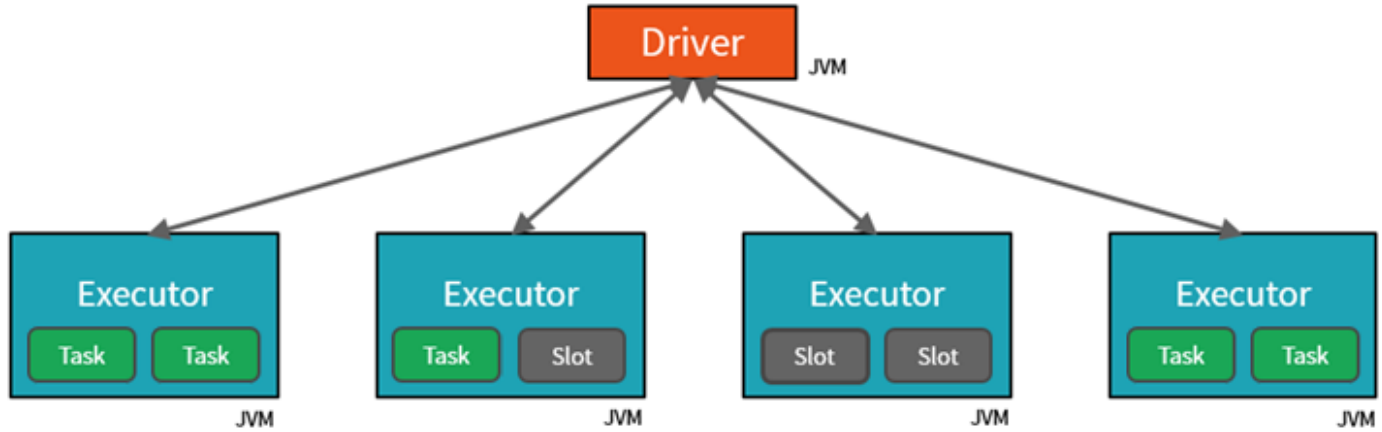
The secret to Spark's high performance is parallelism. Scaling *vertically* (by adding resources to a single computer) is limited to a finite amount of RAM, Threads and CPU speeds; but clusters scale *horizontally*, adding new nodes to the cluster as needed.

Spark parallelizes jobs at two levels:

The first level of parallelization is the *executor* - a Java virtual machine (JVM) running on a worker node, typically, one instance per node.

The second level of parallelization is the *slot* - the number of which is determined by the number of cores and CPUs of each node.

Each executor has multiple slots to which parallelized tasks can be assigned.



The JVM is naturally multi-threaded, but a single JVM, such as the one coordinating the work on the driver, has a finite upper limit. By splitting the work into tasks, the driver can assign units of work to *slots in the executors on worker nodes for parallel execution. Additionally, the driver determines how to partition the data so that it can be distributed for parallel processing. So, the driver assigns a partition of data to each task so that each task knows which piece of data it is to process. Once started, each task will fetch the partition of data assigned to it.

Jobs and stages
Depending on the work being performed, multiple parallelized jobs may be required. Each job is broken down into *stages*. A useful analogy is to imagine that the job is to build a house:
The first stage would be to lay the foundation.
The second stage would be to erect the walls.
The third stage would be to add the roof.
Attempting to do any of these steps out of order just doesn't make sense, and may in fact be impossible. Similarly, Spark breaks each job into stages to ensure everything is done in the right order.

Describe data visualization in Microsoft Power BI

• Identify capabilities of Power BI

While the process of data analysis focuses on the tasks of cleaning, modeling, and visualizing data, the concept of data analysis and its importance to business should not be understated. To analyze data, core components of analytics are divided into the following categories:

Descriptive analytics

Descriptive analytics help answer questions about what has happened based on historical data. Descriptive analytics techniques summarize large datasets to describe outcomes to stakeholders.
By developing key performance indicators (KPIs), these strategies can help track the success or failure of key objectives. Metrics such as return on investment (ROI) are used in many industries, and specialized metrics are developed to track performance in specific industries.
An example of descriptive analytics is generating reports to provide a view of an organization's sales and financial data.

Diagnostic analytics

Diagnostic analytics help answer questions about why events happened. Diagnostic analytics techniques supplement basic descriptive analytics, and they use the findings from descriptive analytics to discover the cause of these events. Then, performance indicators are further investigated to discover why these events improved or became worse. Generally, this process occurs in three steps:
Identify anomalies in the data. These anomalies might 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

Predictive analytics help answer questions about what will happen in the future. Predictive analytics techniques use historical data to identify trends and determine if they're likely to recur. Predictive analytical tools provide valuable insight into what might happen in the future. Techniques include a variety of statistical and machine learning techniques such as neural networks, decision trees, and regression.

Prescriptive analytics

Prescriptive analytics help answer questions about which actions should be taken to achieve a goal or target. By using insights from prescriptive analytics, organizations can make data-driven decisions. This technique allows businesses to make informed decisions in the face of uncertainty. Prescriptive analytics techniques rely on machine learning as one of the strategies to find patterns in large datasets. By analyzing past decisions and events, organizations can estimate the likelihood of different outcomes.

