

Domain 4: Introduction to Microsoft Azure

Data analytics in Azure

Explore fundamentals of large-scale analytics

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/1-introduction>

Introduction

Completed

Large-scale data analytics 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, however, 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. The combination of flexible data lake storage and data warehouse SQL analytics has led to the emergence of a large-scale analytics design often called a *data lakehouse*.

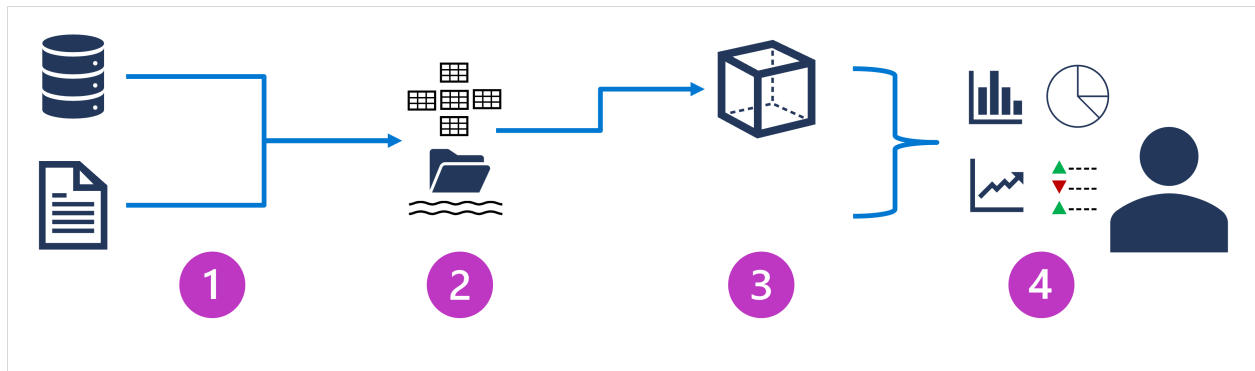
2. Describe data warehousing architecture

<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/2-describe-warehousing>

Describe data warehousing architecture

Completed

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



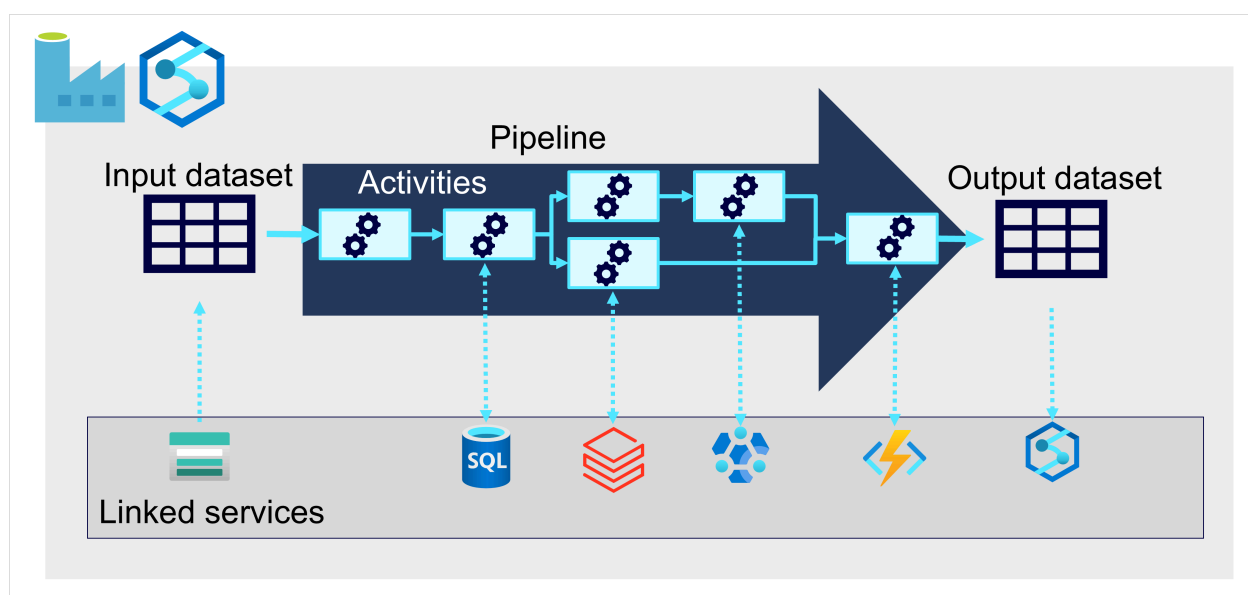
1. **Data ingestion and processing** – data from one or more transactional data stores, files, real-time streams, or other sources is loaded into a data lake or a relational data warehouse. The load operation usually involves an *extract, transform, and load* (ETL) or *extract, load, and transform* (ELT) 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.

3. Explore data ingestion pipelines

Explore data ingestion pipelines

Completed

Now that you understand a little about the architecture of a large-scale data warehousing solution, and some of the distributed processing technologies that can be used to handle large volumes of data, it's time to explore how data is ingested into an analytical data store from one or more sources.



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](#), or you can use the pipeline capability in [Microsoft Fabric](#) 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. For example, you might use an Azure Blob Store linked service to ingest the input dataset, and then use services such as Azure SQL Database to run a stored procedure that looks up related data values, before running a data processing task on Azure Databricks, or apply custom logic using an Azure Function. Finally, you can save the output dataset to a destination such as a Microsoft Fabric Lakehouse or Data Warehouse. Pipelines can also include some built-in activities, which don't require a linked service.

4. Explore analytical data stores

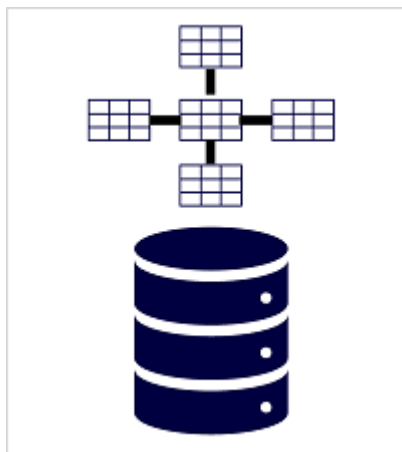
<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/4-analytical-data-stores>

Explore analytical data stores

Completed

There are two common types of analytical data store.

