



# Ingesting and Preparing Data

## AWS Academy Data Engineering

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

### | Slide number 1

### | Instructor notes

| This module focuses on introductory concepts that are related to ingestion and data wrangling. If your students are data scientists or data analysts who are familiar with wrangling their own data, you might present this module as a brief review to set the context for the module that follows.

|

### | Student notes

Welcome to the Ingesting and Preparing Data module.



| Slide number 2

| Instructor notes

|

| Student notes

This introduction section describes the content of this module.

## Module objectives

This module prepares you to do the following:

- Distinguish between the processes of extract, transform, and load (ETL) and extract, load, and transform (ELT).
- Define data wrangling in the context of ingesting data into a pipeline.
- Describe key tasks in each of these data wrangling steps:
  - Discovery
  - Structuring
  - Cleaning
  - Enriching
  - Validating
  - Publishing



### | Slide number 3

### | Instructor notes

|

### | Student notes

This module builds on the concepts that were presented in the prior modules. In this module, the focus is on the extract, transform, and load (ETL) tasks that are part of ingesting data into your pipeline. The module compares the extract, transform, and load (ETL) and extract, load, and transform (ELT) approaches. It also introduces the steps involved in wrangling data to use in your pipeline.

## Module overview

---

### Presentation sections

- ETL and ELT comparison
- Data wrangling introduction
- Data discovery
- Data structuring
- Data cleaning
- Data enriching
- Data validating
- Data publishing

### Knowledge checks

- Online knowledge check
- Sample exam question



### | Slide number 4

### | Instructor notes

| Each module has an introduction, content sections, and a wrap-up. The wrap-up for this module contains a sample exam question for you to review with the students.

|

### | Student notes

The objectives of this module are presented across multiple sections.

The module wraps up with a sample exam question and an online knowledge check that covers the presented material.

# ETL and ELT comparison

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| **Slide number 5**

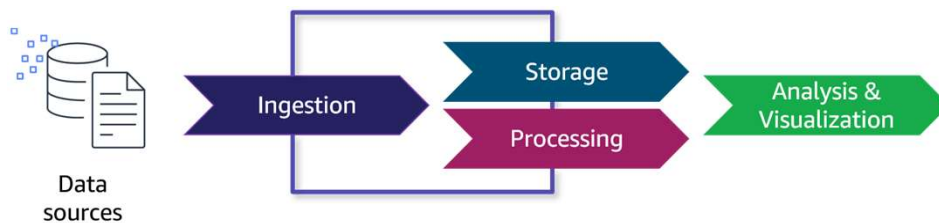
| **Instructor notes**

|

| **Student notes**

This section looks at the high-level processes that are used to ingest data into an analytics pipeline. This section also compares ETL and ELT.

## Ingesting data involves storing and processing data



### | Slide number 6

### | Instructor notes

|

### | Student notes

In the Data-Driven Organizations module, you learned about the layers of a data pipeline. This simplified drawing represents that pipeline and shows each layer as distinct. In practice, ingesting data is about getting data from an external source into your pipeline, where the data can be used. The ingestion process is tied to the tasks of storing and processing data.

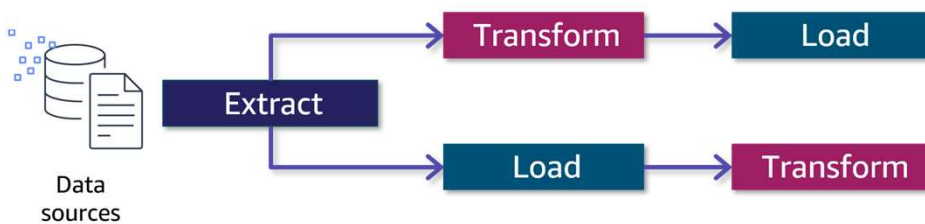
Specifically, the ingestion process incorporates extracting data from a source that isn't already available in your pipeline, loading the data into some type of temporary storage in the pipeline, and processing the data by transforming it before or after the data is loaded into more permanent storage.

The type and scope of transformations, and where they occur across the data pipeline, highly depend on the business need and the nature of the data that is being ingested.

## ETL and ELT

### Extract transform load (ETL)

1. **Extract** structured data.
2. **Transform** the data into a format that matches the destination.
3. **Load** the data into structured storage for defined analytics.



### Extract load transform (ELT)

1. **Extract** unstructured or structured data.
2. **Load** the data into the data lake in the format that is as close to the raw form as possible.
3. **Transform** the data as needed for analytic scenarios.



### | Slide number 7

### | Instructor notes

|

### | Student notes

This slide mirrors the layers that were depicted on the previous slide and compares two methods of ingesting data. You need to extract data from a source and then use a combination of loading the data into storage and transforming or processing the data for its intended purpose.

The traditional method of ingesting and preparing data for analytics was ETL. With ETL, data is *extracted* from its source, *transformed* into a structured format that is ready to be used in analytics applications, and *loaded* into structured storage, such as a data warehouse. Data scientists or analysts query the data warehouse and might perform additional transformations and processing as part of their analyses.

As described in the Design Principles and Patterns for Data Pipelines module, data stores evolved from handling only structured data to include options for handling high volumes of unstructured data. Specifically, there has been a move from structured data in data warehouses towards unstructured and semistructured data in data lakes. Tools and methods for ingesting data into a pipeline evolved along the same lines. ETL techniques

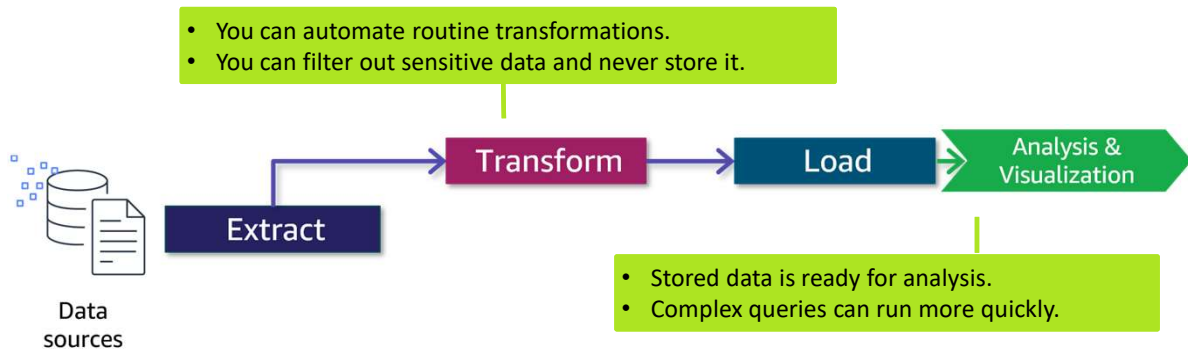
have continued to evolve and are still a valuable part of data architectures for structured data.

However, the trend towards collecting high volumes of unstructured and semistructured data and the desire to make that data available for a variety of uses led to a different approach for data ingestion: ELT. With an ELT approach, data is extracted from its source and transformed only enough to store it in a consumable but relatively raw state in a data lake.

Transformations that might be needed for specific uses are done later, as consumers access data and build analytics solutions with it.



## Benefits of extract transform load (ETL)



### | Slide number 8

### | Instructor notes

|

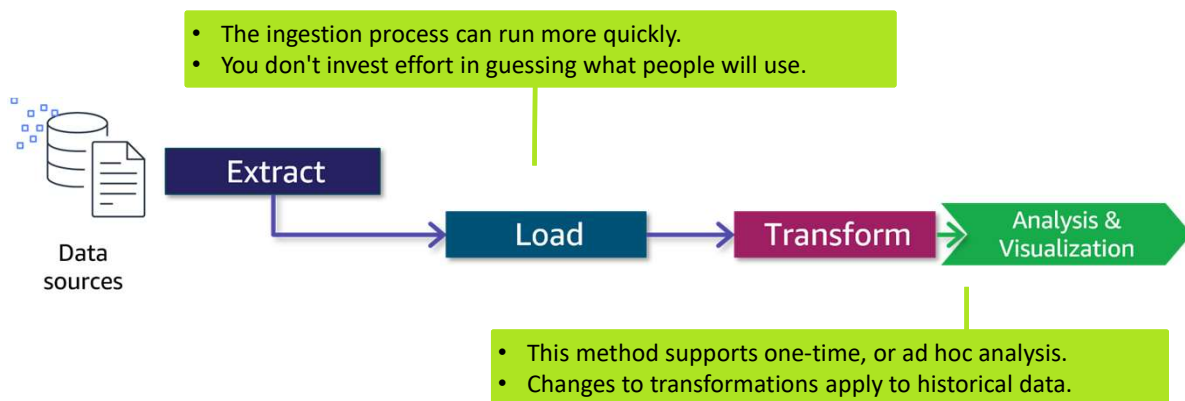
### | Student notes

In addition to the general guidance that ETL works well for structured data that is destined for a data warehouse, and ELT works best for unstructured data that is destined for a data lake, there are other reasons to choose one or the other. This slide highlights a few considerations.

ETL stores data that is ready to be analyzed, so it can save time for an analyst. If analysts routinely perform the same transformations on data that they analyze, it might be more efficient to incorporate those transformations into the ingestion process. So, with ETL, you trade longer transformation processing time before the data is available for faster access when you are ready to query it.

Another advantage of performing transformations up front is that you can filter out personally identifiable information (PII) or other sensitive data that could create a compliance risk. If you never store the sensitive data, you reduce the risk without the need to evaluate additional security measures.

## Benefits of extract load transform (ELT)



### | Slide number 9

### | Instructor notes

|

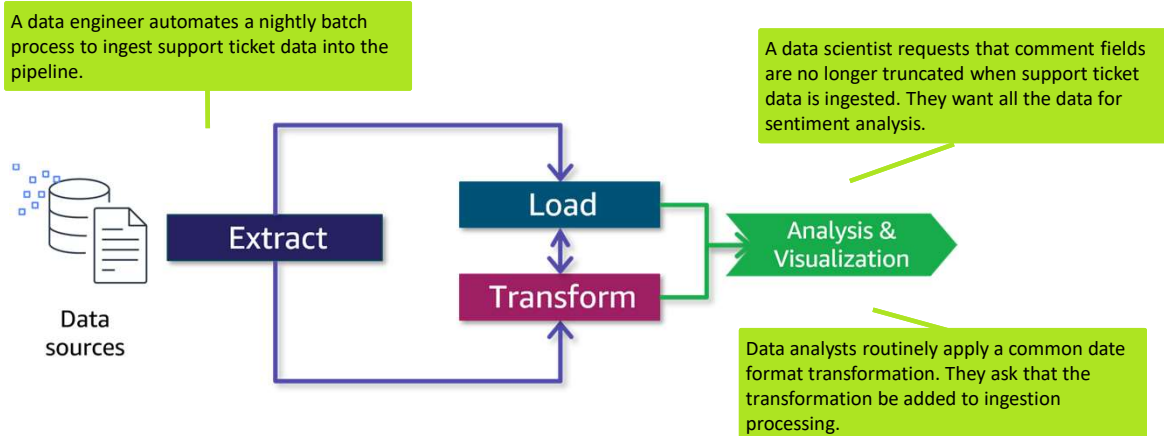
### | Student notes

On the other hand, ELT provides more flexibility to create new queries because analysts have access to greater amounts of raw data.

With ELT, the process to get the raw data into your pipeline can be quicker because you aren't required to transform the data as much before making it available in the data lake. So, with ELT, your trade-off is that you get the raw data available more quickly, but you spend more time to transform the data when you're ready to work with it.

Another advantage of storing the raw data and only transforming it downstream is that changes to the transformation (for example, including additional fields or formatting data differently) would be run against all the historical data that was loaded when the transformation is updated. If the transformations occur before you load the data, your changes can apply to new data that is ingested, but it might be impossible or difficult to apply them to historical data.

## The ingestion process evolves as the pipeline is used



### | Slide number 10

#### | Instructor notes

| Depending on the makeup of the class, it might be worth spending a bit more time talking with the students about their primary role and how they will interact with the pipeline and the other roles.

|

#### | Student notes

As other modules also note, how you design your ingestion process really depends on the business problem that you're addressing and the trade-offs that make the most sense for your use case. The design is likely to evolve based on both the builder's experience and the pipeline users' experience.

In modern architectures, ETL and ELT approaches are blurred. Each role that works with the pipeline might perform a different part of the ingestion and transformation process by using different tools and access. For example, the data engineer might use an ingestion tool that handles common cleaning or formatting transformations before loading a dataset into your data lake. Additional transformations might occur only when the data is pulled for a particular analytics use case. The data analyst might apply a set of transformations on the ingested data to prepare it for a specific report. A data scientist might perform additional

discovery and transformation on the ingested data to investigate hypotheses about relationships within the data.

As patterns of use and understanding of the dataset evolve, you might revisit where and how transformations occur. For example, if many analysts perform the same recurring transformation to work with a dataset, adding those transformations into the processing that occurs before the data is loaded might improve performance or reduce costs.

Alternatively, you might recognize additional needs that you could serve by reducing the amount of transformation that is performed before loading the data. For example, you might truncate a large field to save costs, but a data scientist determines that there's value in the truncated field and asks you to remove that transformation.

As the data engineer, you want to monitor how the pipeline performs, how it is accessed and used, and how you might modify your processes to optimize the value.

## Key takeaways: ETL and ELT comparison



- Ingestion involves pulling data into the pipeline and applying transformations on the data.
- Ingestion by using an ETL flow works well with structured data that is destined for a data warehouse or other structured store.
- Ingestion by using an ELT flow works well for unstructured data that is destined for a data lake for a variety of use cases.
- ETL and ELT processing within a pipeline should evolve to optimize its value.

**| Slide number 11**

**| Instructor notes**

|

**| Student notes**

Here are a few key points to summarize this section.

Ingestion pulls data into the storage layer and applies transformations to it.

Both ETL and ELT are common approaches to ingest data, with ETL being more geared to structured data and ELT being more for unstructured data.

# Data wrangling introduction

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| **Slide number 12**

| **Instructor notes**

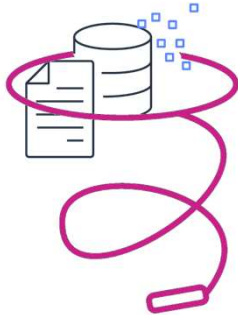
|

| **Student notes**

This section looks at the high-level processes to prepare data as it is ingested into an analytics pipeline.