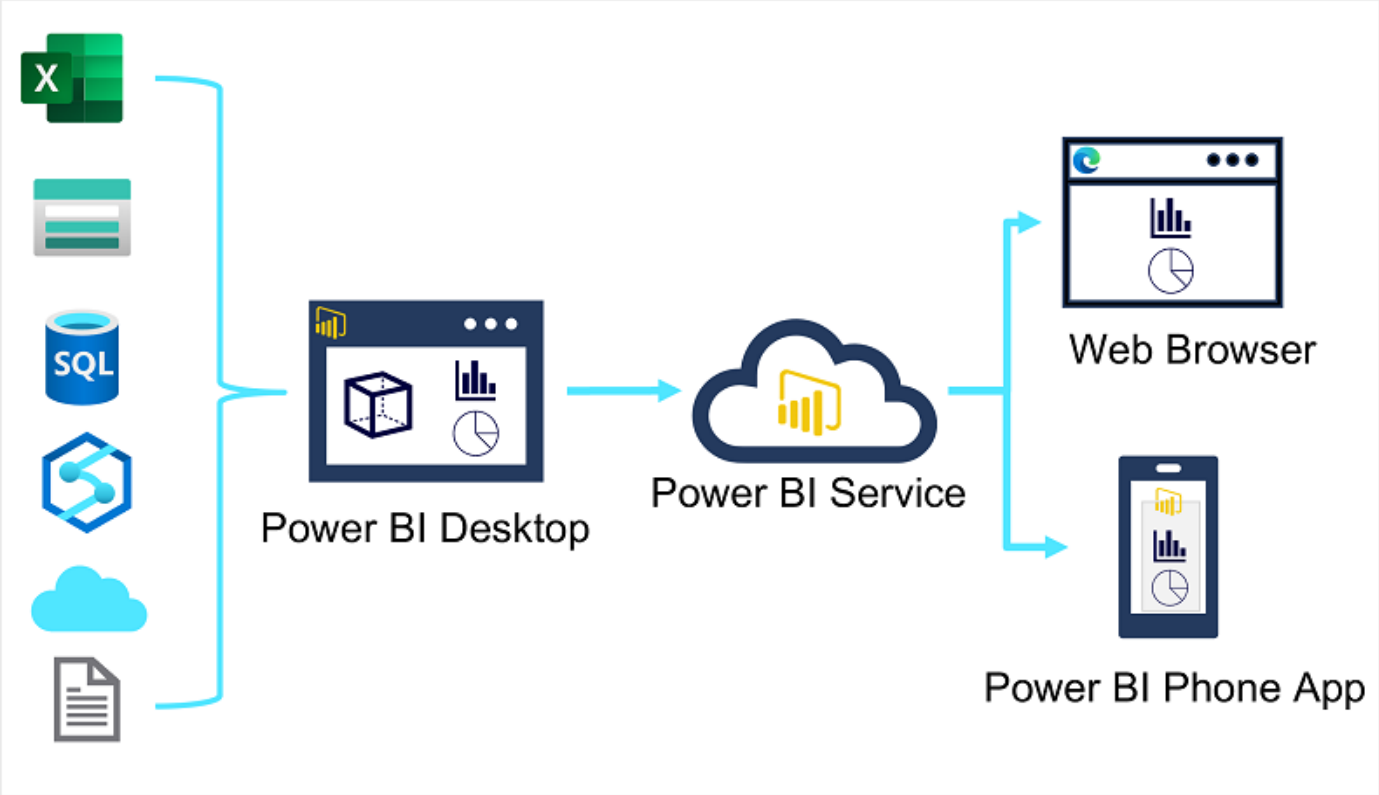
Cognitive analytics

Cognitive analytics attempt to draw inferences from existing data and patterns, derive conclusions based on existing knowledge bases, and then add these findings back into the knowledge base for future inferences, a self-learning feedback loop. Cognitive analytics help you learn what might happen if circumstances change and determine how you might handle these situations.
Inferences aren't structured queries based on a rules database; rather, they're unstructured hypotheses that are gathered from several sources and expressed with varying degrees of confidence. Effective cognitive analytics depend on machine learning algorithms, and will use several natural language processing concepts to make sense of previously untapped data sources, such as call center conversation logs and product reviews.

• Describe features of data models in Power BI

These three elements—**Desktop**, the **service**, and **Mobile** apps—are designed to let people create, share, and consume business insights in the way that serves them, or their role, most effectively.

The flow of work in Power BI



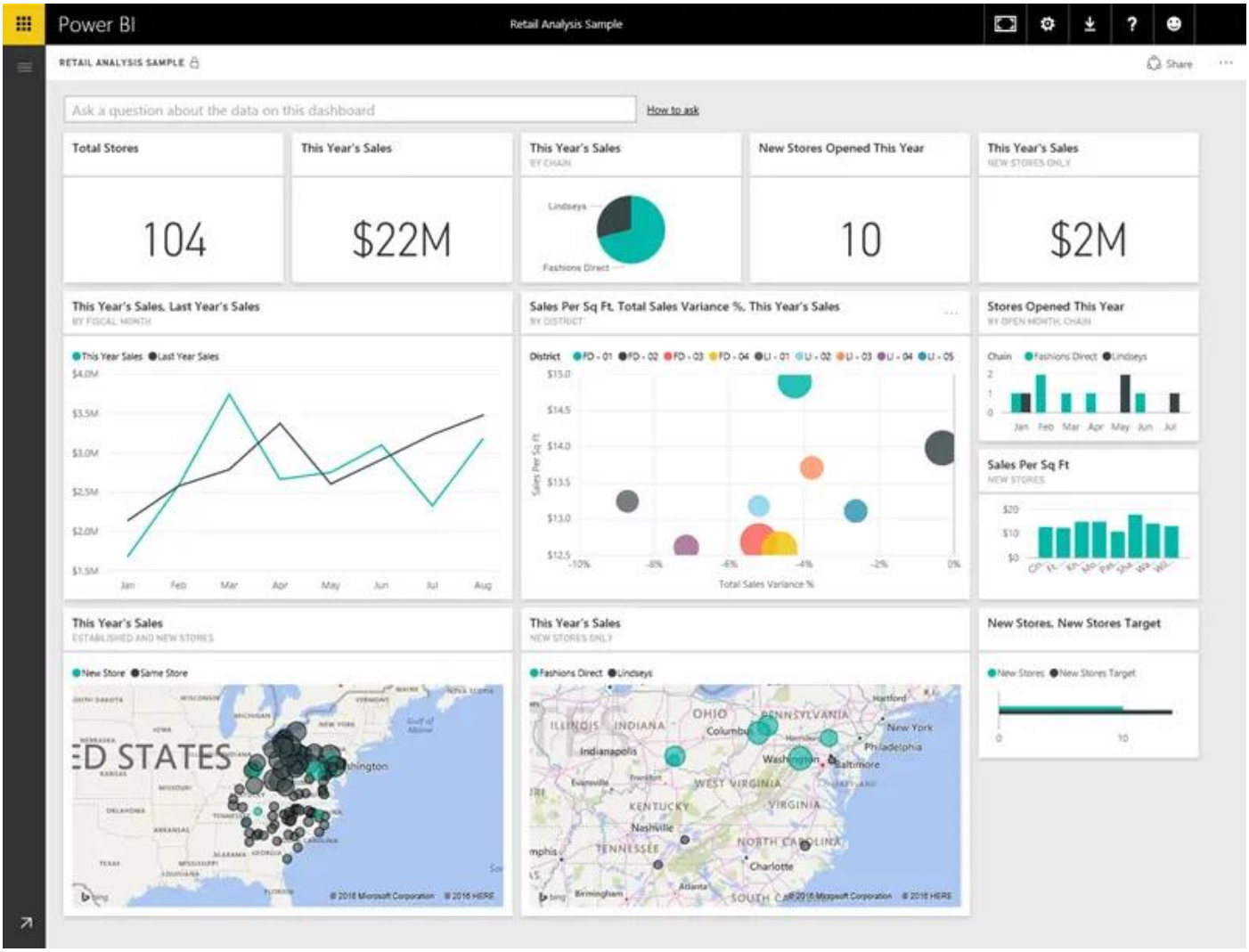
A common flow of work in Power BI begins in **Power BI Desktop**, where a report is created. That report is then published to the **Power BI service** and finally shared, so that users of **Power BI Mobile** apps can consume the information.
It doesn't always happen that way, and that's okay. But we'll use that flow to help you learn the different parts of Power BI and how they complement each other.
Okay, now that we have an overview of this module, what Power BI is, and its three main elements, let's take a look at what it's like to use **Power BI**.