Data warehouses



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. For example a fact table might contain sales order data, which can be aggregated by customer, product, store, and time dimensions (enabling you, for example, to easily find monthly total sales revenue by product for each store). 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.

Data lakes



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.

Hybrid approaches

You can use a hybrid approach that combines features of data lakes and data warehouses in a *data lakehouse*. The raw data is stored as files in a data lake, and Microsoft Fabric SQL analytics endpoints expose them as tables, which can be queried using SQL. When you create a Lakehouse with Microsoft Fabric, a SQL analytics endpoint is automatically created. 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.

Azure services for analytical stores

On Azure, there are several services that you can use to implement a large-scale analytical store including:



Microsoft Fabric 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 Microsoft Fabric Real-Time Intelligence, as well as built in data pipelines for data ingestion and transformation. Each Microsoft Fabric product experience has its own home, for

example the Data Factory Home. Each Fabric Home displays the items that you create and have permission to use from all the workspace that you access. Microsoft Fabric is a great choice when you want to create a single, unified analytics solution.



Azure 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 multicloud environment or support a cloud-portable solution.

Note

Each of these services can be thought of as an analytical data *store*, in the sense that they provide a schema and interface through which the data can be queried. In many cases however, the data is actually stored in a data lake and the service is used to *process* the data and run queries. Some solutions might even combine the use of these services. An *extract, load, and transform* (ELT) ingestion process might copy data into the data lake, and then use one of these services to transform the data, and another to query it. For example, a pipeline might use a notebook running in Azure Databricks to process a large volume of data in the data lake, and then load it into tables in a Microsoft Fabric Warehouse.

5. Exercise: Explore data analytics in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/5-exercise-data-analytics-fabric>

Exercise: Explore data analytics in Microsoft Fabric

Completed

In this exercise, you create a Microsoft Fabric data lakehouse and use it to ingest and analyze some data.

The exercise is designed to familiarize you with some key elements of a large-scale data analytics solution, not as a comprehensive guide to performing advanced data analysis with Microsoft Fabric. The exercise should take around 30 minutes to complete.

Note

To complete this lab, you will need a Microsoft Fabric account, or if you don't have an account yet, sign up for a [free trial](#).

Go to [Getting started with Fabric](#) for more information.

Launch the exercise and follow the instructions.

Launch Exercise

6. Knowledge check

<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/6-knowledge-check>

Knowledge check

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/examine-components-of-modern-data-warehouse/7-summary>

Summary

Completed

Large-scale data warehousing is a complex workload that can involve many different technologies. This module provided a high-level overview of the key features of a modern data warehousing solution, and explored some of the services in Azure that you can use to implement one.

In this module, you learned how to:

- Identify common elements of a large-scale data warehousing solution
- Describe key features for data ingestion pipelines
- Identify common types of analytical data store and related Azure services
- Provision Microsoft Fabric and use it to ingest, process, and query data

Next steps

Now that you learned about large-scale data warehousing, consider learning more about data-related workloads on Azure by pursuing a Microsoft certification in [Azure Data Fundamentals](#).

Explore fundamentals of real-time analytics

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/1-introduction>

Introduction

Completed

Increased use of technology by individuals, companies, and other organizations, together with the proliferation of smart devices and Internet access has led to a massive growth in the volume of data that can be generated, captured, and analyzed. Much of this data can be processed in real-time (or at least, *near* real-time) as a perpetual *stream* of data, enabling the creation of systems that reveal instant insights and trends, or take immediate responsive action to events as they occur.

Note

This module is designed to present a conceptual overview of real-time processing and describe Azure services that can be used to build real-time analytics solutions. It is not intended to teach implementation details for creating a stream processing solution.

2. Understand batch and stream processing

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/2-batch-stream>

Understand batch and stream processing

Completed

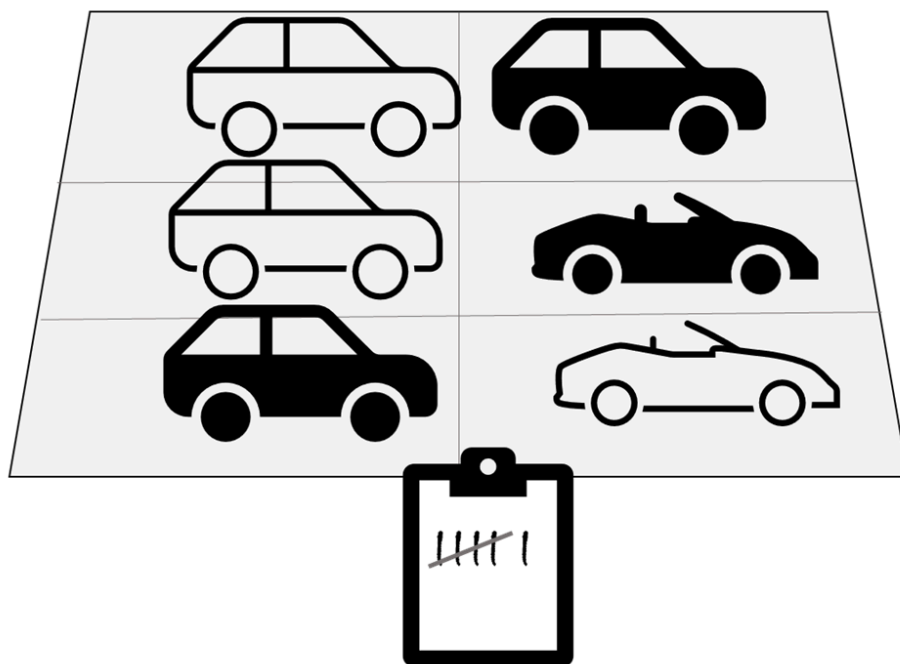
Data processing is simply the conversion of raw data to meaningful information through a process. There are two general ways to process data:

- *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.

Understand batch processing

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.

For example, suppose you want to analyze road traffic by counting the number of cars on a stretch of road. A batch processing approach to this would require that you collect the cars in a parking lot, and then count them in a single operation while they're at rest.



If the road is busy, with a large number of cars driving along at frequent intervals, this approach may be impractical; and note that you don't get any results until you have parked a batch of cars and counted them.