## Data wrangling

---



Transforming large amounts of unstructured or structured raw data from multiple sources with different schemas into a meaningful set of data that has value for downstream processes or users



### | Slide number 13

### | Instructor notes

|

### | Student notes

Regardless of when data is transformed, the data engineer needs to develop methods to extract and combine disparate data sources into a useful dataset. Data analysts and data scientists need to explore data, test hypotheses, and determine how to enrich or modify their dataset to meet their business needs.

The term *data wrangling* reflects the complexity and messiness of transforming large amounts of unstructured data or sets of structured data with multiple schemas across multiple sources into a meaningful set of data that has value for downstream processes or users.

The content in this module takes the perspective of the data engineer. However, data engineers, data scientists, and data analysts might each perform different types of wrangling based on what they need from the data and the state of data that they start from—for example, from a data warehouse compared to a data lake. For ML applications in particular, a data scientist might perform lots of iterative wrangling to get a dataset that is suitable for building and training an ML model.

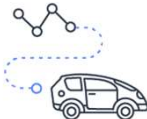
## Data wrangling addresses managing data variety



On-premises databases or file stores



Public datasets



Events, Internet of Things (IoT) devices, sensors

A healthcare company has structured patient data.

They want to combine patient data with semistructured data from a public dataset.

They also want to include unstructured data from a mobile app that logs real-time events.



### | Slide number 14

### | Instructor notes

| This is a good opportunity to prompt students about other examples that they have for each data source type. You could also discuss the type of elements or attributes that might be pulled from each type of data source to create a combined data source.

|

### | Student notes

The Elements of Data module introduced this slide to illustrate how an organization might combine different data types to create a personalized customer experience.

The organization can query data from the on-premises database to find patients who are overdue for a visit.

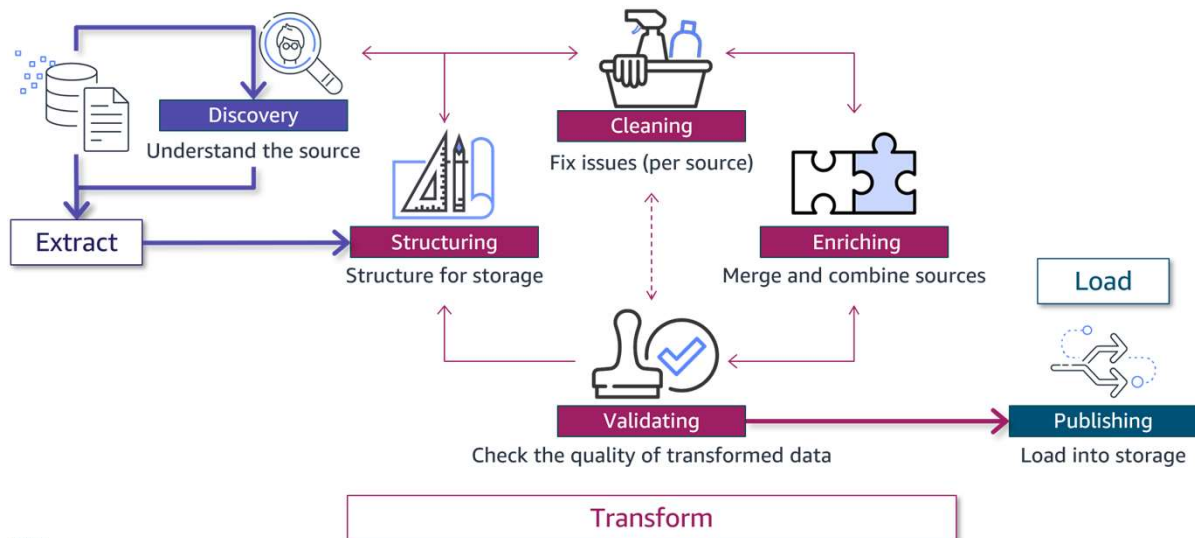
By using public healthcare datasets, the company might combine public health data that identifies demographic risk factors for heart attacks with customer data to personalize information for a patient who shares the risk factors. If patients had a mobile app that sent real-time heart monitoring data at some interval, the data could be evaluated for anomalies that coincide with known risks. Then, the patient or their doctor could be alerted.



You can see in this example how pulling data from many sources can provide different levels of personalization and prediction.

As the data engineer, you need to figure out how to put all this data together and make it available in your pipeline. With the data available, data scientists can build the machine learning (ML) models to provide the personalized predictions that are desired.

## Data wrangling steps



| Slide number 15

| Instructor notes

|

| Student notes

This slide presents the steps of data wrangling as distinct and part of ingesting data into the pipeline. In practice, there might not be clear delineations between the steps, there might be iterations within and across steps, and individual steps might not occur as part of ingesting a data source into the pipeline. Each step might be more or less complex, depending on the goal of the specific ingestion that you are working on.

The overall ingestion and data wrangling process could be as simple as opening a source file in Microsoft Excel, manipulating columns and individual values in the spreadsheet, manually checking the validity of the data, and then saving it as a .csv file and uploading it to an Amazon Simple Storage Service (Amazon S3) bucket. At the other end of the spectrum, complex transformations might be needed to support high volumes of incoming data for an ML application. The data engineer and data scientist might use a variety of scripts and tools to wrangle the data for this use case.

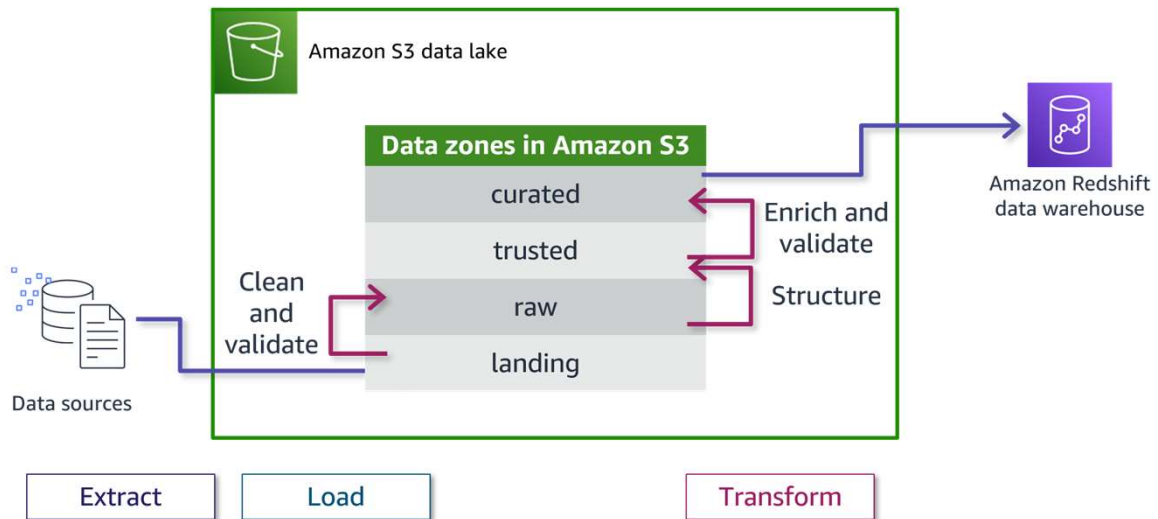
As illustrated on this slide, the steps of data wrangling generally coincide with the steps of ETL processing. However, it's worth making a distinction here between ETL processing and data wrangling.

Traditionally, ETL processing has been more of the data engineer's domain. ETL processing encompasses scripting and automating transformation steps with batch jobs that run on infrastructure that is controlled by IT. Analysts and data scientists worked only with data that was in a defined, consumable state.

Data wrangling is more associated with an ELT flow where business users or data scientists transform datasets that are available to them within the pipeline storage layer but that haven't been refined for a specific use case.

The continuing trend in cloud tools and services is to abstract infrastructure and coding tasks to provide more types of users with direct access to extract and transform data. A data engineer must be able to build complete analytics pipelines and transform data as needed for a use case. But it's also important to stay aware of tool updates that can provide their team's analysts and data scientists with more autonomy to work with available data.

## Example: ELT and data wrangling in a modern data architecture



### | Slide number 16

### | Instructor notes

|

### | Student notes

There is overlap between the ingestion, storage, and processing layer concepts that you have learned so far. It's important to remember that the defined layers help to describe the activities that are performed and tools that are used as data moves through and is transformed in the pipeline. But pipeline activities are iterative within and across layers. Each pipeline's architecture might look quite different depending on the business problem being addressed, type of data being ingested, and skills and expertise of the team that supports the pipeline and accesses the data.

This diagram is derived from the modern data architecture in the Design Principles and Patterns for Data Pipelines module. The diagram illustrates an ELT flow and highlights how the data wrangling steps might occur. In this reference architecture, data is ingested into a storage layer that includes an Amazon S3 data lake and an Amazon Redshift data warehouse. Within the data lake, S3 buckets are used as named zones to store data in different states.

The landing zone bucket serves as temporary storage from which you perform just enough cleaning and validating to permanently save the ingested data into the raw zone bucket. You can then use data in the raw zone for additional use cases. This data might be structured, semistructured, or unstructured. At this stage, the initial extract and load steps are complete.

In this example, the transform part of ELT occurs when data in the raw zone of the data lake is transformed into a highly structured state that the Amazon Redshift data warehouse can use.

The data engineer or data scientist would need to structure, enrich, and validate the data before the warehouse can use it. Data is stored in different zones as these steps are performed.

Data structuring prepares the data for the trusted zone. Preparation for this zone might include activities such as conforming to the schema and partitioning specifications for the dataset based on data warehouse tables. Then, data enriching is performed on data in the trusted zone, modeling the data so that it can be joined with other datasets and stored in the curated layer. Data validation would also occur as part of these transformations before storing data into the next zone.

Datasets in the curated layer are ready for the Amazon Redshift data warehouse to ingest to make them available for low-latency access and complex SQL querying.

Depending on the use case, additional discovery might occur before the structuring, enriching, and validating steps take place. Additional cleaning might be needed as well. The process is often iterative.

If the process to load data into the data warehouse had been accomplished using a traditional ETL flow, the same wrangling tasks would still be needed. However, all the transformations (cleaning, structuring, enriching, and validating) would be performed on the data in a temporary staging location before the data is loaded directly into the data warehouse. Users would not have access to the data until processing was complete, and the raw data would not be available.

The next sections look at each of the steps in the wrangling process in a bit more detail.

## Key takeaways: Data wrangling introduction



- Data wrangling describes the multi-step process of transforming large amounts of unstructured data or sets of structured data from multiple sources for an analytics use case.
- Data wrangling is especially important for data scientists when building ML models.
- Data wrangling steps might overlap, be iterative, or not occur at all in some ingestion processes.
- Data wrangling steps include discovery, structuring, cleaning, enriching, validating, and publishing.

**| Slide number 17**

**| Instructor notes**

|

**| Student notes**

Here are a few key points to summarize this section.

Data wrangling is a multi-step process, with steps that overlap and iterate differently depending on the data source and the use case.

Data wrangling steps include discovery, structuring, cleaning, enriching, validating, and publishing.



### **| Slide number 18**

### **| Instructor notes**

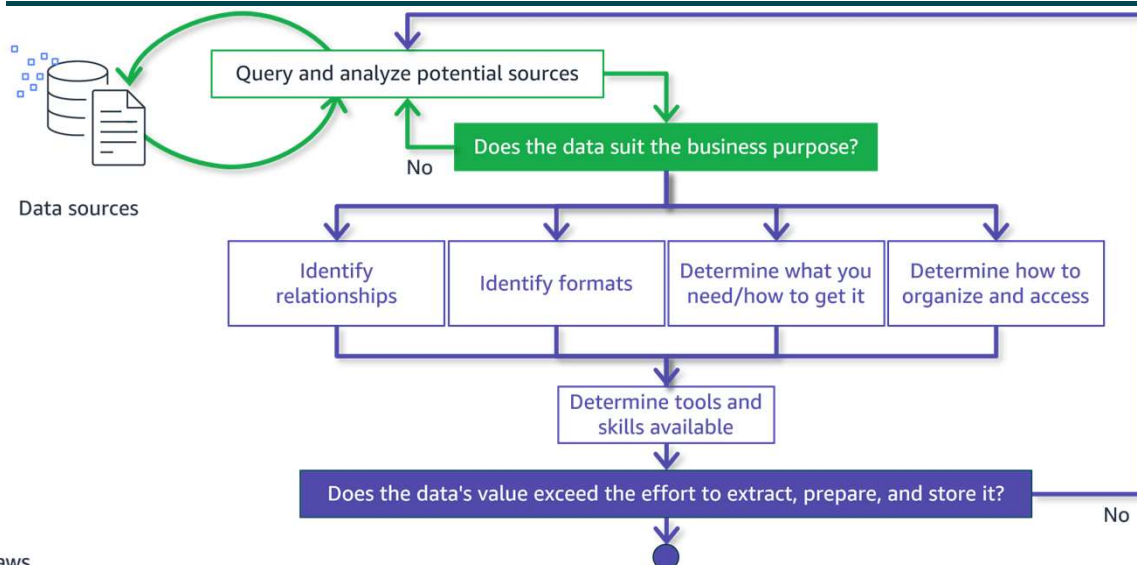
| The sections that follow for each data wrangling step are based on an example scenario that is presented in this section. The scenario is then worked with example structures and data through the publishing step. The slides are designed to provide all the information that is required to lecture to the slides. However, you could also do this as a live demo and discussion and talk about each step. You could even have students perform the wrangling themselves by using sample .json files. The .json files that match the example scenario are included in the instructor files for the course.

|

### **| Student notes**

This section looks at the data discovery step of the data wrangling process.