The activities and analyses that you'll learn with Power BI generally follow a common flow. The **common flow** of activity looks like this:
Bring data into Power BI Desktop, and create a report.
Publish to the Power BI service, where you can create new visualizations or build dashboards.
Share dashboards with others, especially people who are on the go.
View and interact with shared dashboards and reports in Power BI Mobile apps.

• Identify appropriate visualizations for data
Here are the basic building blocks in Power BI:
Visualizations
Datasets
Reports
Dashboards
Tiles

Visualizations

A **visualization** (sometimes also referred to as a **visual**) is a visual representation of data, like a chart, a color-coded map, or other interesting things you can create to represent your data visually. Power BI has all sorts of visualization types, and more are coming all the time. The following image shows a collection of different visualizations that were created in Power BI.



Visualizations can be simple, like a single number that represents something significant, or they can be visually complex, like a gradient-colored map that shows voter sentiment about a certain social issue or concern. The goal of a visual is to present data in a way that provides context and insights, both of which would probably be difficult to discern from a raw table of numbers or text.

Datasets

A **dataset** is a collection of data that Power BI uses to create its visualizations.

You can have a simple dataset that's based on a single table from a Microsoft Excel workbook, similar to what's shown in the following image.

C2132 | 2

	B	C	D	E	F	G	H
1	Year	Month	Month Name	Calendar Month	Births	Births Per Day	Births (Normalized)
2119	2004	1	January	1/1/2004	2,937	94.7	2842
2120	2004	2	February	2/1/2004	2,824	97.4	2921
2121	2004	3	March	3/1/2004	3,128	100.9	3027
2122	2004	4	April	4/1/2004	2,896	96.5	2896
2123	2004	5	May	5/1/2004	3,008	97.0	2911
2124	2004	6	June	6/1/2004	3,047	101.6	3047
2125	2004	7	July	7/1/2004	2,981	96.2	2885
2126	2004	8	August	8/1/2004	3,079	99.3	2980
2127	2004	9	September	9/1/2004	3,219	107.3	3219
2128	2004	10	October	10/1/2004	3,547	114.4	3433
2129	2004	11	November	11/1/2004	3,365	112.2	3365
2130	2004	12	December	12/1/2004	3,143	101.4	3042
2131	2005	1	January	1/1/2005	2,921	94.2	2827
2132	2005	2	February	2/1/2005	2,699	96.4	2892
2133	2005	3	March	3/1/2005	3,024	97.5	2926
2134	2005	4	April	4/1/2005	3,037	101.2	3037
2135	2005	5	May	5/1/2005	3,231	104.2	3127
2136	2005	6	June	6/1/2005	3,163	105.4	3163
2137	2005	7	July	7/1/2005	3,119	100.6	3018
2138	2005	8	August	8/1/2005	3,156	101.8	3054
2139	2005	9	September	9/1/2005	3,439	114.6	3439

Datasets can also be a combination of many different sources, which you can filter and combine to provide a unique collection of data (a dataset) for use in Power BI.

For example, you can create a dataset from three database fields, one website table, an Excel table, and online results of an email marketing campaign. That unique combination is still considered a single **dataset**, even though it was pulled together from many different sources.

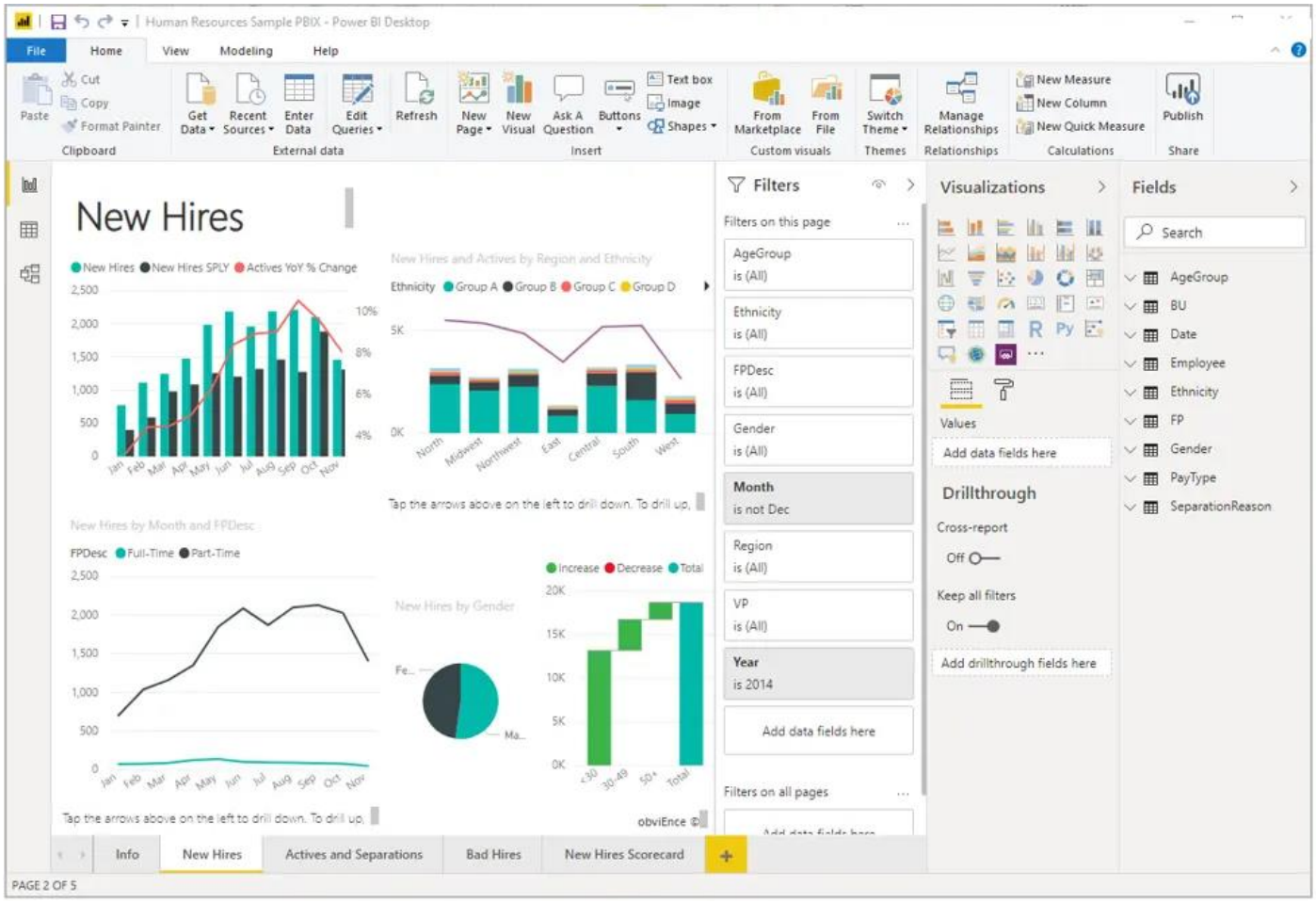
Filtering data before bringing it into Power BI lets you focus on the data that matters to you. For example, you can filter your contact database so that only customers who received emails from the marketing campaign are included in the dataset. You can then create visuals based on that subset (the filtered collection) of customers who were included in the campaign. Filtering helps you focus your data—and your efforts.