A real world example of batch processing is the way that credit card companies handle billing. The customer doesn't receive a bill for each separate credit card purchase but one monthly bill for all of that month's purchases.

Advantages of batch processing include:

- Large volumes of data can be processed at a convenient time.

- It can be scheduled to run at a time when computers or systems might otherwise be idle, such as overnight, or during off-peak hours.

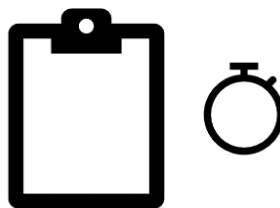
Disadvantages of batch processing include:

- The time delay between ingesting the data and getting the results.
- All of a batch job's input data must be ready before a batch can be processed. This means data must be carefully checked. Problems with data, errors, and program crashes that occur during batch jobs bring the whole process to a halt. The input data must be carefully checked before the job can be run again. Even minor data errors can prevent a batch job from running.

Understand stream processing

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.

For example, a better approach to our hypothetical car counting problem might be to apply a *streaming* approach, by counting the cars in real-time as they pass:



In this approach, you don't need to wait until all of the cars have parked to start processing them, and you can aggregate the data over time intervals; for example, by counting the number of cars that pass each minute.

Real world examples of streaming data include:

- A financial institution tracks changes in the stock market in real time, computes value-at-risk, and automatically rebalances portfolios based on stock price movements.

- An online gaming company collects real-time data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real time, offers incentives and dynamic experiences to engage its players.
- A real-estate website that tracks a subset of data from mobile devices, and makes real-time property recommendations of properties to visit based on their geo-location.

Stream processing is ideal for time-critical operations that require an instant real-time response. For example, a system that monitors a building for smoke and heat needs to trigger alarms and unlock doors to allow residents to escape immediately in the event of a fire.

Understand differences between batch and streaming data

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

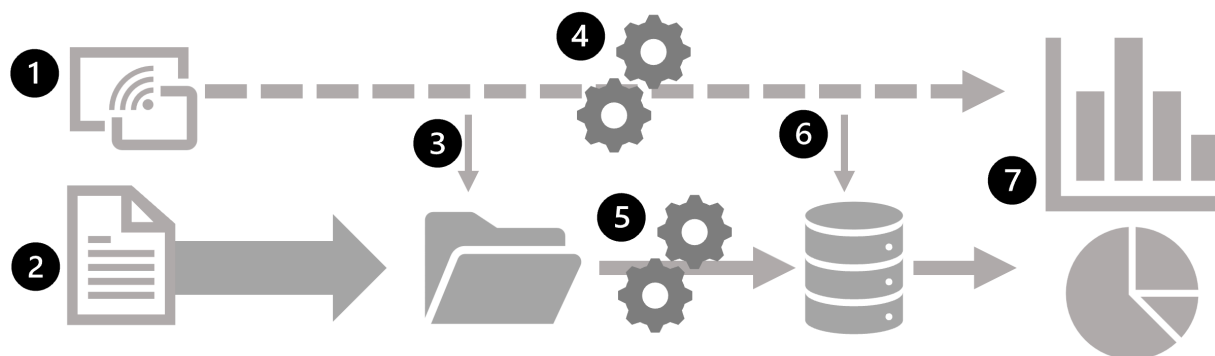
- *Data scope*: Batch processing can process all the data in the dataset. Stream processing typically only has access to the most recent data received, or within a rolling time window (the last 30 seconds, for example).
- *Data size*: Batch processing is suitable for handling large datasets efficiently. 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.

Combine batch and stream processing

Many large-scale analytics solutions include a mix of batch and stream processing, enabling both historical and real-time data analysis. It's common for stream processing solutions to capture real-time data, process it by filtering or aggregating it, and present it through real-time dashboards and visualizations (for example, showing the running total of cars that have passed along a road within the current hour), while also persisting the processed results in a data store for historical analysis alongside batch processed data (for example, to enable analysis of traffic volumes over the past year).

Even when real-time analysis or visualization of data is not required, streaming technologies are often used to capture real-time data and store it in a data store for subsequent batch processing (this is the equivalent of redirecting all of the cars that travel along a road into a parking lot before counting them).

The following diagram shows some ways in which batch and stream processing can be combined in a large-scale data analytics architecture.



1. Data events from a streaming data source are captured in real-time.
2. Data from other sources is ingested into a data store (often a *data lake*) for batch processing.
3. If real-time analytics is not required, the captured streaming data is written to the data store for subsequent batch processing.
4. When real-time analytics is required, a stream processing technology is used to prepare the streaming data for real-time analysis or visualization; often by filtering or aggregating the data over temporal windows.
5. The non-streaming data is periodically batch processed to prepare it for analysis, and the results are persisted in an analytical data store (often referred to as a *data warehouse*) for historical analysis.
6. The results of stream processing may also be persisted in the analytical data store to support historical analysis.
7. Analytical and visualization tools are used to present and explore the real-time and historical data.

Note

Commonly used solution architectures for combined batch and stream data processing include *lambda* and *delta* architectures. Details of these architectures are beyond the scope of this course, but they incorporate technologies for both large-scale batch data processing and real-time stream processing to create an end-to-end analytical solution.

3. Explore common elements of stream processing architecture

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/3-explore-common-elements>

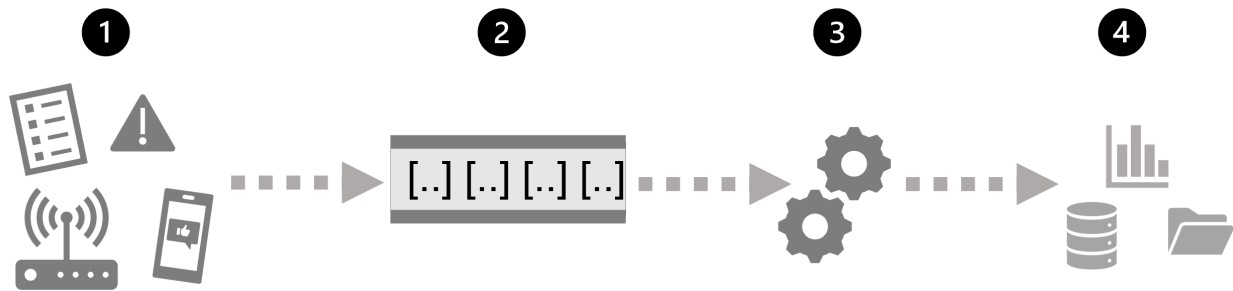
Explore common elements of stream processing architecture

Completed

There are many technologies that you can use to implement a stream processing solution, but while specific implementation details may vary, there are common elements to most streaming architectures.