## Decisions and tasks in discovery



### | Slide number 19

#### | Instructor notes

| You might want to stress or explore with students how this discovery flow presents the data engineer's perspective, but each role who is involved with the data pipeline has their own discovery tasks. Depending on the roles and backgrounds of your students, you could also lead a discussion to help students see the perspective of each role.

|

#### | Student notes

The data discovery step is iterative, and different roles (analyst, data scientist, data engineer) might look for different things at this step. As a data engineer, you need to find sources that you think might be useful, query them, and analyze the raw data to decide whether the sources have value for your business purpose. An analyst or data scientist might tell you about a source that they want to use. What you learn from your initial results might lead you to refine your queries and analyze the updated results before you decide how to use each source.

Key tasks of this step include the following:

- Decide if a source seems to serve your business purpose. If it does, then you want to dig



deeper into the data to do the following:

- Identify what relationships exist within and between the raw data sources that you expect to use.
- Identify the data formats (for example, CSV, Parquet, Optimized Row Columnar [ORC]).
- Determine the range of data that you want from each source (for example, a time period or range of attribute values). Determine how to get the desired range out of the source and into your pipeline.
- Determine how you should organize the data to make it easy—but secure—to access and cost-effective to ingest and store. For example, what folder structure is appropriate? What's the best way to manage file size? How should you partition your target database table? You might need to choose a partition key that optimizes performance for your use case. Or you might need to establish folder structures in Amazon S3 that provide granular access to parts of the dataset to different users. For example, you might need to create Amazon S3 prefixes (or folders) for a particular segment of data so that you can grant access to each folder only to the team members that are authorized to access that segment. Your decisions about organization should also consider the structure that you want to use as part of the data wrangling process. For example, you might use the zone approach, which was illustrated earlier, to segment data that comes into the data lake into "raw," "landing," and "trusted" zones.
- Based on your analysis of the data relationships, formats, and organizational requirements, determine what tools you need. Also, determine what you have available to store, secure, transform, and query the type of data in each source. Consider whether you have the skills and resources that are needed to extract data from the source and prepare it for use.
- Based on these discovery activities, you might decide to work with the source. Or, you might determine that the effort and cost to extract and prepare the data for the intended business purpose doesn't provide enough value. If you determine that it's not worth the effort, you might start the whole process again and seek additional sources or look for ways to reduce the volume of data to be ingested, amount and type of storage that are required, or complexity of preparing the data for use.

## Data discovery example scenario

---

- A company with a software as a service (SaaS) product recently acquired a startup company.
- The startup company's customer support team uses a different ticketing system than the company that acquired them.
- A data engineer has been asked by a data analyst to ingest support ticket data from the two products to perform an initial exploration of relationships between support experiences, ticket volumes, and contract renewals.
- The analyst wants to know the number and types of technical issues that have been worked on or closed for each customer in 2020. They also want each regional sales team to have access to only their customers.
- The existing sales analytics pipeline is built on AWS services and currently houses customer data in an Amazon Redshift data warehouse. However, it doesn't include customer support data.



### | Slide number 20

#### | Instructor notes

| If you go through this scenario as a discussion, you might pause here and capture feedback from students about what the most important characteristics are to ingest this data, based on what the students have learned. Key callouts include that there are two different systems, and both are still being used; that this request is intended to be *exploratory* in nature; the need to filter the data by year and type of ticket; the access requirements; and the existing pipeline that serves the sales organization.

|

#### | Student notes

The next few sections will use the simplified ingestion example that is described on this slide to walk through each step in the wrangling process.

## Data discovery example: Query data sources

Ticket table in customer support system (supp1)

ticket_id	requestor_id	submitter	assignee	group	subject	status	priority	ticket_type	create_date	updated_date	solved_date
							High				
							Low				

Ticket table in the acquired customer support system (supp2)

issue_id	cust_num	description	status	priority	create_date	updated_date	closed_date
				1			
				3			

Customer table in the data warehouse

customer_id	cust_name	primary_poc	status	sales_group



### | Slide number 21

#### | Instructor notes

| You might pause here before showing the next slide and have a discussion about what these example tables tell you about what you need to do to meet the request. For example, what are the data relationships that are relevant to the stated requirement? What would you query in the two customer support tables (supp1 and supp2) to get the subset of data that you need? What information might you need to be able to organize the data based on the access requirements that were stated in the example?

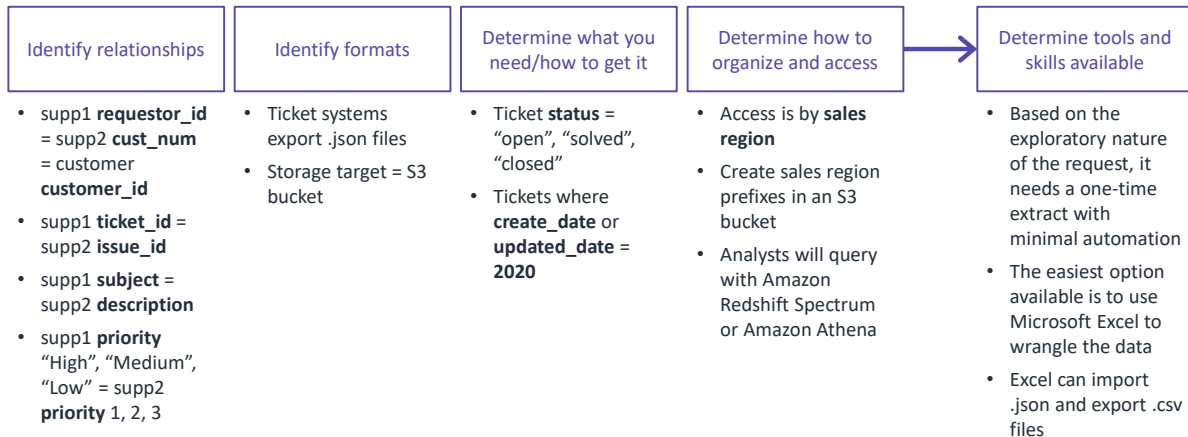
|

#### | Student notes

For the initial discovery step, the data engineer might look at the fields and data that are available in the source data for customer support tickets. The engineer might also determine which information that is stored in the data warehouse is related to their business need. Based on this, they would determine how to get the desired data out of the source and into their pipeline.

Looking at these table structures, what do you think is relevant to the request?

## Data discovery tasks: Example scenario



### | Slide number 22

### | Instructor notes

| You might choose to use this slide as an extension of the discussion exercise. To support the option to discuss by heading, the text boxes are animated reveals (all at once).

|

### | Student notes

This slide summarizes the discovery activities for the example scenario. The data engineer uses this information to decide how to extract the data and structure it for ingestion.

## Key takeaways: Data discovery



- Data discovery is iterative and might have a different focus depending on your role.
- Tasks performed during the data discovery step include the following:
  - Identifying relationships and formats
  - Determining how to filter the incoming data and how to organize it in your target storage
- The two primary outcomes of the data discovery phase are identification of the tools and resources required and a decision to move forward.

**| Slide number 23**

**| Instructor notes**

|

**| Student notes**

Here are a few key points to summarize this section.

Data discovery is an iterative process that each role who is involved in the business need should perform.

During data discovery, the data engineer should determine the relationships between sources, how to filter the sources, and how to organize the data in the target storage.

The discovery phase helps you to identify the tools and resources that you will need to move forward.

# Data structuring

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 24

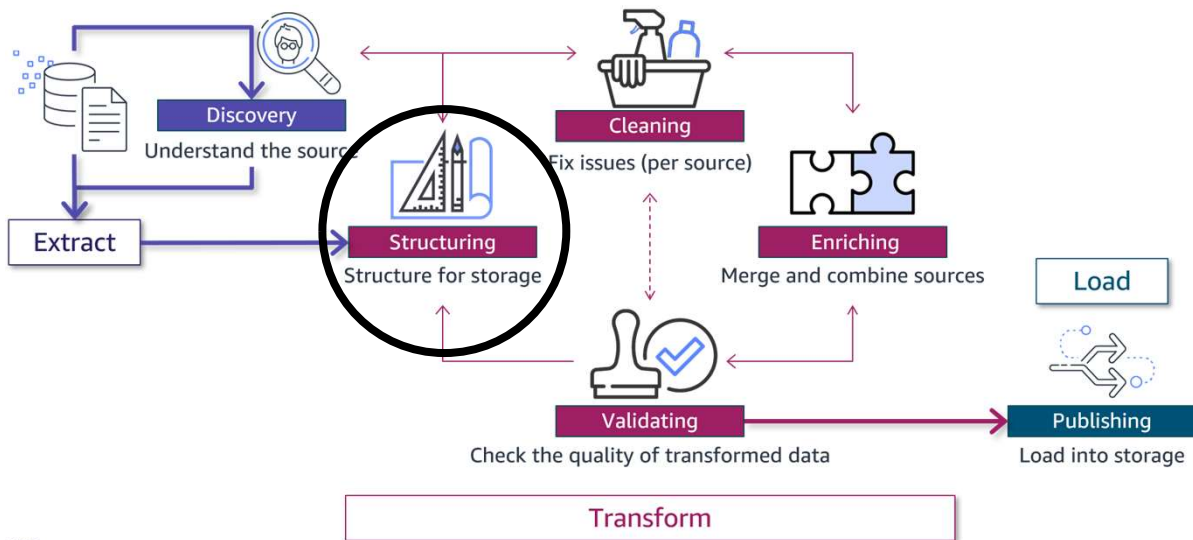
| Instructor notes

|

| Student notes

This section looks at the data structuring step of the data wrangling process.

## The data structuring step within data wrangling



| Slide number 25

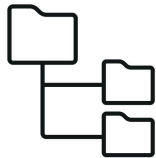
| Instructor notes

|

| Student notes

The structuring step is the second defined step within the data wrangling process, following discovery. Data structuring is about mapping data from the source file or files into a format that supports combining it with other data and storing it for its intended purpose. A key goal of this step is to optimize the structure of the raw dataset to minimize costs and maximize the performance of your processing pipeline.

## Decisions and tasks in data structuring



### Organize storage

- Control access.
- Create the folder structure.
- Establish partitions.

```
{  
  "ticket_id": 900864,  
  "requestor_id": 1800445,  
}
```

### Parse the source file

- Convert strings and patterns into fields or attributes.

SALES	
CUST	

### Map fields

- Match source fields to target fields.



### Manage file size

- Split or merge files.
- Compress files.



### | Slide number 26

### | Instructor notes

|

### | Student notes

Key tasks in data structuring include the following:

- Organize storage.
- Parse the source file or files.
- Map fields.
- Manage file size.

Data structuring involves implementing the decisions that you made during discovery and structuring your source file.

First, you need to create any infrastructure that is needed to organize the incoming data and set up appropriate access controls. For example, create IAM access policies for each S3 bucket that you create to segment the files, or set up access control on tables in your data warehouse.

By parsing the source file, you can convert defined strings or patterns in the source into formats that can store the strings into structured tables or categorize data in the data lake.



Your parsing approach might be straightforward if you are using structured or semistructured source files. But for unstructured sources, you might need to explore the data to find the relevant patterns—for example, pulling out error codes from within a log file.

In addition to parsing the file into relevant fields or attributes, you need to map fields in the source file to the appropriate fields in your target storage. For example, you might map a field called `requestor_id` in the source file to a field called `customer_id` field in your target.

Another decision that you need to make as part of data structuring is how you will optimize file size for storage and retrieval. This might mean splitting up large files or combining small files. You would also probably apply a compression mechanism before saving your file to the data lake.

Your data structuring process might be simple. For example, you could import a file into Excel, organize its columns, and delete the columns that you don't need. You could update column names to match fields in your storage target, export the data as a .csv file, compress it, and upload it to a designated S3 bucket and folder.

For high-volume, high-velocity, or complex data ingestion, you would typically use an integration tool, scripts, or batch jobs to automatically parse, map, and apply the desired structure to your incoming data and move it to the target location.

## Data structuring example: Parse supp2 file and map fields

Import .json and let Excel parse the file.

Update column headings to map to supp1 and data warehouse field names.

A	B	C	D	E	F	G	H
issue_id ticket_id	cust_num customer_id	description subject	status	priority	create_date	updated_date	closed_date solved_date
900865		Error message 808 <*&#	Closed	2	6/26/20 2:45:29	6/27/20 21:31:48	6/27/20 23:30:09
900866	2020411	Nullpointer exception on save	Open	1	4/2/20 12:15	4/5/20 11:15	
900867	2022010	Question about grades report	Open	1	4/6/20 10:56		



### | Slide number 27

### | Instructor notes

|

### | Student notes

For the scenario that was described previously, the structuring step includes exporting a .json file from the customer support ticket system, loading the .json file into Excel, and letting Excel parse the file. For the mapping step for the supp2 data, the data engineer would modify the cust\_num field to match the customer\_id field in the data warehouse. The engineer would modify the issue\_id, description, and closed\_date columns to match what's in supp1.

For this example, you would perform additional data wrangling steps before compressing the file for upload to the S3 bucket.

## Key takeaways: Data structuring



- Data structuring is about mapping raw data from the source file or files into a format that supports combining it and storing it with other data.
- Data structuring includes organizing storage and access, parsing the source file, and mapping the source file to the target.
- Data structuring also includes strategies and methods to optimize file size, such as splitting or compressing files.

### | Slide number 28

### | Instructor notes

|

### | Student notes

Here are a few key points to summarize this section.

Data structuring is about mapping raw data from the source file or files into a format that supports storing it and combining it with other data.

During the structuring step, you need to parse the file, map source fields to target fields, and establish the structures that are necessary in the target to organize the dataset.

Your structuring decisions should also include the best ways to optimize file size.