An important and enabling part of Power BI is the multitude of data **connectors** that are included. Whether the data you want is in Excel or a Microsoft SQL Server database, in Azure or Oracle, or in a service like Facebook, Salesforce, or MailChimp, Power BI has built-in data connectors that let you easily connect to that data, filter it if necessary, and bring it into your dataset.

After you have a dataset, you can begin creating visualizations that show different portions of it in different ways, and gain insights based on what you see. That's where reports come in.

Reports

In Power BI, a **report** is a collection of visualizations that appear together on one or more pages. Just like any other report you might create for a sales presentation or write for a school assignment, a report in Power BI is a collection of items that are related to each other. The following image shows a **report** in Power BI Desktop—in this case, it's the second page in a five-page report. You can also create reports in the Power BI service.



Reports let you create many visualizations, on multiple pages if necessary, and let you arrange those visualizations in whatever way best tells your story.

You might have a report about quarterly sales, product growth in a particular segment, or migration patterns of polar bears. Whatever your subject, reports let you gather and organize your visualizations onto one page (or more).

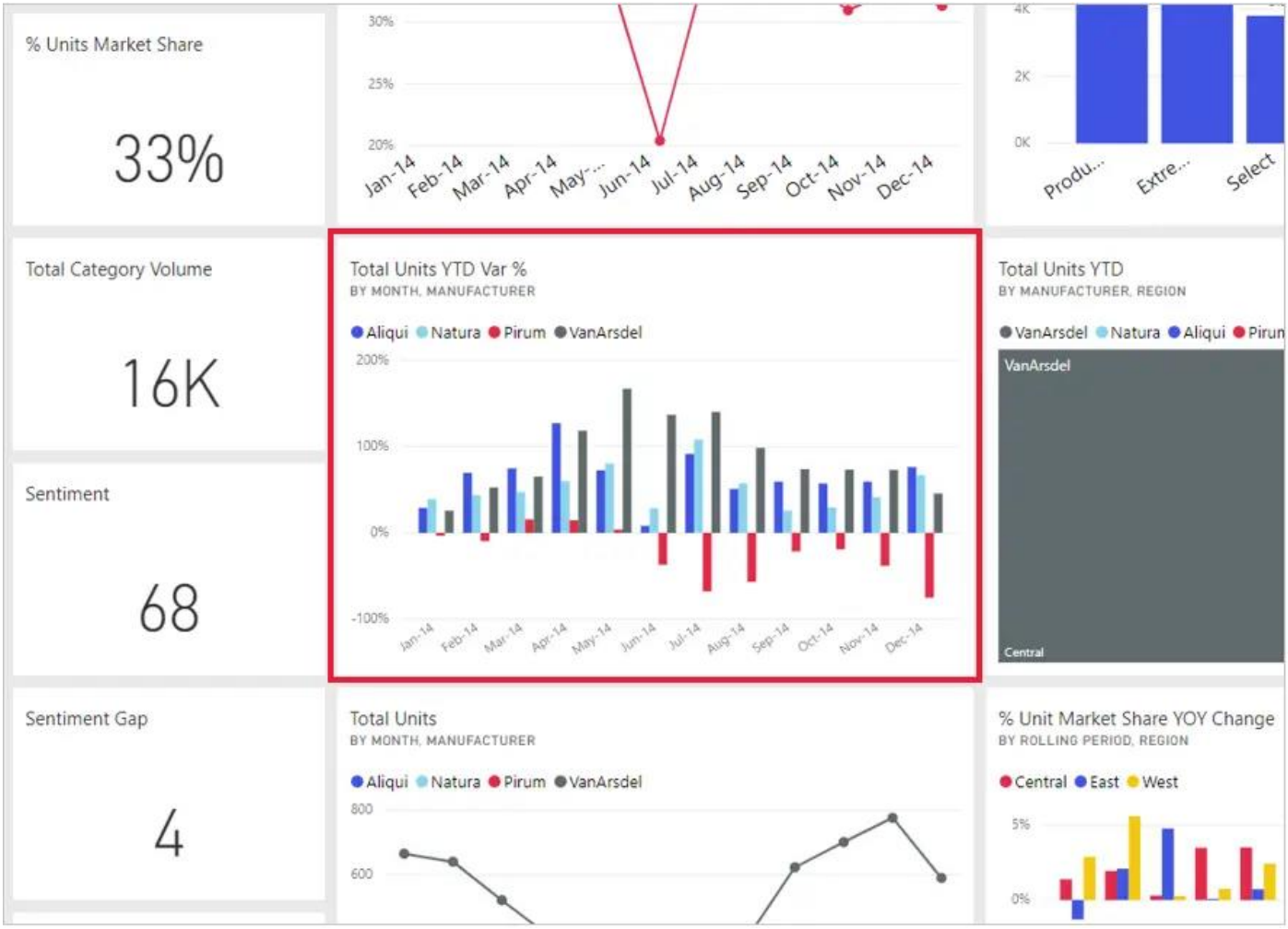
Dashboards

When you're ready to share a report, or a collection of visualizations, you create a **dashboard**. Much like the dashboard in a car, a Power BI **dashboard** is a collection of visuals that you can share with others. Often, it's a selected group of visuals that provide quick insight into the data or story you're trying to present.