A general architecture for stream processing

At its simplest, 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.

Real-time analytics services

Microsoft 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 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 **Microsoft Fabric** and **Azure Databricks**.

- **Microsoft Fabric:** A high-performance database and analytics platform that includes Data Engineering, Data Factory, Data Science, Real-Time Intelligence, Data Warehouse, and Databases.

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

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, Microsoft OneLake, or Azure blob storage:** Used to persist the processed results as a file.
- **Azure SQL Database, Azure Databricks, or Microsoft Fabric:** Used to persist the processed results in a table for querying and analysis.
- **Microsoft Power BI:** Used to generate real time data visualizations in reports and dashboards.

4. Explore Microsoft Fabric Real-Time Intelligence

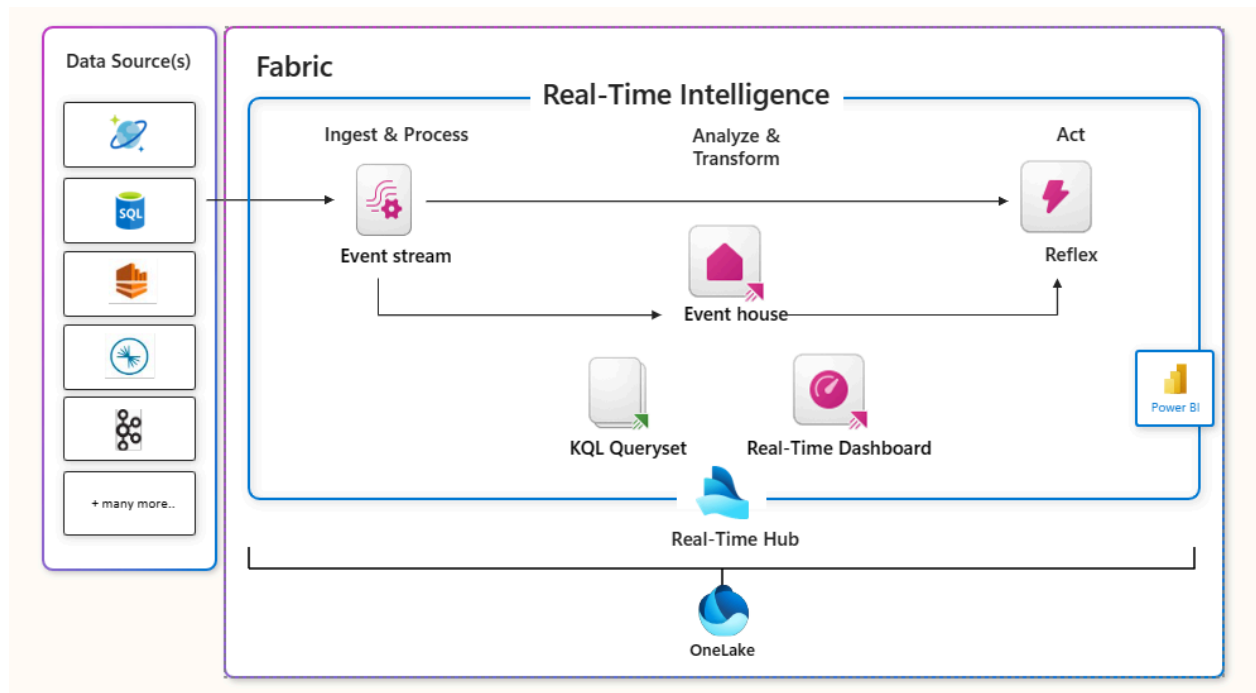
<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/4-fabric-real-time-intelligence>

Explore Microsoft Fabric Real-Time Intelligence

Completed

Microsoft Fabric Real-Time Intelligence empowers organizations to extract insights and visualize data in motion. Real-Time Intelligence offers an end-to-end solution for event-driven scenarios, streaming data, and data logs. Whether dealing with gigabytes or petabytes, all organizational data converges

in the Real-Time Hub. No-code connectors seamlessly link time-based data from various sources, enabling immediate visual insights, geospatial analysis, and trigger-based reactions. Real-Time Intelligence transforms data into a dynamic, actionable resource that drives value across the entire organization, seamlessly aligning with all Fabric offerings.



Real-time hub

The Microsoft Fabric real-time hub acts as a centralized catalog for your organization. It simplifies access, addition, exploration, and data sharing. By broadening data sources, it enhances insights and visual clarity across domains. Crucially, this hub ensures data availability and accessibility, enabling swift decision-making and informed actions. Sharing streaming data from diverse sources unlocks comprehensive business intelligence across your organization.

Exploring data with real-time intelligence

To explore data with Real-Time Intelligence, initially choose a data stream from your organization or connected external/internal sources and then you can utilize Real-Time Intelligence's tools for data exploration and to visualize data patterns, anomalies, and forecasting quantities.

Real-Time dashboards simplify data comprehension, accessible to all via visual tools, Natural Language, and Copilot. You can then turn the insights into actions by setting up Activator alerts to react in real-time.

Note

To learn more about the capabilities of Microsoft Fabric Real-Time Intelligence, see the [Real-Time Intelligence documentation in Microsoft Fabric](#).

5. Explore Apache Spark structured streaming

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/5-spark-streaming>

Explore Apache Spark structured streaming

Completed

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

- Microsoft Fabric
- Azure Databricks

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 Microsoft Fabric 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.

Note

For more information about Delta Lake, see [Lakehouse and Delta Lake tables](#).

6. Exercise: Explore Microsoft Fabric Real-Time Intelligence

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/6-exercise-real-time-intelligence>

Exercise: Explore Microsoft Fabric Real-Time Intelligence

Completed

Now it's your opportunity to explore Microsoft Fabric Real-Time Intelligence in a sample solution to set up and use the main features of Real-Time Intelligence with a sample set of data.

Note

To complete this lab, you will need a Microsoft Fabric account, or if you don't have an account yet, sign up for a [free trial](#).