# Data cleaning

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 29

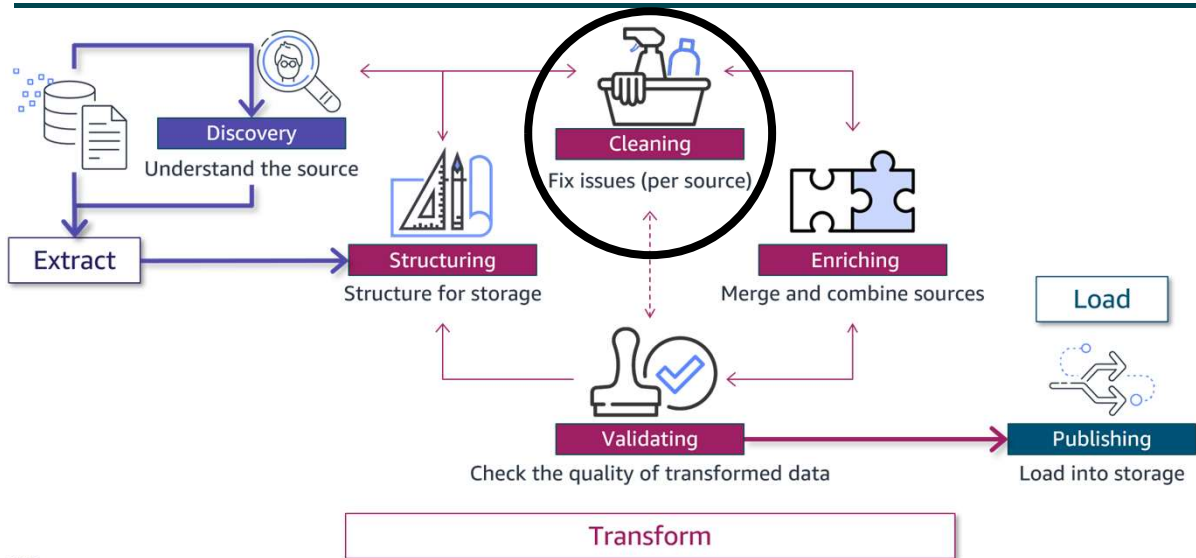
| Instructor notes

|

| Student notes

This section looks at the data cleaning step of the data wrangling process.

## The cleaning step within data wrangling



| Slide number 30


| Instructor notes

|

| Student notes

The cleaning step is the third defined step within the data wrangling process, following structuring. In the data cleaning step, you start to make the raw data ready for use in your pipeline. Usually, at this step, cleaning is per source, based on the characteristics of that source.


## Decisions and tasks in cleaning



001	42	4/2	abc
002		4/5	a
003	78	5/5	
004	14	5/8	def
005	4200	5/9	fed

### Remove data


- Remove unwanted columns.
- Remove duplicate values.
- Remove "garbage" values.



001	42		4/2
002	0		4/5
003	78		5/5
004	14		5/8
005	4200		5/9

### Fill missing data


- Fill null columns.
- Fill missing values for mandatory fields.



001	42	22-04-02
002	0	22-04-05
003	78	22-05-05
004	14	22-05-08
005	4200	22-05-09

### Check data types

- Validate or modify data types.



001	42	22-04-02
002	0	22-04-05
003	78	22-05-05
004	14	22-05-08
005	42	22-05-09

### Fix outliers

- Identify outliers, and either remove or fix data as appropriate.



| Slide number 31

| Instructor notes

|

| Student notes

Data cleaning includes tasks such as the following:

- Remove unwanted data.
- Fill in missing data values.
- Validate or modify data types.
- Fix outliers.

Depending on who does the cleaning, the methods and results might be different. For example, a data engineer who is only responsible for successfully loading the data into the pipeline might make sure that blank fields are replaced with values that fit the data type. However, a data analyst might replace blanks by adding corrected values.

Let's look at each task in a bit more detail.

Even when you intend to load data into your data lake for a variety of use cases, there are elements that you don't want to store. For example, columns that include PII or fields that aren't relevant for analysis. You would also want to remove duplicate values that are

present in the source file. You might need to remove values that don't represent the intended contents for a field or attribute. For example, a field with corrupted data or unwanted characters that won't conform to the target's requirements or that would prevent the row from being joined with other related data successfully.

Data cleaning also addresses missing data. For example, fields that are blank might need to be converted to zeroes if the field should represent a number. You might also need to fill in data for mandatory fields so that the dataset will load into a target structure. For example, if the target table expects a postal code for each record, but your source has some records that are missing this field, the cleaning process would need to put a generic value in to replace the missing value.

Another cleaning step that helps to ensure that the data being ingested can be stored and referenced as intended is to validate that data types in the source file match the target and to modify those that don't. For example, if the source file stores dates as a text string and the target requires a specifically formatted date field, you would need to convert the strings to properly formatted date fields.

One cleaning step that might require a bit more analysis is to identify outliers and either remove them from the dataset or fix them at the source. A data engineer might run a script that identifies values that are far outside the range of other data values and remove those rows. A data analyst or data scientist might dig deeper to determine if the outlier is an error that should be fixed at the source or data that should be discarded. In the example presented here, the value of 4200 in row 5 was an outlier. The analyst replaced it with 42.

## Data cleaning example: Remove and fix data

Replace blanks in the required **customer\_id** field with a placeholder.

Remove invalid characters from the **subject** field.

Replace the numeric **priority** field with the text field equivalent of supp1.

Modify the timestamps to conform to the data warehouse date format.

A	B	C	D	E	F	G	H
ticket_id	customer_id	subject	status	priority	create_date	updated_date	solved_date
900865	9999999	Error message 808 <*&#	Closed	Medium	2020-06-26 02:45:29	2020-06-27 21:31:48	2020-06-27 23:30:09
900866	2020411	Nullpointer exception on save	Open	High	2020-04-02 12:15:00	2020-04-05 11:15:00	
900867	2022010	Question about grades report	Open	High	2020-04-06 10:56:00		



### | Slide number 32

### | Instructor notes

|

### | Student notes

Using the example provided, the cleaning step would include replacing the blank **customer\_id** with a placeholder that uses the correct number of digits so that the dataset won't fail to load into the data warehouse. Depending on the use case, the data engineer might pass the file to the data analyst to directly correct the **customer\_id**.

Another cleaning task that's illustrated is removing corrupted data in column C that was pasted into the subject from an error message. The data engineer recognizes that some of the special characters could create issues when loading the data or trying to analyze it.

The final cleaning step for this example is to validate and modify data types. In this example, the data engineer modifies the numeric priority values, such as 1, 2, and 3, to match the text values in the other ticket system. The engineer maps 1 to high, 2 to medium, and 3 to low. They also modify the date fields to match the timestamp format that the data warehouse uses, which is YYYY-MM-DD HH:MI:SS.



## Key takeaways: Data cleaning



- Data cleaning is about preparing the source data for use in a pipeline.
- Cleaning is usually done for each source based on the characteristics of that source.
- Cleaning includes tasks such as removing unneeded, duplicate, or invalid data, and fixing missing values and data types.
- How the cleaning tasks resolve issues depends on the role of the person who cleans the data.

**| Slide number 33**

**| Instructor notes**

|

**| Student notes**

Here are a few key points to summarize this section.

Data cleaning is about preparing the raw data for use in a pipeline.

Cleaning is usually done per source and includes tasks such as removing duplicates, fixing values, and adjusting data types.

# Data enriching

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| Slide number 34

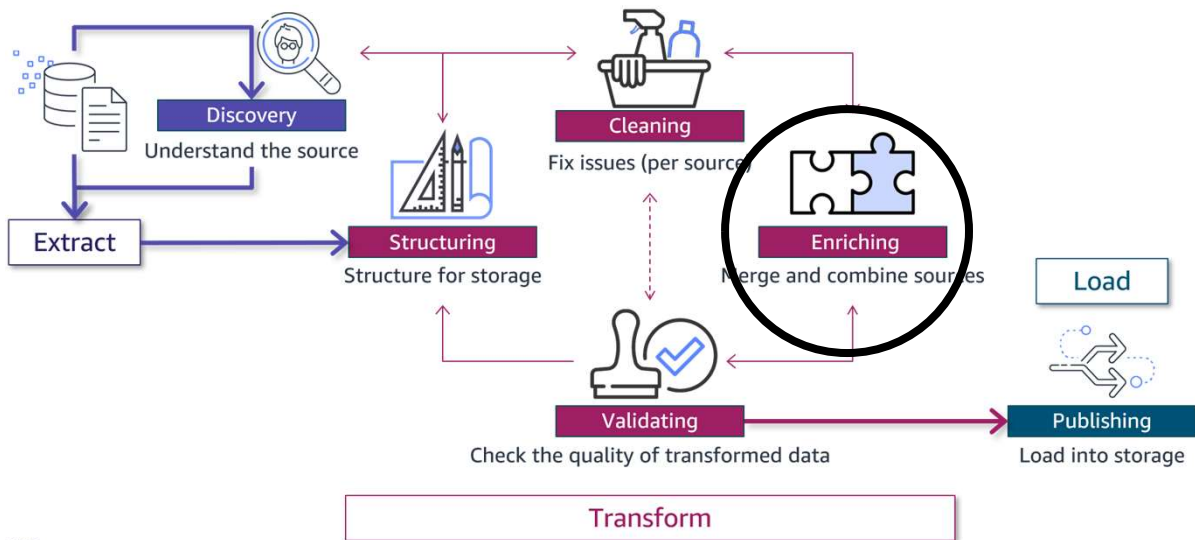
| Instructor notes

|

| Student notes

This section looks at the data enriching step of the data wrangling process.

## The enriching step within data wrangling



| Slide number 35

| Instructor notes

|

| Student notes

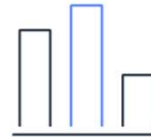
The enriching step is the fourth defined step within the data wrangling process, following cleaning. In the data enriching step, you start to pull together related data from disparate sources to create the dataset that you need for your analysis. At this stage, you start to merge or combine data from different sources to support your use cases.

## Decisions and tasks in enriching



### Merge sources

- Combine data from cleaned source files into a single dataset.



### Supplement data

- Add additional values to support analysis or desired visualization.



### | Slide number 36

### | Instructor notes

|

### | Student notes

Data enriching could be relatively simple, such as merging similar files from multiple sources. For example, you could combine the support tickets from ticket system 1 and ticket system 2.

This step could also be much more complex. The Elements of Data module presents an example where public health data is combined with customer data from a healthcare provider's patient system plus event data from a mobile app that monitors heart rate. This would require a much higher level of transformation and processing to merge these three very different sources into a dataset to be analyzed or acted upon.

Enriching might also include aggregations or other calculations that are added as new fields to the dataset.

## Data enriching example: Combine disparate sources

Combine rows from supp1 with rows from supp2.

Add a **ticket\_type** category for supp2 rows.

Add the sales region.

A	B	C	D	E		F	G	H	
ticket_id	customer_id	subject	status	priority	ticket_type	create_date	updated_date	solved_date	sales_region
900865	9999999	Error message 808	Closed	Medium	Problem	2020-06-26 02:45:29	2020-06-27 21:31:48	2020-06-27 23:30:09	
900866	2020411	Nullpointer exception on save	Open	High	Problem	2020-04-02 12:15:00	2020-04-05 11:15:00		USEast
900867	2022010	Question about grades report	Open	High	Question	2020-04-06 10:56:00			USWest
900862	1744899	Nightly batch failing	Closed	Medium	Problem	2020-01-04 08:06:00	2020-04-31 09:00:00	2020-04-31 09:00:00	Europe
900865	1744005	Intermittent error message	Open	Low	Problem	2020-01-06 11:14:00			Europe



### | Slide number 37

### | Instructor notes

|

### | Student notes

Using the support ticket example, the primary enriching step is to combine the two sets of support ticket files into a single file. Assuming that you use Excel to perform the same set of structuring and cleaning steps on the export from supp1, the enriching step could start with appending the supp1 data to the bottom of the supp2 data, making sure that all the columns match up.

Because the intent in the example scenario is to access the ticket information based on the sales region, the data engineer might query the sales system to get the sales owner by customer\_id and use that to create a script to add the sales\_region column to all rows in the combined file.

An optional enriching step might be for the data analyst to review the tickets for the supp2 system and categorize the subjects into ticket types. This would make it possible to organize the tickets into categories that are similar to the ticket types in supp1. This might be done by manual review in this example, but on a larger scale, this might be a step that happens later as its own processing step. For example, an ML model might be run against the data

that is stored in the data lake to perform analysis on the ticket subjects to determine whether they represent problems or questions.

## Key takeaways: Data enriching



- Data enriching is about combining your data sources together and adding value to the data.
- During enriching, you might merge sources, add additional fields, or calculate new values.

**| Slide number 38**

**| Instructor notes**

|

**| Student notes**

Here are a couple of key points to summarize this section.

The data enriching step is about pulling together related data from disparate sources to create the dataset that you need for your analysis. During enriching, you might merge sources, add additional fields, or calculate new values.



| Slide number 39

| Instructor notes

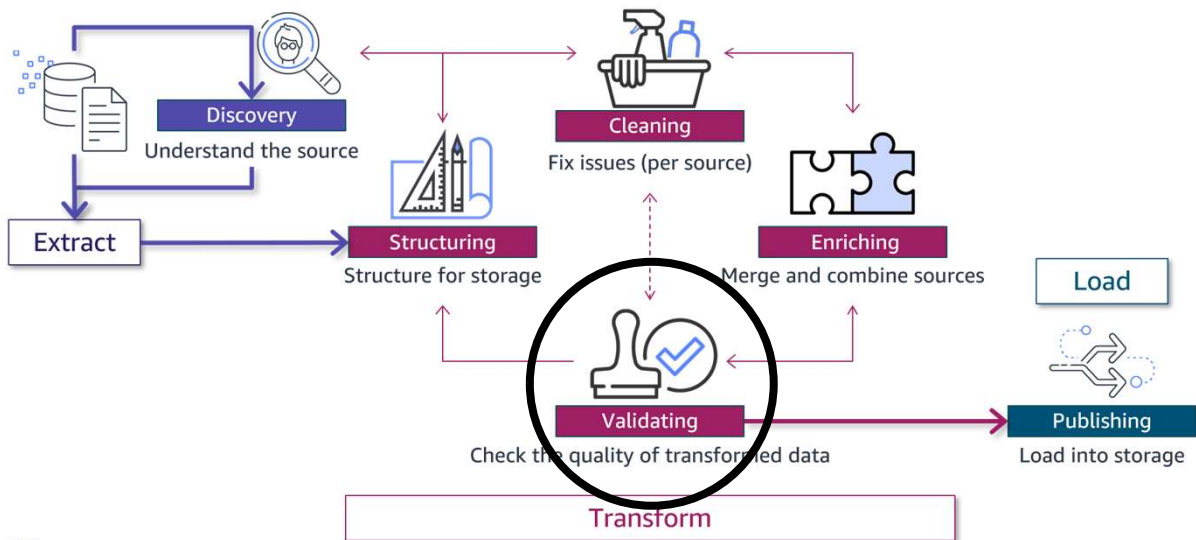
|

| Student notes

This section looks at the data validating step of the data wrangling process.