A dashboard must fit on a single page, often called a canvas (the canvas is the blank backdrop in Power BI Desktop or the service, where you put visualizations). Think of it like the canvas that an artist or painter uses—a workspace where you create, combine, and rework interesting and compelling visuals. You can share dashboards with other users or groups, who can then interact with your dashboards when they're in the Power BI service or on their mobile device.

Tiles

In Power BI, a **tile** is a single visualization on a dashboard. It's the rectangular box that holds an individual visual. In the following image, you see one tile, which is also surrounded by other tiles.

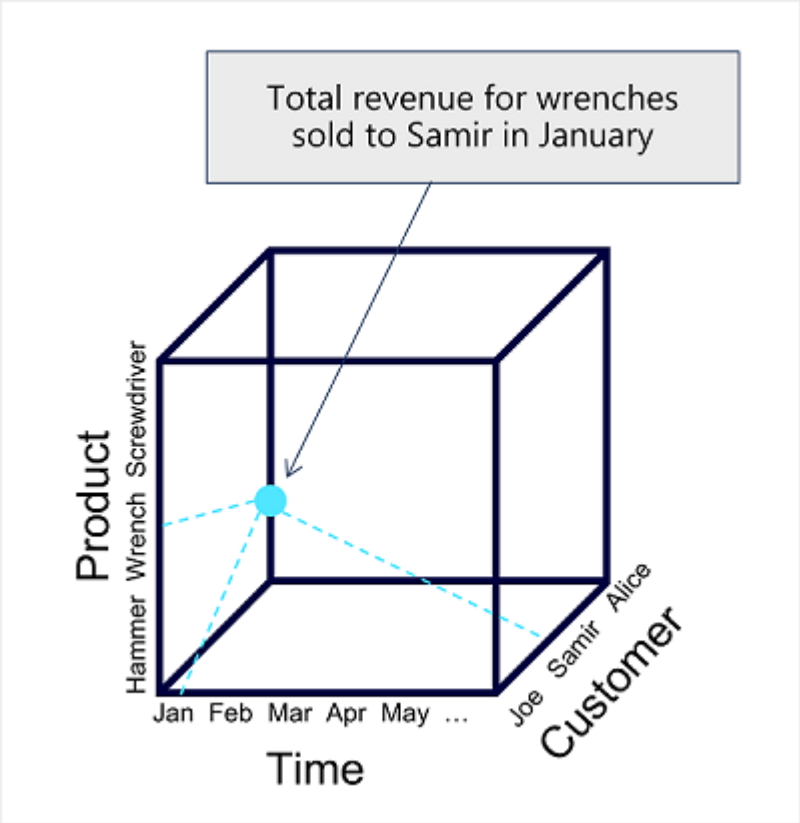


When you're *creating* a dashboard in Power BI, you can move or arrange tiles however you want. You can make them bigger, change their height or width, and snuggle them up to other tiles.

When you're *viewing*, or *consuming*, a dashboard or report—which means you're not the creator or owner, but the report or dashboard has been shared with you—you can interact with it, but you can't change the size of the tiles or their arrangement.

Describe core concepts of data modeling

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*). For example, a model might include a table containing numeric measures for sales (such as revenue or quantity) and dimensions for products, customers, and time. This would enable you aggregate sale measures across one or more dimensions (for example, to identify total revenue by customer, or total items sold by product per month). 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.)



Note

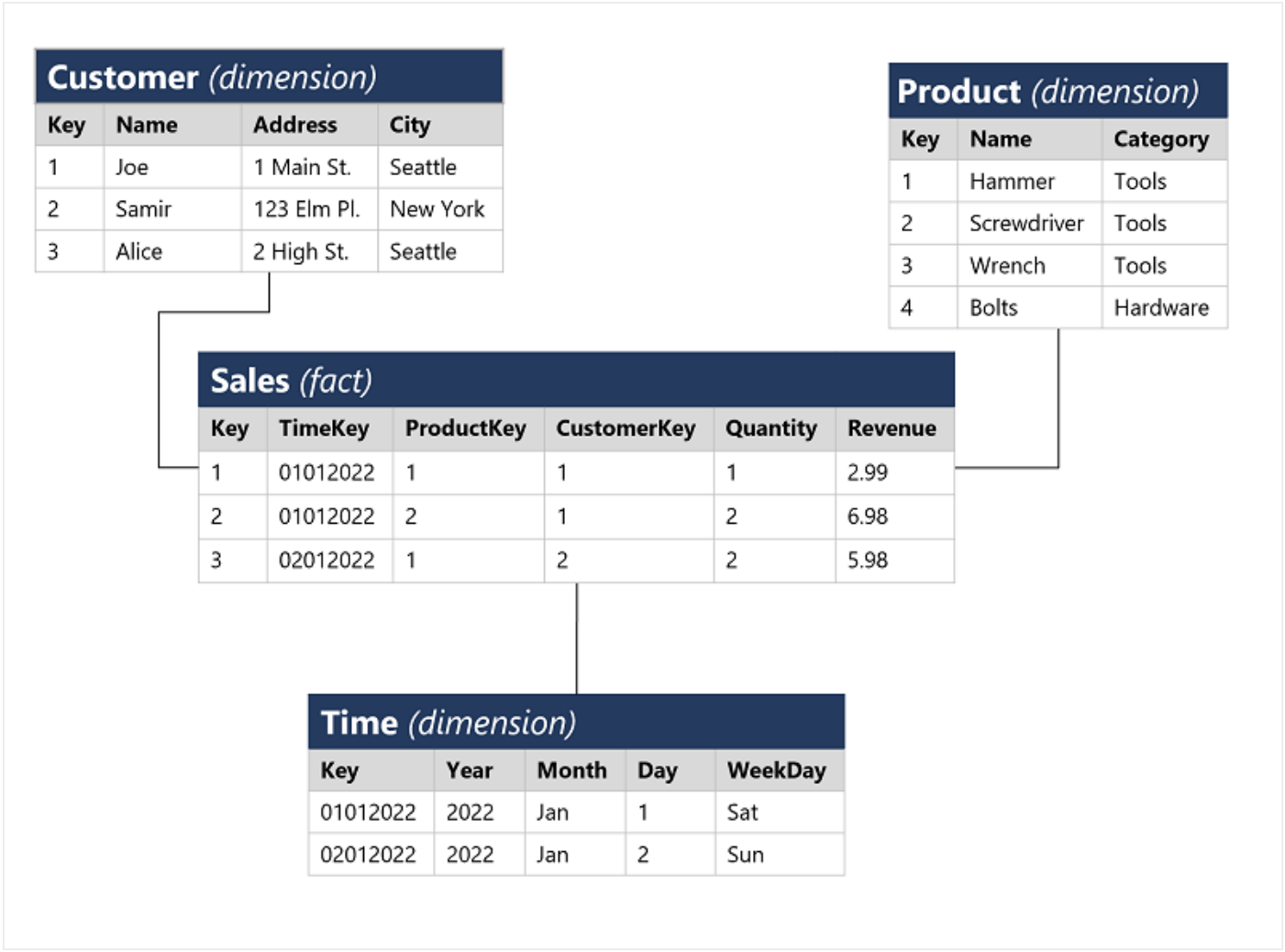
Although we commonly refer to an analytical model as a *cube*, there can be more (or fewer) than three dimensions – it's just not easy for us to visualize more than three!