See [Getting started with Fabric](#) for more information.

Launch the exercise and follow the instructions.

Launch Exercise

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-stream-processing/8-summary>

Summary

Completed

Real-time processing is a common element of enterprise data analytics solutions. Microsoft Azure offers a variety of services that you can use to implement stream processing and real-time analysis.

In this module, you learned how to:

- Compare batch and stream processing
- Describe common elements of streaming data solutions
- Describe features and capabilities of Microsoft Fabric Real-Time Intelligence

- Describe features and capabilities of Spark Structured Streaming on Azure
- Explore real-time analytics in Microsoft Fabric

Next steps

Now that you've learned about stream processing and real-time analytics, consider learning more about data-related workloads on Azure by pursuing a Microsoft certification in [Azure Data Fundamentals](#).

Explore fundamentals of data visualization

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/1-introduction>

Introduction

Completed

Data modeling and visualization is at the heart of business intelligence (BI) workloads supported by large-scale data analytics solutions. Essentially, data visualization powers reporting and decision making that helps organizations succeed.

In this module, you learn about fundamental principles of analytical data modeling and data visualization, using Microsoft Power BI as a platform to explore these principles in action.

2. Describe Power BI tools and workflow

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/2-power-bi>

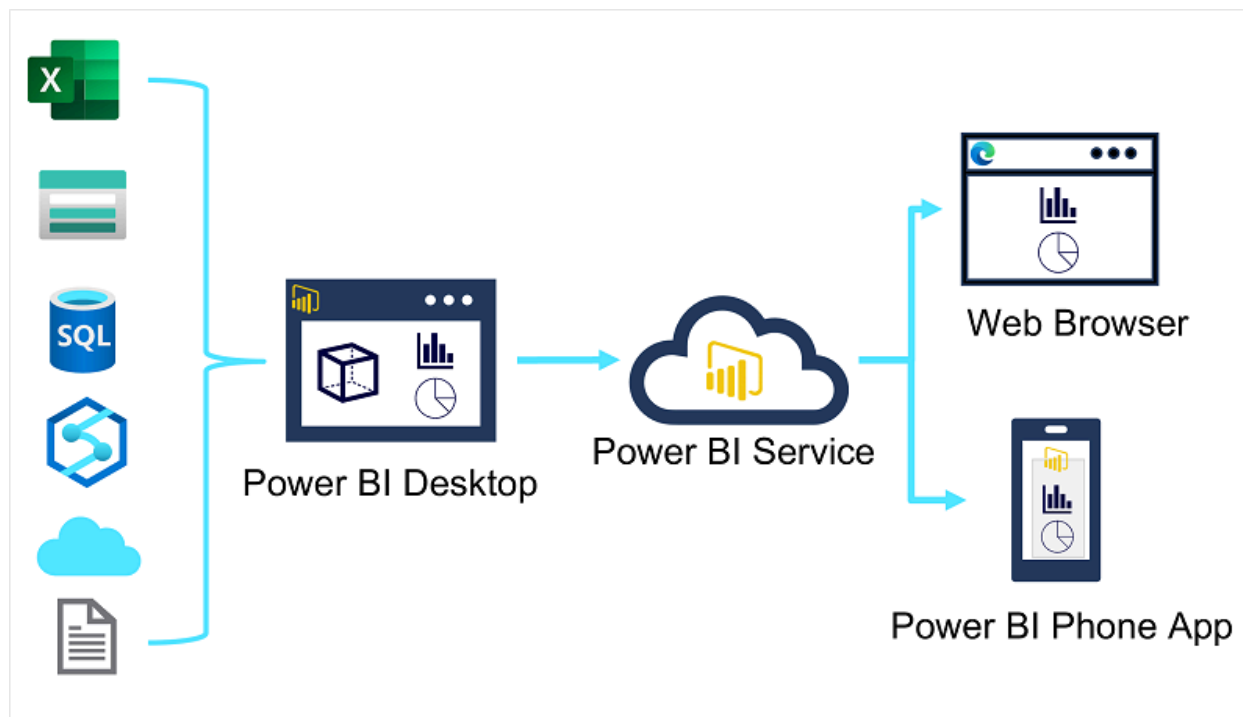
Describe Power BI tools and workflow

Completed

There are many data visualization tools that data analysts can use to explore data and summarize insights visually; including chart support in productivity tools like Microsoft Excel and built-in data visualization widgets in notebooks used to explore data in services such as Microsoft Fabric and Azure Databricks. However, for enterprise-scale business analytics, an integrated solution that can support complex data modeling, interactive reporting, and secure sharing is often required.

Microsoft Power BI

Microsoft Power BI is a suite of tools and services within Microsoft Fabric 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**.

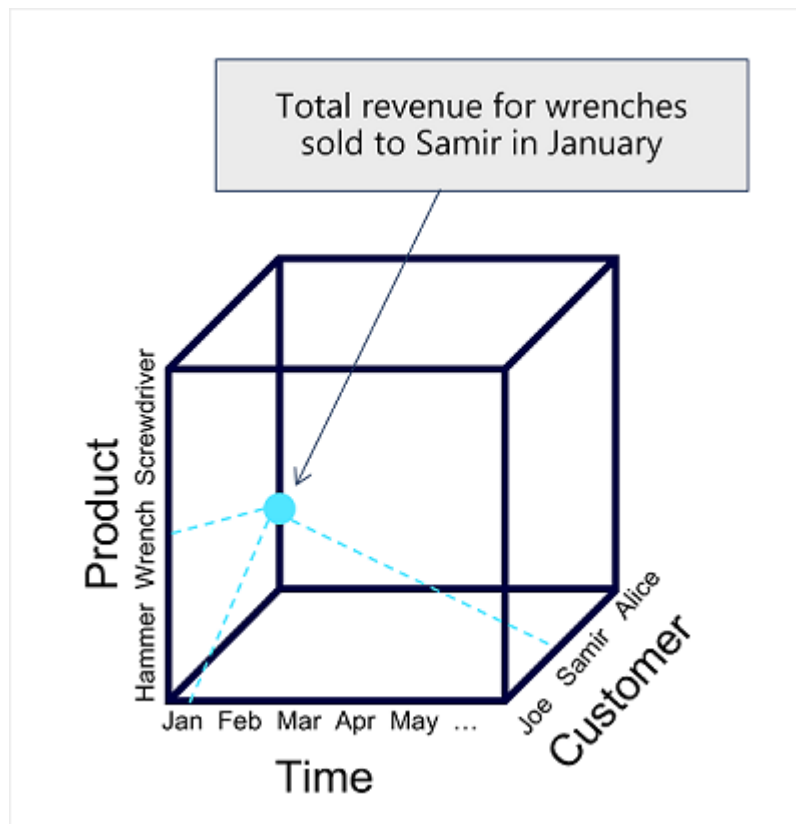
3. Describe core concepts of data modeling