## The validating step within data wrangling



| Slide number 40

| Instructor notes

|

| Student notes

The validating step is the fifth defined step within the data wrangling process, following enriching. In the data validating step, you make sure that the dataset you have created for your analytics or ML use case has the veracity that you need.

## Decisions and tasks in validating



### Audit your work

- Count expected rows.
- Check consistency.
- Check expected formats and data types.
- Check for duplicates.
- Check for PII.
- Check for outliers.



### Fix the data

- Address audit findings.
- Address issues at the source.
- Fix issues with integration tools or scripts.



### | Slide number 41

### | Instructor notes

|

### | Student notes

Validating has some similarities with the cleaning step. The initial cleaning step focuses on making sure that the data is suitable to successfully load into your pipeline. But the validating step is about ensuring the integrity of the dataset that you have created through the structuring, cleaning, and enriching steps for the combined source files.

For example, during cleaning, you might make sure that an email field has a value of the correct data type. But during validation, you might verify whether the email address and domain represent valid values. There is definitely overlap in these two steps, and both focus on the veracity and, by extension, the value of the dataset that you are creating. Both are also iterative.

As noted with other steps, validating might include a combination of automated and manual checks, and could be performed by different roles.

The data engineer might focus on tasks that validate the accuracy of any automated structuring, cleaning, or enriching activities. For example, the engineer could check the row

counts and validate that the data formats are consistent and correct for the target data store. They would also check for duplicates on any field that should be unique and ensure that no fields contain PII, unless it's necessary to fulfill the business need.

A data analyst might look at the dataset and determine its accuracy from a business perspective. Do the ranges of data make sense? Are there outliers? Are there errors that can be fixed in the source systems?

As you've heard before multiple times, this step might be iterative as you check the results of your work, find issues, and address them. A data engineer might need to change configurations in an integration tool or in the scripts that they wrote to perform data wrangling steps, and then validate the results until they are satisfied.

## Data validating example

Address duplicate ticket IDs.

Find the correct **customer\_id**, fix in the source, and run a script to get **sales\_region**.

A	B	C	D	E		F	G	H	
ticket_id	customer_id	subject	status	priority	ticket_type	create_date	updated_date	solved_date	sales_region
900865	2070555	Error message 808	Closed	Medium	Problem	2020-06-26 02:45:29	2020-06-27 21:31:48	2020-06-27 23:30:09	USEast
900866	2020411	Nullpointer exception on save	Open	High	Problem	2020-04-02 12:15:00	2020-04-05 11:15:00		USEast
900867	2022010	Question about grades report	Open	High	Question	2020-04-06 10:56:00			USWest
900862	1744899	Nightly batch failing	Closed	Medium	Problem	2020-01-04 08:06:00	2020-04-31 09:00:00	2020-04-31 09:00:00	Europe
900865	1744005	Intermittent error message	Open	Low	Problem	2020-01-06 11:14:00			Europe

Count total rows and check against rows from individual systems.

Supp2 rows()=3  
Supp1 rows()=2  
Target rows()=5



### | Slide number 42

### | Instructor notes

|

### | Student notes

Using the support ticket example, the validation updates would first include modifying one of the duplicate ticket IDs. Then, the data engineer would find the correct **customer\_id** value for the first row so that the **sales\_region** can be matched to the customer. And finally, the engineer would count the total rows in the combined file to make sure that it matches the sum of the two sets of data.

## Key takeaways: Data validating



- Data validating is about ensuring the integrity of the dataset that you have created.
- Validating tasks might overlap with cleaning tasks.
- This step might be iterative as you check the results of your work, find issues, and address them.

### | Slide number 43

### | Instructor notes

|

### | Student notes

Here are a couple of key points to summarize this section.

The purpose of the validating step is to ensure the integrity of the dataset that you have created through the structuring, cleaning, and enriching steps for the combined source files.

The tasks of validating might overlap with cleaning steps, but the main distinction is that, during validation, you make sure that the data is valid—not just that it will load successfully.

This is also a step that will be iterative as you check results, find and fix errors, and then rerun source extracts.

# Data publishing

Ingesting and Preparing Data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

| **Slide number 44**

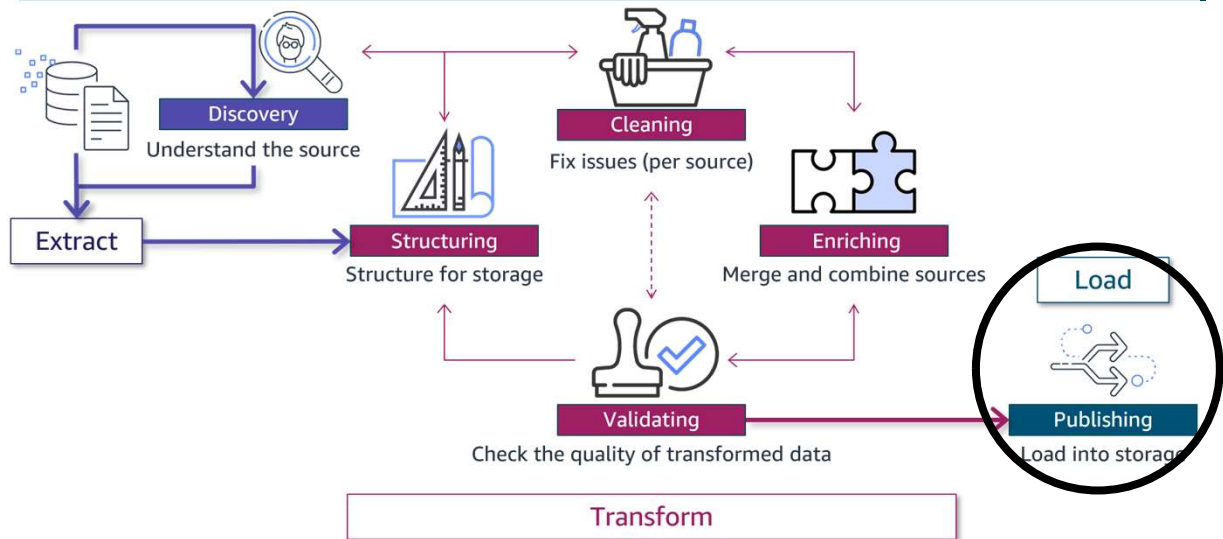
| **Instructor notes**

|

| **Student notes**

This section looks at the data publishing step of the data wrangling process.

## The publishing step within data wrangling



| Slide number 45

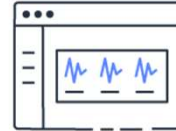
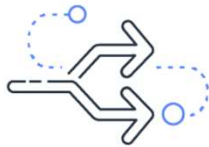
| Instructor notes

|

| Student notes

The publishing step is the sixth defined step within the data wrangling process, following validating. In the data publishing step, you make the dataset available for use within the pipeline.

## Decisions and tasks in publishing



### Move data into permanent storage

- Apply file management techniques (formats, compression, organization).
- Securely transfer data from source to destination.
- Write the dataset or datasets to the correct storage locations.

### Make data available to consumers

- Apply appropriate access controls to data.
- Save metadata about the dataset.
- Configure how often and when to run the extract.
- Log the update history.



### | Slide number 46

### | Instructor notes

|

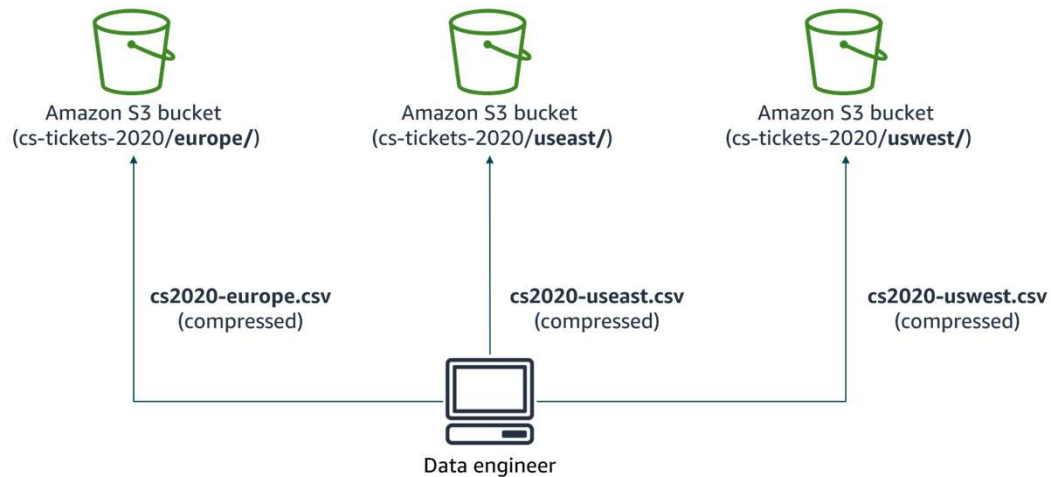
### | Student notes

During publishing, you apply your structuring decisions. You need to write data to the appropriate folders or partitions, and apply the file management techniques that you decided on. For example, you might split files to make them load faster and compress them to save on storage costs in the data lake.

As part of your publishing process, you also need to make sure that the storage locations that you have set up allow least-privilege access to the users or systems that should be able to consume the data. You also need to save metadata about the dataset to make it findable to consumers.



## Data publication example: Moving data to permanent storage



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

47

### | Slide number 47

### | Instructor notes

|

### | Student notes

Continuing with the customer support ticket example, to meet the desired request to segment access by sales region, the first step to publish the dataset would be to group the dataset by sales\_region and export each region's rows to a .csv file. In this small sample, that means three .csv files: one for Europe, one for US East, and one for US West. The data engineer would then compress each file.

The next step would be for the data engineer to sign in to an AWS account that has access to the designated S3 bucket and securely upload each file to the bucket or folder that is associated with the matching region. As illustrated on the slide, the S3 bucket is cs-tickets-2020, and regional files are uploaded to the folders called europe, useast, and uswest.

## Data publication example: Uploading the files

Export to **europa.csv**; zip as **europa.zip**; upload to **s3://cs-tickets-2020/europa/cs2020-europa.zip**

```
ticket_id,customer_id,subject,status,priority,ticket_type,create_date,updated_date,solved_date,
sales_region
900862,1744899,Nightly batch failing,Closed,Medium,Problem,2020-01-04 08:06:00,2020-04-31
09:00:00,2020-04-31 09:00:00,Europe
100865,1744005,Intermittent error message,Open,Low,Problem,2020-01-06 11:14:00,,Europe
```

Export to **useast.csv**; zip as **useast.zip**; upload to **s3://cs-tickets-2020/useast/cs2020-useast.zip**

```
ticket_id,customer_id,subject,status,priority,ticket_type,create_date,updated_date,solved_date,
sales_region
900865,2070555,Error message 808,Closed,Medium,Problem,2020-06-26 02:45:29,2020-06-27
21:31:48,2020-06-27 23:30:09,USEast
900866,2020411,NullPointerException on save,Open,High,Problem,2020-04-02 12:15:00,2020-04-05
11:15:00,,USEast
```

Export to **uswest.csv**; zip as **uswest.zip**; upload to **s3://cs-tickets-2020/uswest/cs2020-uswest.zip**

```
ticket_id,customer_id,subject,status,priority,ticket_type,create_date,updated_date,solved_date,
sales_region
900867,2022010,Question about grades report,Open,High,Question,2020-04-06 10:56:00,,USwest
```

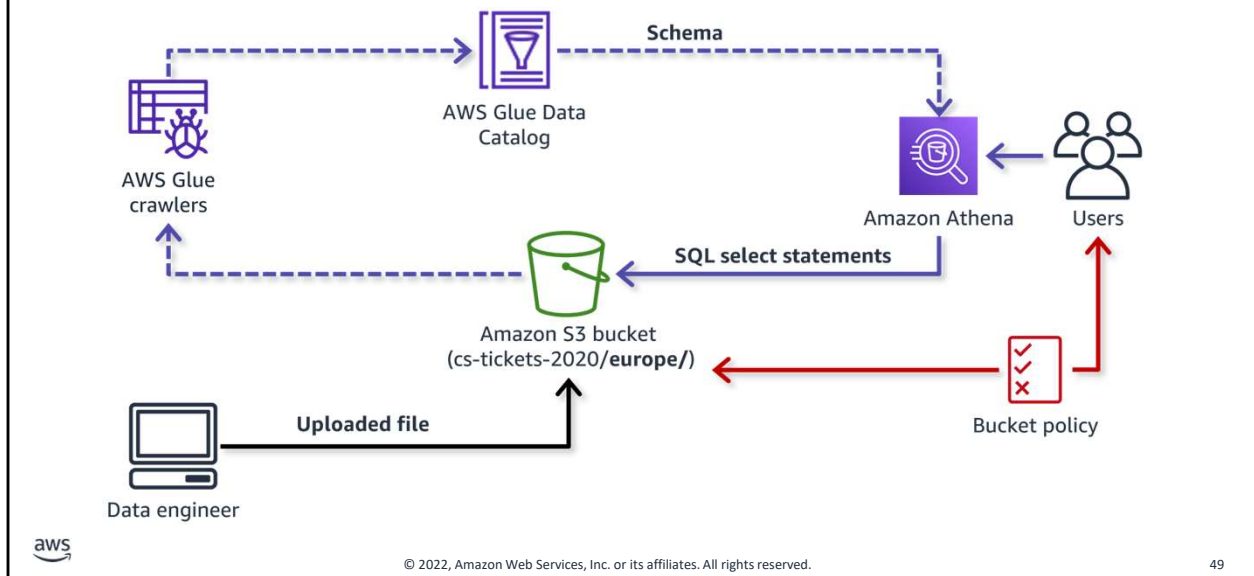


### | Slide number 48

### | Instructor notes

| This hidden slide shows the actual contents of each file based on the sample data in the prior slides should you choose to use it. Note that the ticket value of 100865 in the Europe file reflects the change to remove the duplicate ticket id shown on the validating example, row 5

## Data publication example: Making the data available



### | Slide number 49

### | Instructor notes

| This slide refers to the modern data architecture, storage layer that the Design Principles and Patterns for Data Pipelines module described. The Ingesting by Batch or by Stream module, which is designed to follow this module, covers AWS Glue features, including crawlers and the data catalog, in more detail. The lab that is associated with that module (Querying Data by Using Athena) introduces the use of Amazon Athena to query Amazon S3.

|

### | Student notes

After the data engineer uploads the files to the S3 buckets, they can run an AWS Glue crawler on the files in the buckets to derive schema information about the files and store it in the AWS Glue Data Catalog. The catalog provides the information that Amazon Athena needs to query the files on Amazon S3. Authorized consumers can use Athena to directly query the data in the data lake or use Amazon Redshift Spectrum to combine data from the customer table in the Redshift data warehouse with data from the data lake files. A bucket policy on each S3 bucket, combined with IAM groups for each regional sales group, limits access to each bucket to the appropriate users and gives them read-only access to the files for their region.