Tables and schema

Dimension tables represent the entities by which you want to aggregate numeric measures – for example product or customer. Each entity is represented by a row with a unique key value. The remaining columns represent attributes of an entity – for example, products have names and categories, and customers have addresses and cities.

It's common in most analytical models to include a *Time* dimension so that you can aggregate numeric measures associated with events over time.

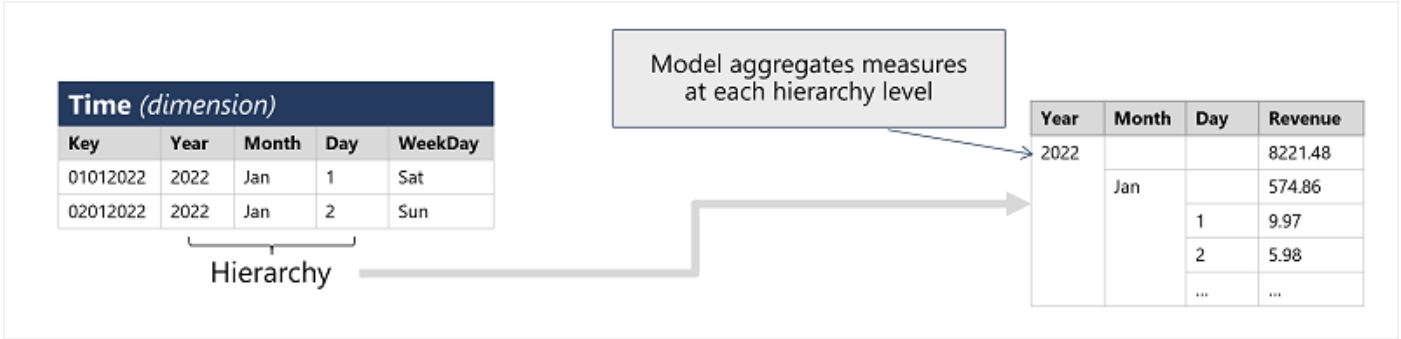
The numeric measures that will be aggregated by the various dimensions in the model are stored in *Fact* tables. Each row in a fact table represents a recorded event that has numeric measures associated with it. For example, the **Sales** table in the schema below represents sales transactions for individual items, and includes numeric values for quantity sold and revenue.



This type of schema, where a fact table is related to one or more dimension tables, is referred to as a star schema (imagine there are five dimensions related to a single fact table – the schema would form a five-pointed star!). You can also define a more complex schema in which dimension tables are related to additional tables containing more details (for example, you could represent attributes of product categories in a separate **Category** table that is related to the **Product** table – in which case the design is referred to as a snowflake schema. The schema of fact and dimension tables is used to create an analytical model, in which measure aggregations across all dimensions are pre-calculated; making performance of analysis and reporting activities much faster than calculating the aggregations each time.)