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/3-data-modeling>

Describe core concepts of data modeling

Completed

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 to 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

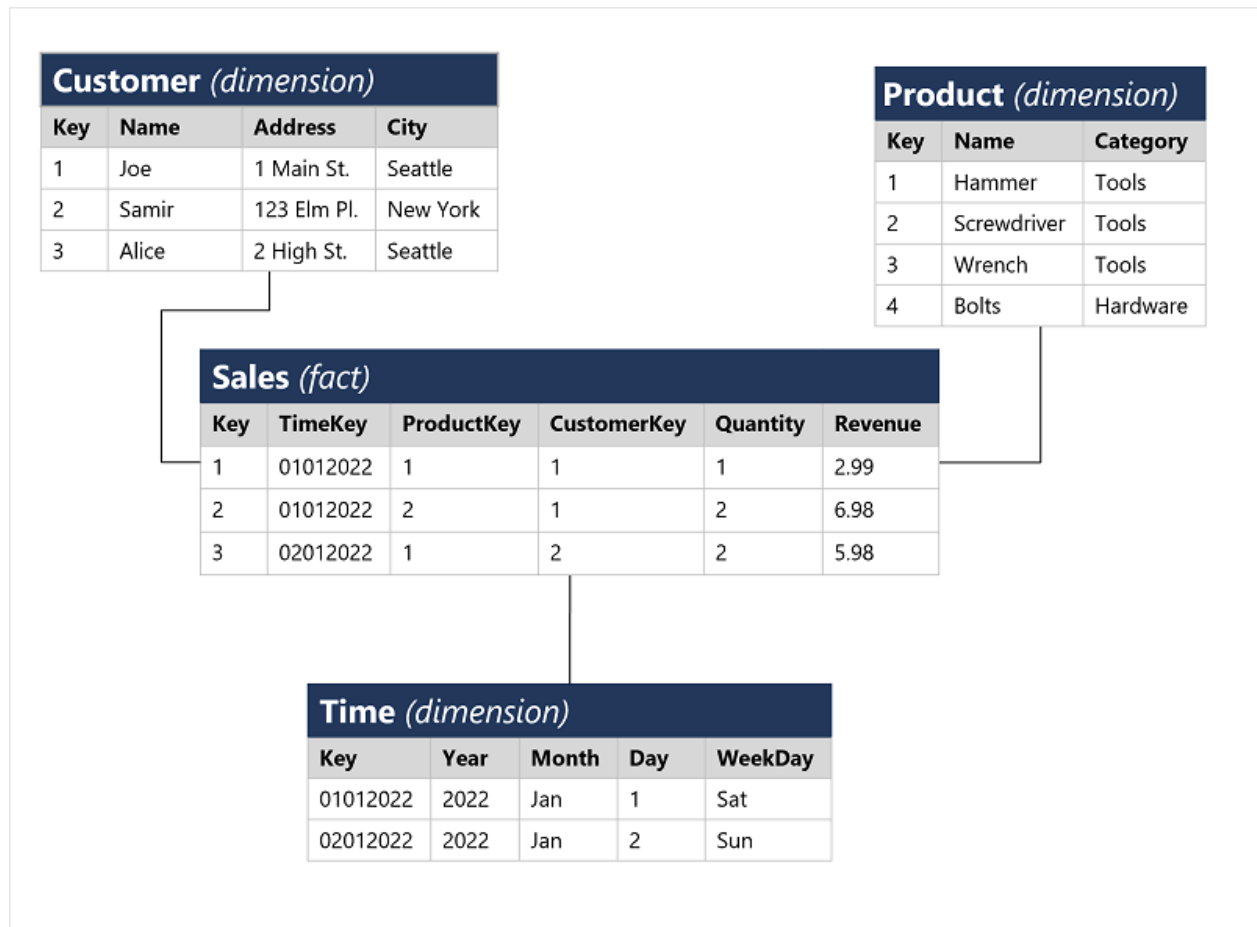
Although we commonly refer to an analytical model as a *cube*, there can be more (or fewer) than three dimensions – it's just not easy for us to visualize more than three!

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.

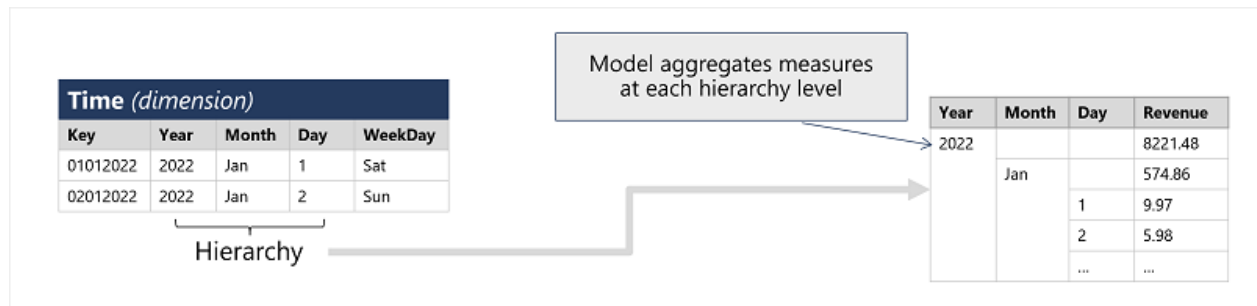


This type of schema, where a fact table is related to one or more dimension tables, is referred to as a star schema (imagine there are five dimensions related to a single fact table – the schema would form a five-pointed star!). You can also define a more complex schema in which dimension tables are 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