AWS Glue is an integration service to simplify ETL processes. AWS Glue crawlers and the AWS Glue Data Catalog are introduced in the Design Principles and Patterns for Data Pipelines module. The Ingesting by Batch or by Stream module describes AWS Glue features in more detail.

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 by using standard SQL. You will use Athena in the Querying Data by Using Athena lab.

AWS Identity and Access Management (IAM) is the main service for securely controlling access to AWS services. The Securing and Scaling the Data Pipeline module discusses IAM in greater detail.

## Key takeaways: Data publishing



- Data publishing is about making the dataset available for use.
- In this step, you move the data to the storage layer of the pipeline. The decisions that you make during the structuring step about organization and file management are used to load the data into storage.
- At this step, you also need to provide methods to update the dataset and monitor ongoing ingestion.

**| Slide number 50**

**| Instructor notes**

|

**| Student notes**

Here are a few key points to summarize this section.

Data publishing is about making the dataset available for use within the pipeline.

To publish data, you move it to the target storage based on decisions that you made during the structuring step.

To continue to make the data available, you also need to set up a method to update the dataset on a timely basis and monitor ongoing ingestion.



| **Slide number 51**

| **Instructor notes**

|

| **Student notes**

This section summarizes what you have learned and brings the module to a close.

## Module summary

---

This module prepared you to do the following:

- Distinguish between the processes of ETL and ELT.
- Define data wrangling in the context of ingesting data into a pipeline.
- Describe key tasks in each of these data wrangling steps:
  - Discovery
  - Structuring
  - Cleaning
  - Enriching
  - Validating
  - Publishing



### | Slide number 52

### | Instructor notes

|

### | Student notes

This module focused on the extract, transform, and load tasks that are part of ingesting data into your pipeline. The module compared the extract, transform, and load (ETL) and extract, load, and transform (ELT) approaches. It also introduced the steps that are involved in wrangling data for use in your pipeline.

## Module knowledge check



- The knowledge check is delivered online within your course.
- The knowledge check includes 10 questions based on material presented on the slides and in the slide notes.
- You can retake the knowledge check as many times as you like.

**| Slide number 53**

**| Instructor notes**

|

**| Student notes**

Use your online course to access the knowledge check for this module.



## Sample exam question

A data engineer has been asked to provide data to support a new sales report. The report will combine sales data for four different products, which have been tracked in four different systems. The resulting data needs to be segmented into regions, and each regional dataset should be available only to sales teams for the region.

What should the data engineer do first to meet this request?

Identify the key words and phrases before continuing.

The following are the key words and phrases:

- Four different systems
- Data **should be available only to sales teams for the region** where the sale occurred
- What should the data engineer do **first** to meet this request?



### | Slide number 54

| **Instructor notes:** The key words section is animated to be revealed on click.

|

### | Student notes

The question notes that you are working with four different systems, which implies differences in fields and file structures. The question also indicates that access should be based on sales region. Finally, the question asks what the data engineer should do *first*. Therefore, you need to look for the answer that would happen first in a data wrangling process.

## Sample exam question: Response choices

A data engineer has been asked to provide data to support a new sales report. The report will combine sales data for four different products, which have been tracked in **four different systems**. The resulting data needs to be segmented into regions, and each regional dataset **should be available only to sales teams for the region**.

What should the data engineer do **first** to meet this request?

Choice	Response
A	Create an S3 bucket with folders for each product.
B	Export data from each system, and merge the data into a single file.
C	Identify the relationships between fields across each of the sources.
D	Replace null values in each source system field with zeroes.



### | Slide number 55

### | Instructor notes

|

### | Student notes

Use the key words that you identified on the previous slide to review each of the responses to determine which one best addresses the question that was presented.

## Sample exam question: Answer

The correct answer is C.

Choice	Response
C	Identify the relationships between fields across each of the sources.



### | Slide number 56

### | Instructor notes

|

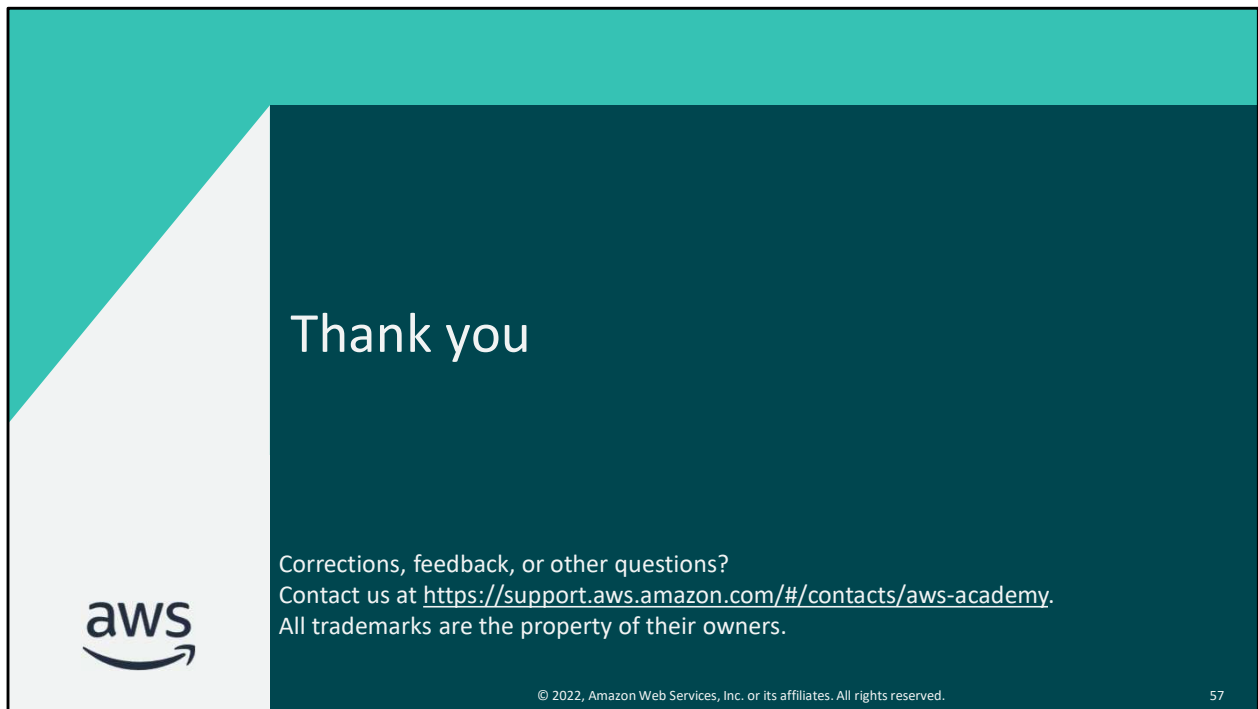
### | Student notes

Choice A (Create an S3 bucket with folders for each product) is incorrect for two reasons. First, creating the folders would not be the first task. Second, the access will be by sales region instead of product, so folders by product would not meet the requirement.

Choice B (Export data from each system and merge them into a single file) is not the best choice because discovery hasn't occurred yet to identify how the fields in each source relate to each other. Therefore, the data engineer can't successfully merge them yet.

Choice D (Replace null values in each source system field with zeroes) is not correct because the data engineer doesn't yet know enough about the data to know whether to replace the nulls with zeroes.

The correct answer is choice C (Identify the relationships between fields across each of the sources). This is a discovery task that will help the data engineer determine how to map the fields between the different source files.



**| Slide number 57**

**| Instructor notes**

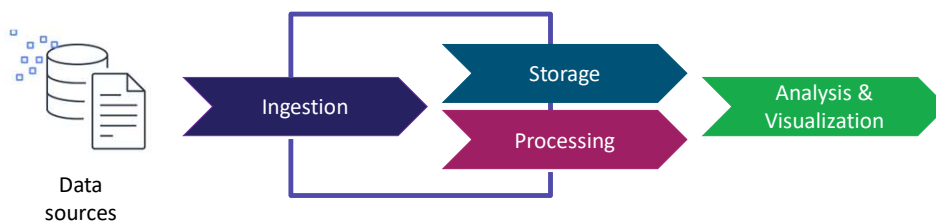
**|**

**| Student notes**

That concludes this module. The Content Resources page of your course includes links to additional resources that are related to this module.

## Ingesting data involves storing and processing data

#6



### | Slide number 58

### | Instructor notes

|

### | Student notes

In the Data-Driven Organizations module, you learned about the layers of a data pipeline. This simplified drawing represents that pipeline and shows each layer as distinct. In practice, ingesting data is about getting data from an external source into your pipeline, where the data can be used. The ingestion process is tied to the tasks of storing and processing data.

Specifically, the ingestion process incorporates extracting data from a source that isn't already available in your pipeline, loading the data into some type of temporary storage in the pipeline, and processing the data by transforming it before or after the data is loaded into more permanent storage.

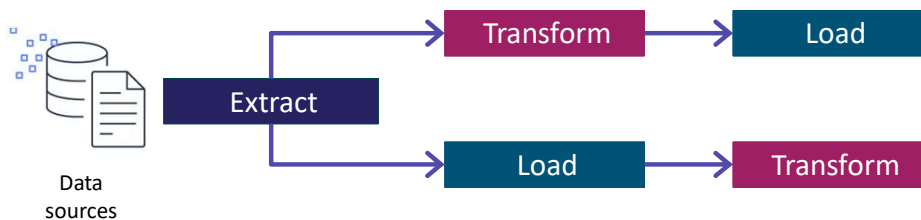
The type and scope of transformations, and where they occur across the data pipeline, highly depend on the business need and the nature of the data that is being ingested.

## ETL and ELT

#7

### Extract transform load (ETL)

1. **Extract** structured data.
2. **Transform** the data into a format that matches the destination.
3. **Load** the data into structured storage for defined analytics.



### Extract load transform (ELT)

1. **Extract** unstructured or structured data.
2. **Load** the data into the data lake in the format that is as close to the raw form as possible.
3. **Transform** the data as needed for analytic scenarios.



| Slide number 59

| Instructor notes

|

| Student notes

This slide mirrors the layers that were depicted on the previous slide and compares two methods of ingesting data. You need to extract data from a source and then use a combination of loading the data into storage and transforming or processing the data for its intended purpose.

The traditional method of ingesting and preparing data for analytics was ETL. With ETL, data is *extracted* from its source, *transformed* into a structured format that is ready to be used in analytics applications, and *loaded* into structured storage, such as a data warehouse. Data scientists or analysts query the data warehouse and might perform additional transformations and processing as part of their analyses.

As described in the Design Principles and Patterns for Data Pipelines module, data stores evolved from handling only structured data to include options for handling high volumes of unstructured data. Specifically, there has been a move from structured data in data warehouses towards unstructured and semistructured data in data lakes. Tools and methods for ingesting data into a pipeline evolved along the same lines. ETL techniques

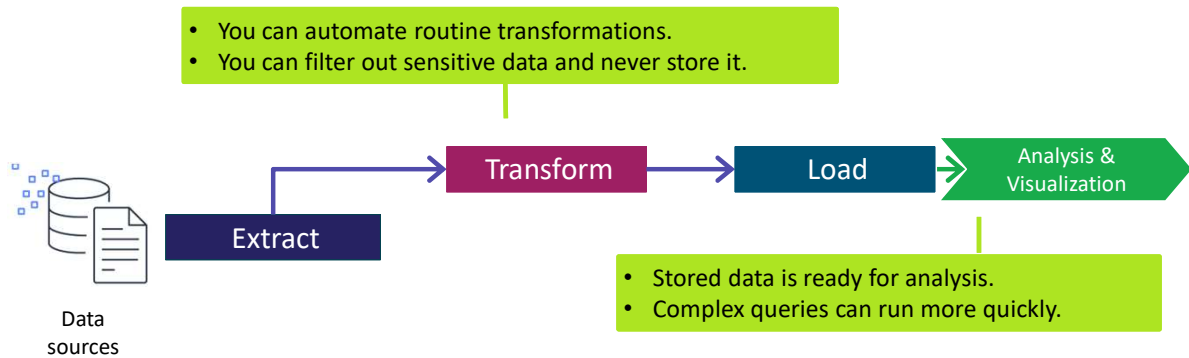
have continued to evolve and are still a valuable part of data architectures for structured data.

However, the trend towards collecting high volumes of unstructured and semistructured data and the desire to make that data available for a variety of uses led to a different approach for data ingestion: ELT. With an ELT approach, data is extracted from its source and transformed only enough to store it in a consumable but relatively raw state in a data lake.

Transformations that might be needed for specific uses are done later, as consumers access data and build analytics solutions with it.

## Benefits of extract transform load (ETL)

#8



| Slide number 60

| Instructor notes

|

| Student notes

In addition to the general guidance that ETL works well for structured data that is destined for a data warehouse, and ELT works best for unstructured data that is destined for a data lake, there are other reasons to choose one or the other. This slide highlights a few considerations.