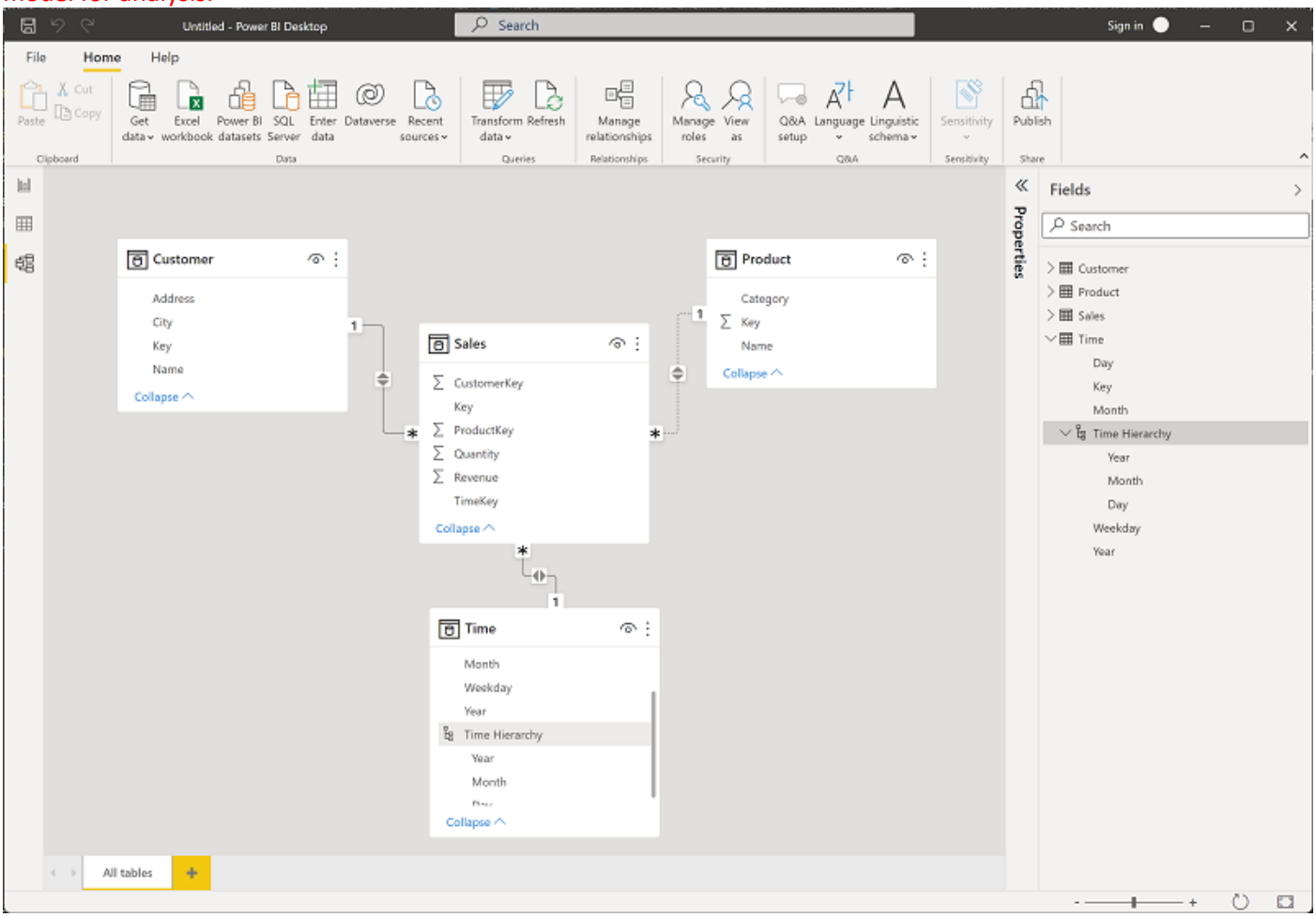
Attribute hierarchies

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. Similarly, in the **Customer** table, a hierarchy could be formed to represent multiple named customers in each city. Finally, in the **Time** table, you can form a hierarchy of year, month, and day. The model can be built with pre-aggregated values for each level of a hierarchy, enabling you to quickly change the scope of your analysis – for example, by viewing total sales by year, and then drilling down to see a more detailed breakdown of total sales by month.



Analytical modeling in Microsoft Power BI

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.



Describe considerations for data visualization

After you've created a model, you can use it to generate data visualizations that can be included in a report.
There are many kinds of data visualization, some commonly used and some more specialized. Power BI includes an extensive set of built-in visualizations, which can be extended with custom and third-party visualizations. The rest of this unit discusses some common data visualizations but is by no means a complete list.
Tables and text

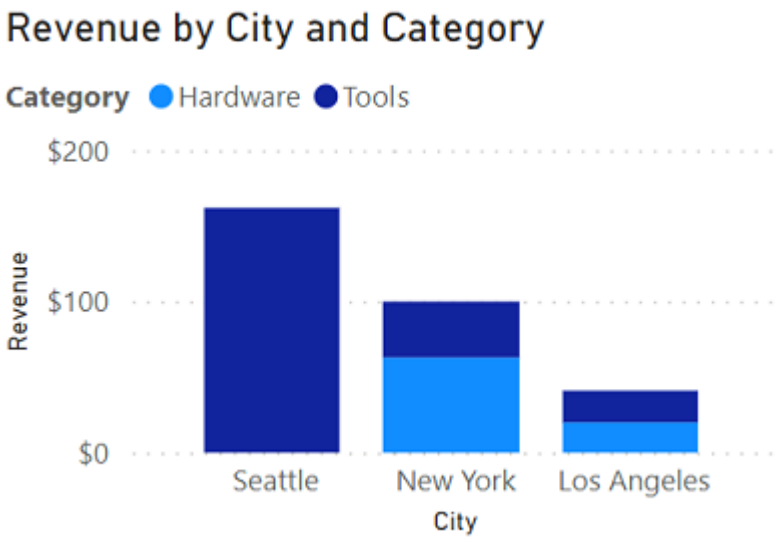
Product Sales

Name	Quantity
Bolts	2
Hammer	4
Nails	1
Screwdriver	2
Screws	2
Wrench	4
Total	**15**

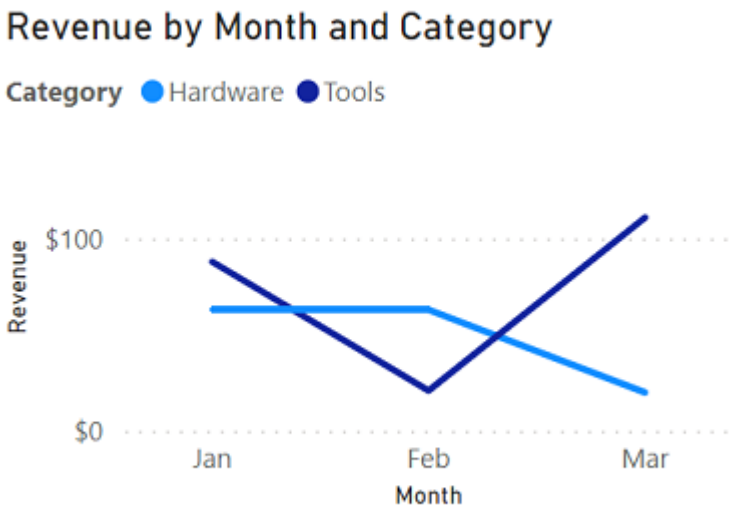
$302.91

Revenue

Tables and text are often the simplest way to communicate data. Tables are useful when numerous related values must be displayed, and individual text values in cards can be a useful way to show important figures or metrics.
Bar and column charts

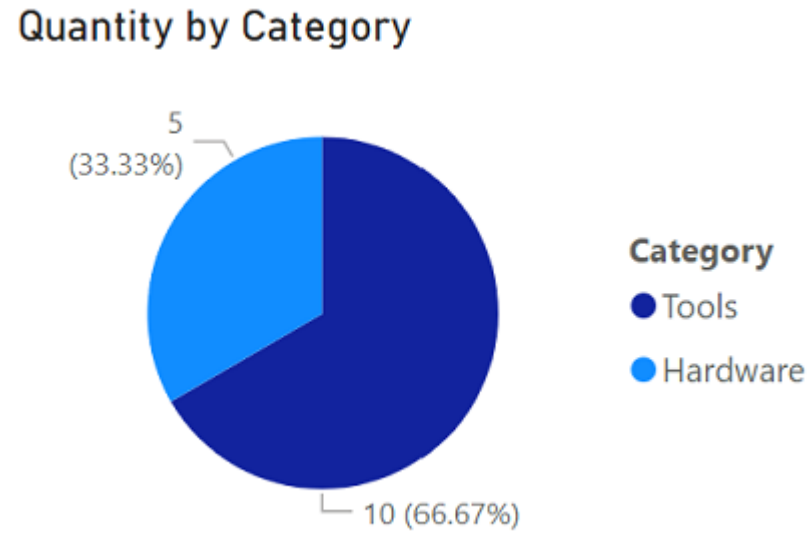


Bar and column charts are a good way to visually compare numeric values for discrete categories.
Line charts