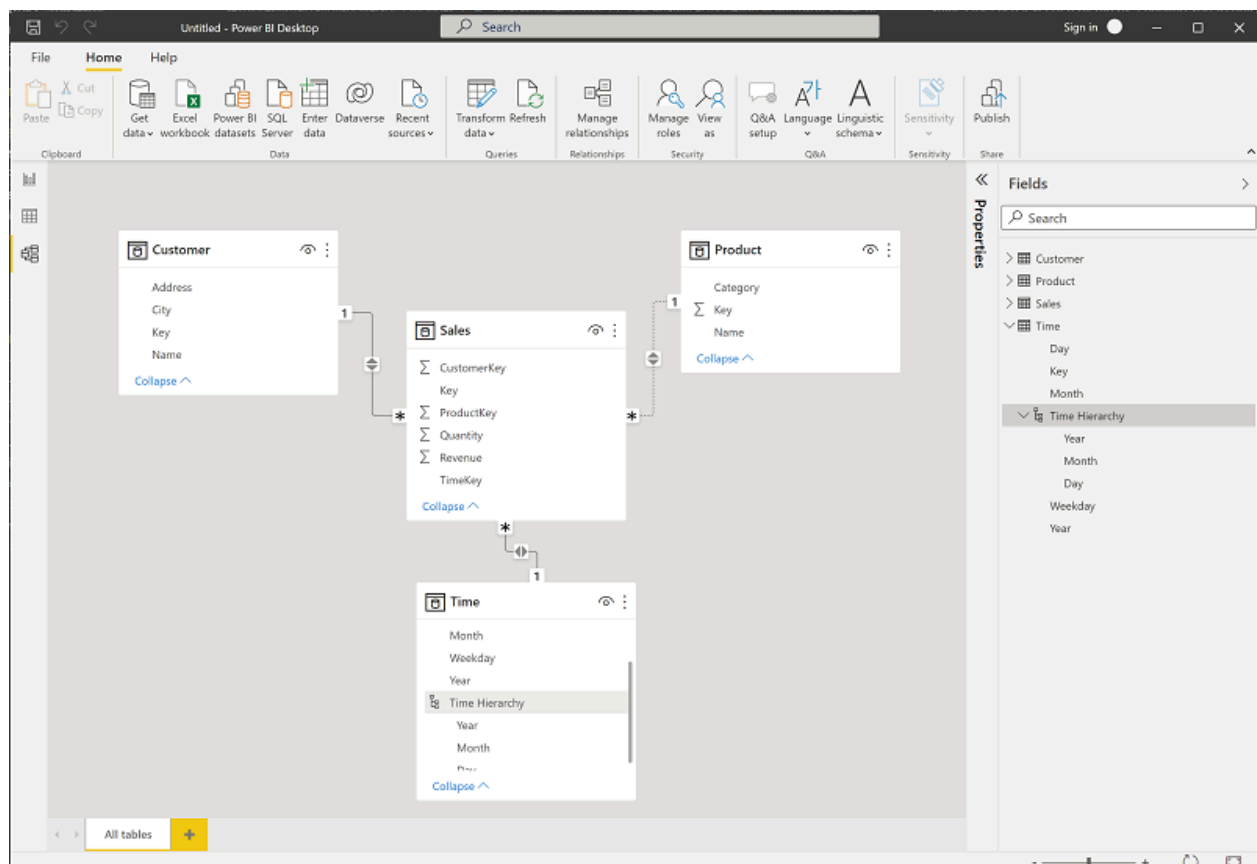
One final thing worth considering about analytical models is the creation of attribute *hierarchies* that enable you to quickly *drill-up* or *drill-down* to find aggregated values at different levels in a

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.