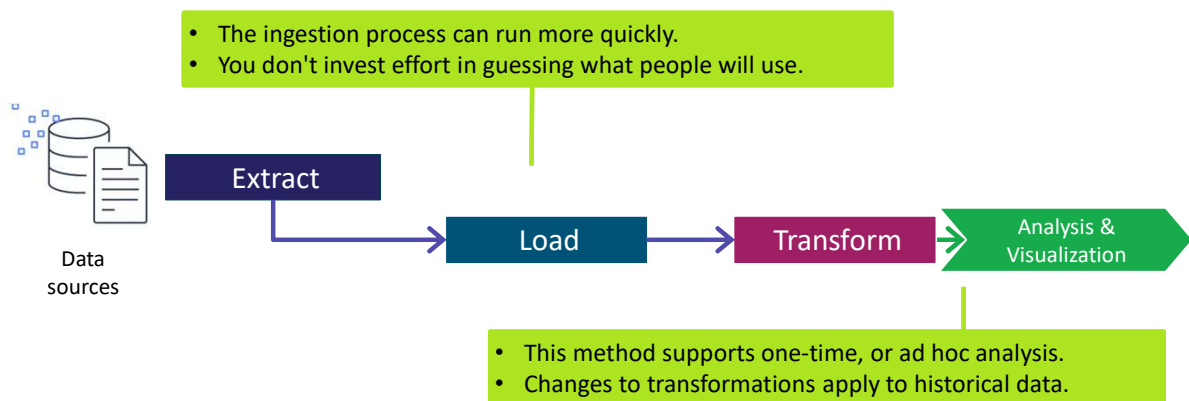
ETL stores data that is ready to be analyzed, so it can save time for an analyst. If analysts routinely perform the same transformations on data that they analyze, it might be more efficient to incorporate those transformations into the ingestion process. So, with ETL, you trade longer transformation processing time before the data is available for faster access when you are ready to query it.

Another advantage of performing transformations up front is that you can filter out personally identifiable information (PII) or other sensitive data that could create a compliance risk. If you never store the sensitive data, you reduce the risk without the need to evaluate additional security measures.



## Benefits of extract load transform (ELT)

#9



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

61

| Slide number 61

| Instructor notes

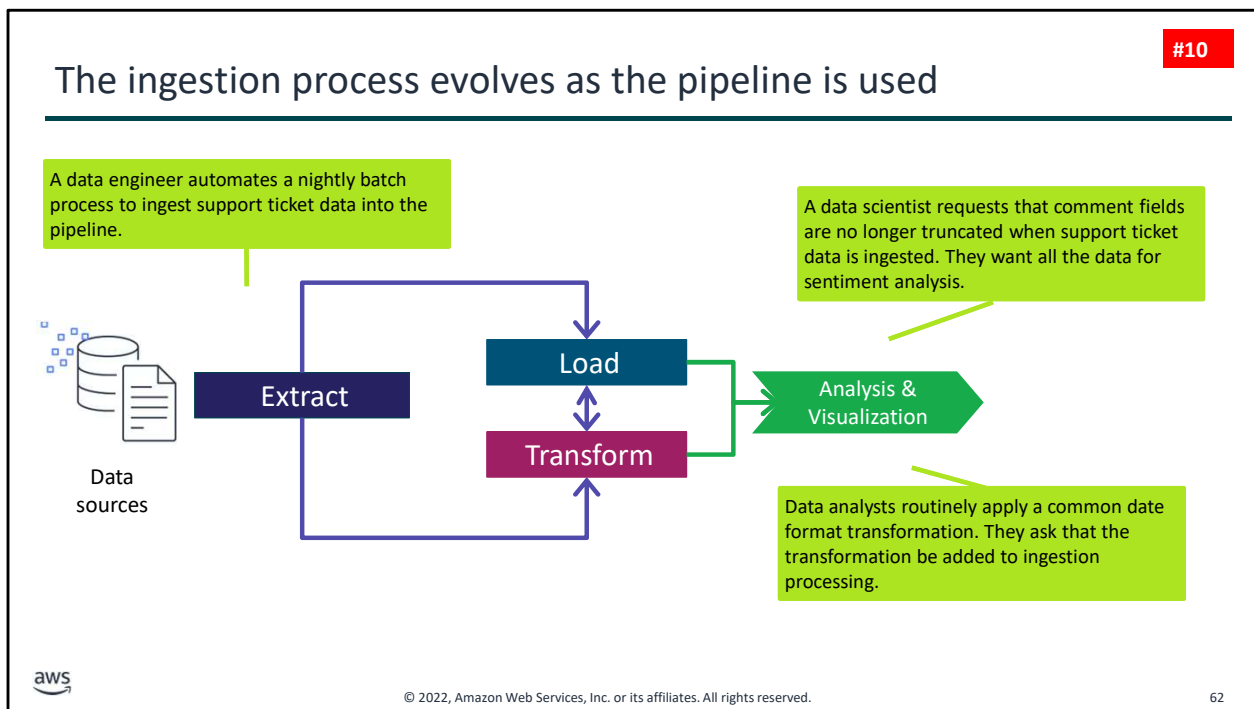
|

| Student notes

On the other hand, ELT provides more flexibility to create new queries because analysts have access to greater amounts of raw data.

With ELT, the process to get the raw data into your pipeline can be quicker because you aren't required to transform the data as much before making it available in the data lake. So, with ELT, your trade-off is that you get the raw data available more quickly, but you spend more time to transform the data when you're ready to work with it.

Another advantage of storing the raw data and only transforming it downstream is that changes to the transformation (for example, including additional fields or formatting data differently) would be run against all the historical data that was loaded when the transformation is updated. If the transformations occur before you load the data, your changes can apply to new data that is ingested, but it might be impossible or difficult to apply them to historical data.



### | Slide number 62

### | Instructor notes

| Depending on the makeup of the class, it might be worth spending a bit more time talking with the students about their primary role and how they will interact with the pipeline and the other roles.

|

### | Student notes

As other modules also note, how you design your ingestion process really depends on the business problem that you're addressing and the trade-offs that make the most sense for your use case. The design is likely to evolve based on both the builder's experience and the pipeline users' experience.

In modern architectures, ETL and ELT approaches are blurred. Each role that works with the pipeline might perform a different part of the ingestion and transformation process by using different tools and access. For example, the data engineer might use an ingestion tool that handles common cleaning or formatting transformations before loading a dataset into your data lake. Additional transformations might occur only when the data is pulled for a particular analytics use case. The data analyst might apply a set of transformations on the ingested data to prepare it for a specific report. A data scientist might perform additional

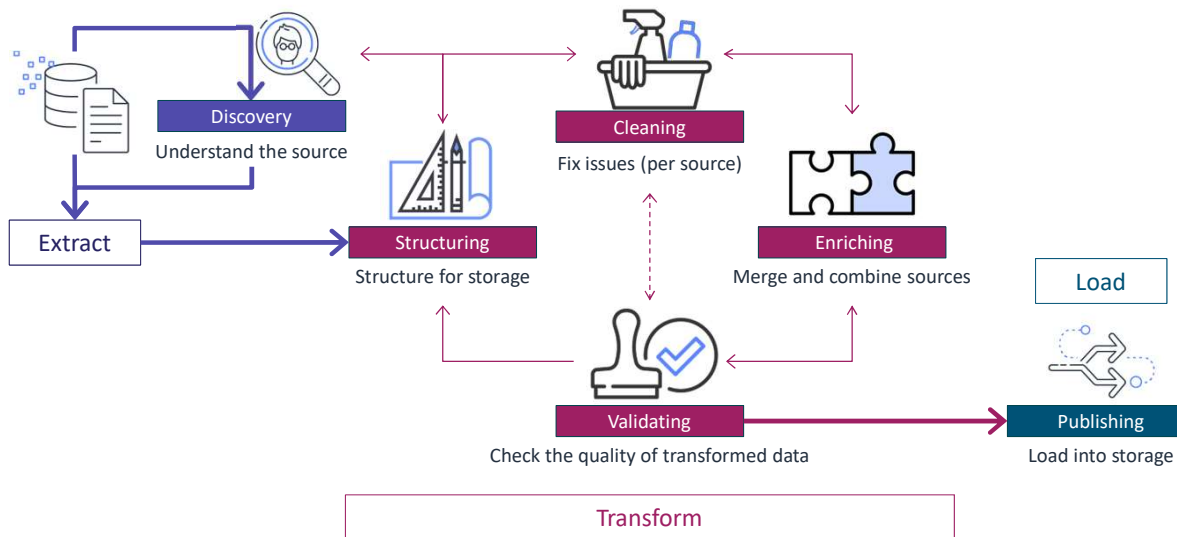
discovery and transformation on the ingested data to investigate hypotheses about relationships within the data.

As patterns of use and understanding of the dataset evolve, you might revisit where and how transformations occur. For example, if many analysts perform the same recurring transformation to work with a dataset, adding those transformations into the processing that occurs before the data is loaded might improve performance or reduce costs.

Alternatively, you might recognize additional needs that you could serve by reducing the amount of transformation that is performed before loading the data. For example, you might truncate a large field to save costs, but a data scientist determines that there's value in the truncated field and asks you to remove that transformation.

As the data engineer, you want to monitor how the pipeline performs, how it is accessed and used, and how you might modify your processes to optimize the value.

## Data wrangling steps



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

63

### | Slide number 63

### | Instructor notes

|

### | Student notes

This slide presents the steps of data wrangling as distinct and part of ingesting data into the pipeline. In practice, there might not be clear delineations between the steps, there might be iterations within and across steps, and individual steps might not occur as part of ingesting a data source into the pipeline. Each step might be more or less complex, depending on the goal of the specific ingestion that you are working on.

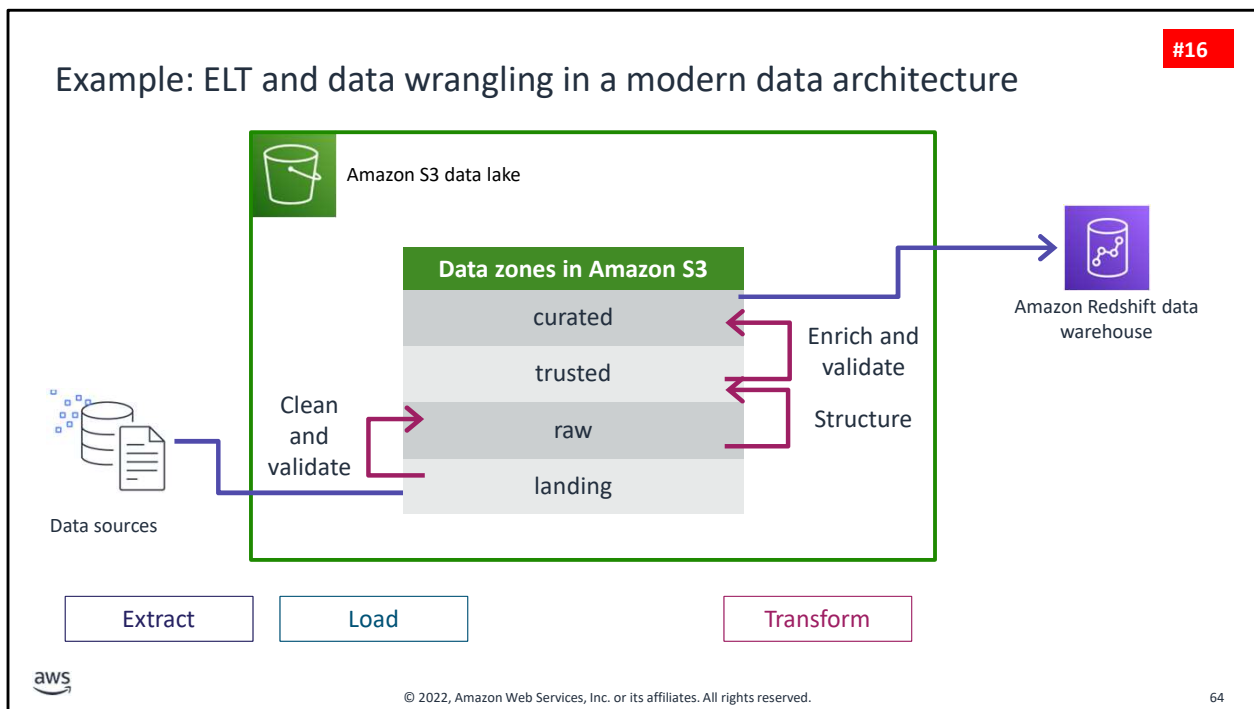
The overall ingestion and data wrangling process could be as simple as opening a source file in Microsoft Excel, manipulating columns and individual values in the spreadsheet, manually checking the validity of the data, and then saving it as a .csv file and uploading it to an Amazon Simple Storage Service (Amazon S3) bucket. At the other end of the spectrum, complex transformations might be needed to support high volumes of incoming data for an ML application. The data engineer and data scientist might use a variety of scripts and tools to wrangle the data for this use case.

As illustrated on this slide, the steps of data wrangling generally coincide with the steps of ETL processing. However, it's worth making a distinction here between ETL processing and data wrangling.

Traditionally, ETL processing has been more of the data engineer's domain. ETL processing encompasses scripting and automating transformation steps with batch jobs that run on infrastructure that is controlled by IT. Analysts and data scientists worked only with data that was in a defined, consumable state.

Data wrangling is more associated with an ELT flow where business users or data scientists transform datasets that are available to them within the pipeline storage layer but that haven't been refined for a specific use case.

The continuing trend in cloud tools and services is to abstract infrastructure and coding tasks to provide more types of users with direct access to extract and transform data. A data engineer must be able to build complete analytics pipelines and transform data as needed for a use case. But it's also important to stay aware of tool updates that can provide their team's analysts and data scientists with more autonomy to work with available data.



**| Slide number 64**

**| Instructor notes**

|

**| Student notes**

There is overlap between the ingestion, storage, and processing layer concepts that you have learned so far. It's important to remember that the defined layers help to describe the activities that are performed and tools that are used as data moves through and is transformed in the pipeline. But pipeline activities are iterative within and across layers. Each pipeline's architecture might look quite different depending on the business problem being addressed, type of data being ingested, and skills and expertise of the team that supports the pipeline and accesses the data.

This diagram is derived from the modern data architecture in the Design Principles and Patterns for Data Pipelines module. The diagram illustrates an ELT flow and highlights how the data wrangling steps might occur. In this reference architecture, data is ingested into a storage layer that includes an Amazon S3 data lake and an Amazon Redshift data warehouse. Within the data lake, S3 buckets are used as named zones to store data in different states.