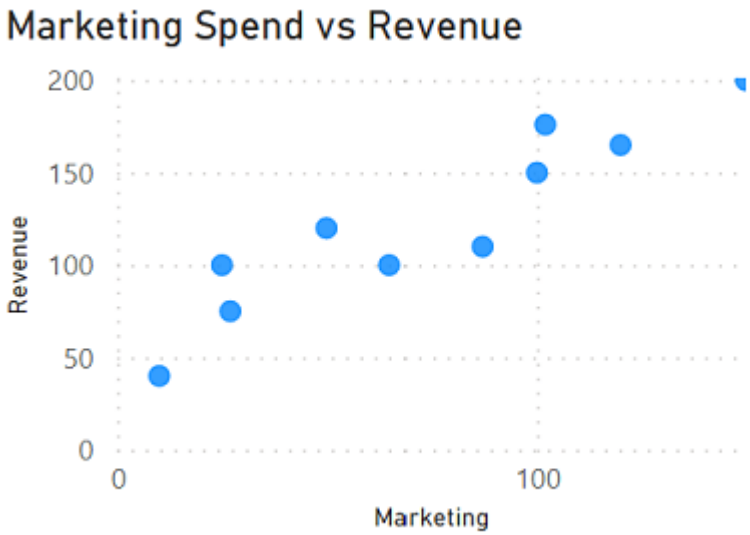
Line charts can also be used to compare categorized values and are useful when you need to examine trends, often over time.

Pie charts



Pie charts are often used in business reports to visually compare categorized values as proportions of a total.

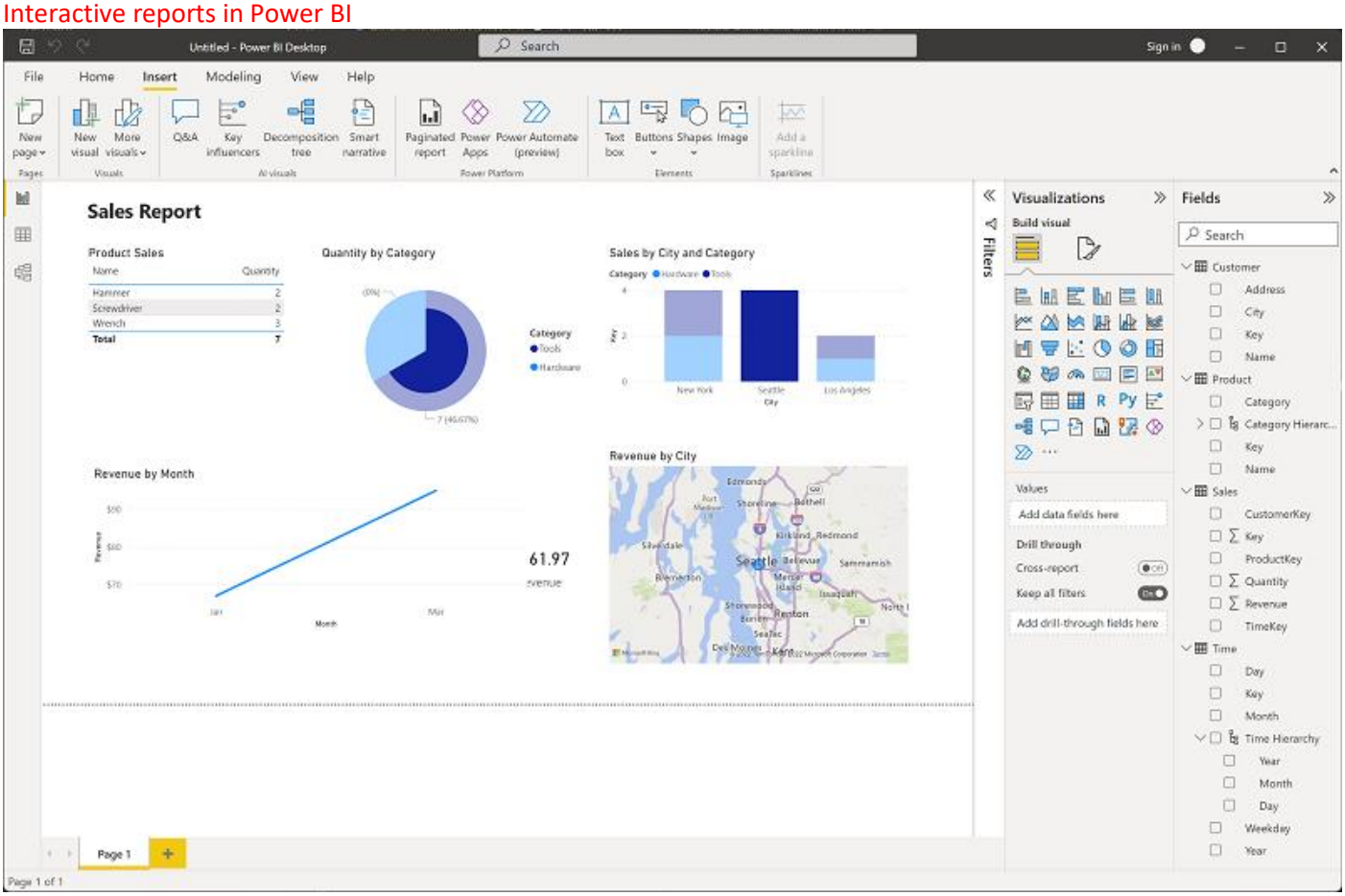
Scatter plots



Scatter plots are useful when you want to compare two numeric measures and identify a relationship or correlation between them.

Maps

Maps are a great way to visually compare values for different geographic areas or locations.
Interactive reports in Power BI



In Power BI, the visual elements for related data in a report are automatically linked to one another and provide interactivity. For example, selecting an individual category in one visualization will automatically filter and highlight that category in other related visualizations in the report. In the image above, the city *Seattle* has been selected in the **Sales by City and Category** column chart, and the other visualizations are filtered to reflect values for Seattle only.