4. Describe considerations for data visualization

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/4-data-visualizations>

Describe considerations for data visualization

Completed

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

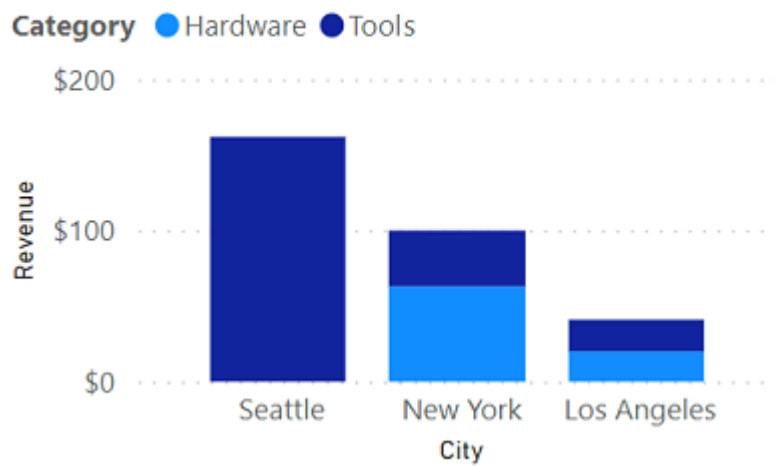
\$302.91

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

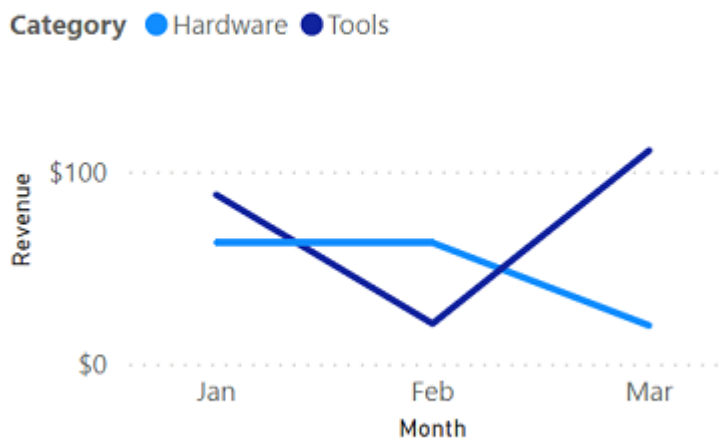
Revenue by City and Category



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

Line charts

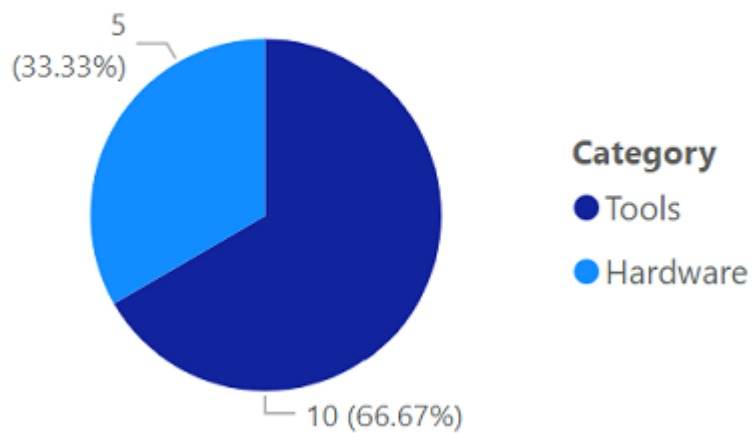
Revenue by Month and Category



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

Pie charts

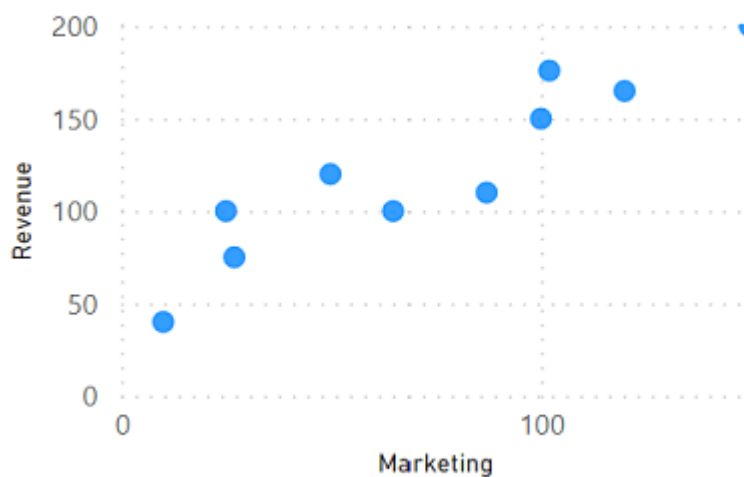
Quantity by Category



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

Scatter plots

Marketing Spend vs Revenue



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

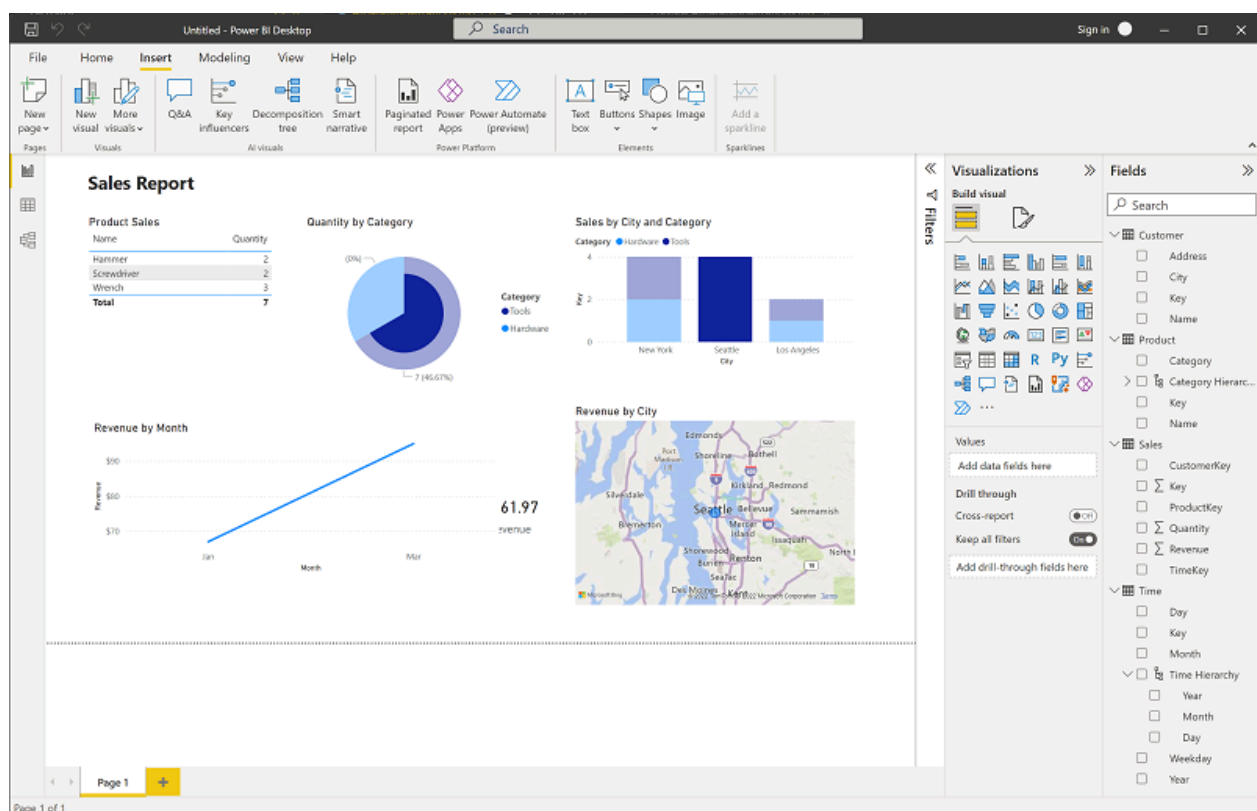
Maps

Revenue by City



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.

5. Exercise - Explore fundamentals of data visualization with Power BI

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/5-exercise-power-bi>

Exercise - Explore fundamentals of data visualization with Power BI

Completed

Now it's your chance to explore data modeling and visualization with Microsoft Power BI.

Note

To complete this exercise, you will need a computer running Microsoft Windows.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/explore-fundamentals-data-visualization/6-knowledge-check>

Module assessment

Completed

7. Summary

Summary

Completed

Data modeling and visualization enables organizations to extract insights from data.

In this module, you learned how to:

- Describe a high-level process for creating reporting solutions with Microsoft Power BI
- Describe core principles of analytical data modeling
- Identify common types of data visualization and their uses
- Create an interactive report with Power BI Desktop

Next steps

Now that you've learned about data modeling and visualization, consider learning more about data-related workloads on Azure by pursuing a Microsoft certification in [Azure Data Fundamentals](#).