The landing zone bucket serves as temporary storage from which you perform just enough cleaning and validating to permanently save the ingested data into the raw zone bucket. You can then use data in the raw zone for additional use cases. This data might be structured, semistructured, or unstructured. At this stage, the initial extract and load steps are complete.

In this example, the transform part of ELT occurs when data in the raw zone of the data lake is transformed into a highly structured state that the Amazon Redshift data warehouse can use.

The data engineer or data scientist would need to structure, enrich, and validate the data before the warehouse can use it. Data is stored in different zones as these steps are performed.

Data structuring prepares the data for the trusted zone. Preparation for this zone might include activities such as conforming to the schema and partitioning specifications for the dataset based on data warehouse tables. Then, data enriching is performed on data in the trusted zone, modeling the data so that it can be joined with other datasets and stored in the curated layer. Data validation would also occur as part of these transformations before storing data into the next zone.

Datasets in the curated layer are ready for the Amazon Redshift data warehouse to ingest to make them available for low-latency access and complex SQL querying.

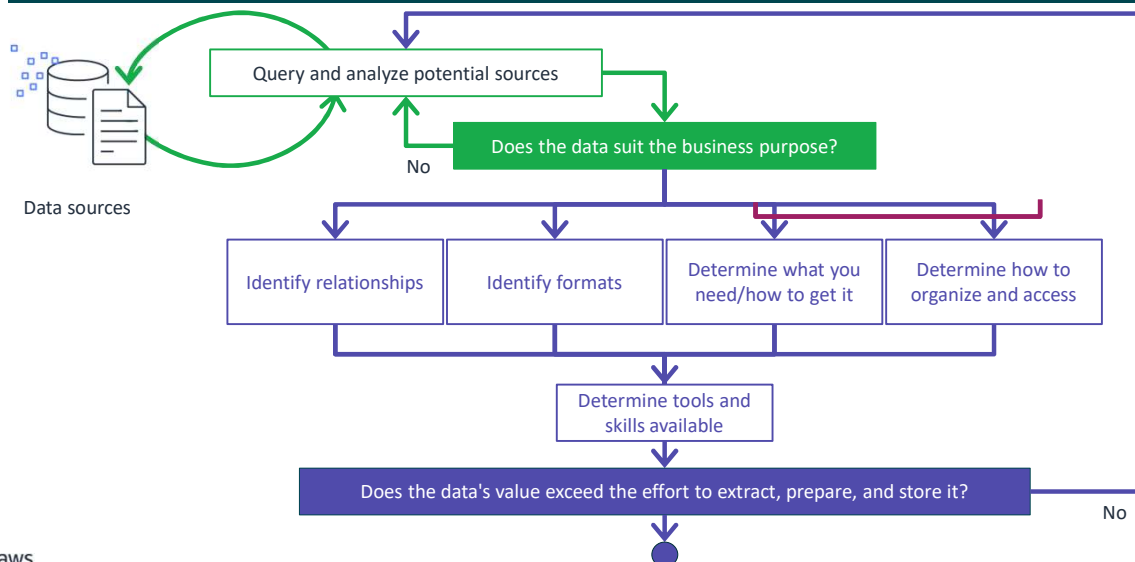
Depending on the use case, additional discovery might occur before the structuring, enriching, and validating steps take place. Additional cleaning might be needed as well. The process is often iterative.

If the process to load data into the data warehouse had been accomplished using a traditional ETL flow, the same wrangling tasks would still be needed. However, all the transformations (cleaning, structuring, enriching, and validating) would be performed on the data in a temporary staging location before the data is loaded directly into the data warehouse. Users would not have access to the data until processing was complete, and the raw data would not be available.

The next sections look at each of the steps in the wrangling process in a bit more detail.

## Decisions and tasks in discovery

#19



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

65

### | Slide number 65

### | Instructor notes

| You might want to stress or explore with students how this discovery flow presents the data engineer's perspective, but each role who is involved with the data pipeline has their own discovery tasks. Depending on the roles and backgrounds of your students, you could also lead a discussion to help students see the perspective of each role.

|

### | Student notes

The data discovery step is iterative, and different roles (analyst, data scientist, data engineer) might look for different things at this step. As a data engineer, you need to find sources that you think might be useful, query them, and analyze the raw data to decide whether the sources have value for your business purpose. An analyst or data scientist might tell you about a source that they want to use. What you learn from your initial results might lead you to refine your queries and analyze the updated results before you decide how to use each source.

Key tasks of this step include the following:


- Decide if a source seems to serve your business purpose. If it does, then you want to dig



deeper into the data to do the following:

- Identify what relationships exist within and between the raw data sources that you expect to use.
- Identify the data formats (for example, CSV, Parquet, Optimized Row Columnar [ORC]).
- Determine the range of data that you want from each source (for example, a time period or range of attribute values). Determine how to get the desired range out of the source and into your pipeline.
- Determine how you should organize the data to make it easy—but secure—to access and cost-effective to ingest and store. For example, what folder structure is appropriate? What's the best way to manage file size? How should you partition your target database table? You might need to choose a partition key that optimizes performance for your use case. Or you might need to establish folder structures in Amazon S3 that provide granular access to parts of the dataset to different users. For example, you might need to create Amazon S3 prefixes (or folders) for a particular segment of data so that you can grant access to each folder only to the team members that are authorized to access that segment. Your decisions about organization should also consider the structure that you want to use as part of the data wrangling process. For example, you might use the zone approach, which was illustrated earlier, to segment data that comes into the data lake into "raw," "landing," and "trusted" zones.
- Based on your analysis of the data relationships, formats, and organizational requirements, determine what tools you need. Also, determine what you have available to store, secure, transform, and query the type of data in each source. Consider whether you have the skills and resources that are needed to extract data from the source and prepare it for use.
- Based on these discovery activities, you might decide to work with the source. Or, you might determine that the effort and cost to extract and prepare the data for the intended business purpose doesn't provide enough value. If you determine that it's not worth the effort, you might start the whole process again and seek additional sources or look for ways to reduce the volume of data to be ingested, amount and type of storage that are required, or complexity of preparing the data for use.


## Decisions and tasks in cleaning



001	42	4/2	a	f
002		4/5	a	
003	78	5/5		
004	14	5/8	d	f
005	4200	5/9	f	a

### Remove data


- Remove unwanted columns.
- Remove duplicate values.
- Remove "garbage" values.



001	42		4/2	
002	0		4/5	
003	78		5/5	
004	14		5/8	
005	4200		5/9	

### Fill missing data


- Fill null columns.
- Fill missing values for mandatory fields.



001	42	22-04-02		
002	0	22-04-05		
003	78	22-05-05		
004	14	22-05-08		
005	4200	22-05-09		

### Check data types

- Validate or modify data types.



001	42	22-04-02		
002	0	22-04-05		
003	78	22-05-05		
004	14	22-05-08		
005	42	22-05-09		

### Fix outliers

- Identify outliers, and either remove or fix data as appropriate.

### | Slide number 66

### | Instructor notes

|

### | Student notes

Data cleaning includes tasks such as the following:

- Remove unwanted data.
- Fill in missing data values.
- Validate or modify data types.
- Fix outliers.

Depending on who does the cleaning, the methods and results might be different. For example, a data engineer who is only responsible for successfully loading the data into the pipeline might make sure that blank fields are replaced with values that fit the data type. However, a data analyst might replace blanks by adding corrected values.

Let's look at each task in a bit more detail.

Even when you intend to load data into your data lake for a variety of use cases, there are elements that you don't want to store. For example, columns that include PII or fields that aren't relevant for analysis. You would also want to remove duplicate values that are

present in the source file. You might need to remove values that don't represent the intended contents for a field or attribute. For example, a field with corrupted data or unwanted characters that won't conform to the target's requirements or that would prevent the row from being joined with other related data successfully.

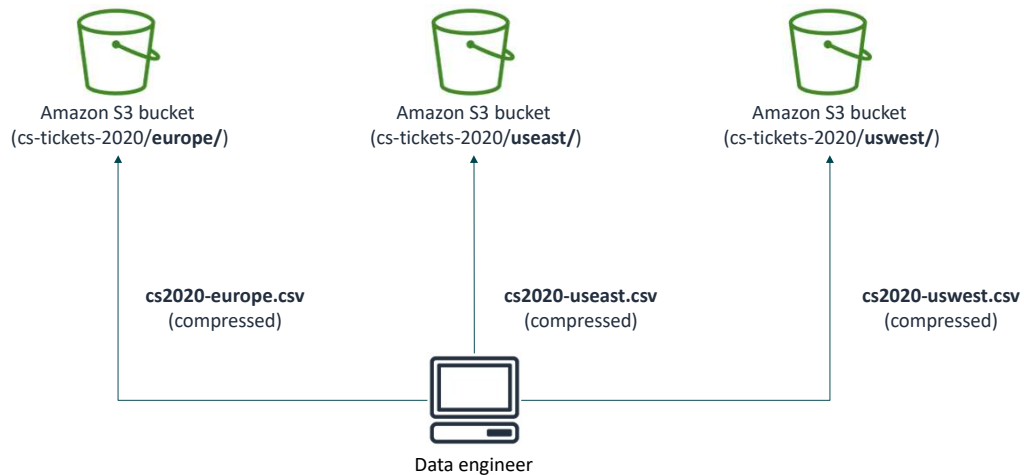
Data cleaning also addresses missing data. For example, fields that are blank might need to be converted to zeroes if the field should represent a number. You might also need to fill in data for mandatory fields so that the dataset will load into a target structure. For example, if the target table expects a postal code for each record, but your source has some records that are missing this field, the cleaning process would need to put a generic value in to replace the missing value.

Another cleaning step that helps to ensure that the data being ingested can be stored and referenced as intended is to validate that data types in the source file match the target and to modify those that don't. For example, if the source file stores dates as a text string and the target requires a specifically formatted date field, you would need to convert the strings to properly formatted date fields.

One cleaning step that might require a bit more analysis is to identify outliers and either remove them from the dataset or fix them at the source. A data engineer might run a script that identifies values that are far outside the range of other data values and remove those rows. A data analyst or data scientist might dig deeper to determine if the outlier is an error that should be fixed at the source or data that should be discarded. In the example presented here, the value of 4200 in row 5 was an outlier. The analyst replaced it with 42.

## Data publication example: Moving data to permanent storage

#47



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

67

### | Slide number 67

### | Instructor notes

|

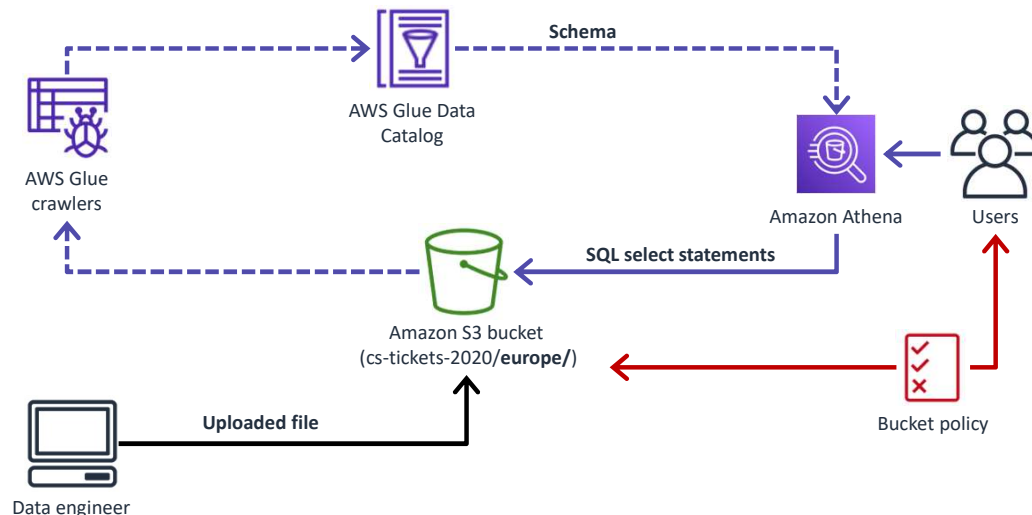
### | Student notes

Continuing with the customer support ticket example, to meet the desired request to segment access by sales region, the first step to publish the dataset would be to group the dataset by sales\_region and export each region's rows to a .csv file. In this small sample, that means three .csv files: one for Europe, one for US East, and one for US West. The data engineer would then compress each file.

The next step would be for the data engineer to sign in to an AWS account that has access to the designated S3 bucket and securely upload each file to the bucket or folder that is associated with the matching region. As illustrated on the slide, the S3 bucket is cs-tickets-2020, and regional files are uploaded to the folders called europe, useast, and uswest.

## Data publication example: Making the data available

#49



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

68

### | Slide number 68

### | Instructor notes

| This slide refers to the modern data architecture, storage layer that the Design Principles and Patterns for Data Pipelines module described. The Ingesting by Batch or by Stream module, which is designed to follow this module, covers AWS Glue features, including crawlers and the data catalog, in more detail. The lab that is associated with that module (Querying Data by Using Athena) introduces the use of Amazon Athena to query Amazon S3.

|

### | Student notes

After the data engineer uploads the files to the S3 buckets, they can run an AWS Glue crawler on the files in the buckets to derive schema information about the files and store it in the AWS Glue Data Catalog. The catalog provides the information that Amazon Athena needs to query the files on Amazon S3. Authorized consumers can use Athena to directly query the data in the data lake or use Amazon Redshift Spectrum to combine data from the customer table in the Redshift data warehouse with data from the data lake files. A bucket policy on each S3 bucket, combined with IAM groups for each regional sales group, limits access to each bucket to the appropriate users and gives them read-only access to the files for their region.

AWS Glue is an integration service to simplify ETL processes. AWS Glue crawlers and the AWS Glue Data Catalog are introduced in the Design Principles and Patterns for Data Pipelines module. The Ingesting by Batch or by Stream module describes AWS Glue features in more detail.

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 by using standard SQL. You will use Athena in the Querying Data by Using Athena lab.

AWS Identity and Access Management (IAM) is the main service for securely controlling access to AWS services. The Securing and Scaling the Data Pipeline module discusses IAM in greater detail.