Copyright Notice

DeepLearning.AI

Source Systems, Data Ingestion, and Pipelines

Week 1

DeepLearning.AI

Source Systems, Data Ingestion, and Pipelines

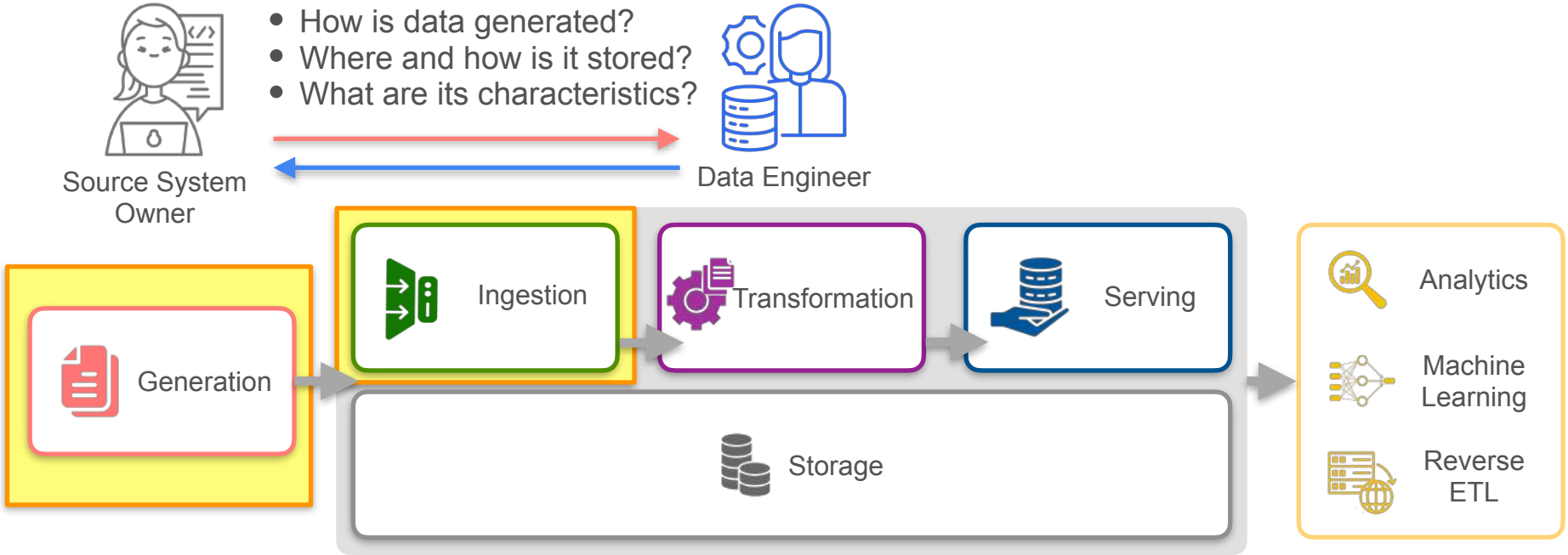
Welcome

DeepLearning.AI

Working with Source Systems

Course 2 Overview

Data Engineering Lifecycle



Course Plan

Week 1 **Common source systems**

- Databases, object storage, and streaming sources
- Working with source systems on AWS

Week 2 **Setting up ingestion from source systems**

Week 3 **DataOps undercurrent**

- Automating some of your pipeline tasks
- Monitoring data quality

Week 4 **Orchestration, monitoring, and automating data pipelines**

- Setting up directed acyclic graphs
- Working with infrastructure as code

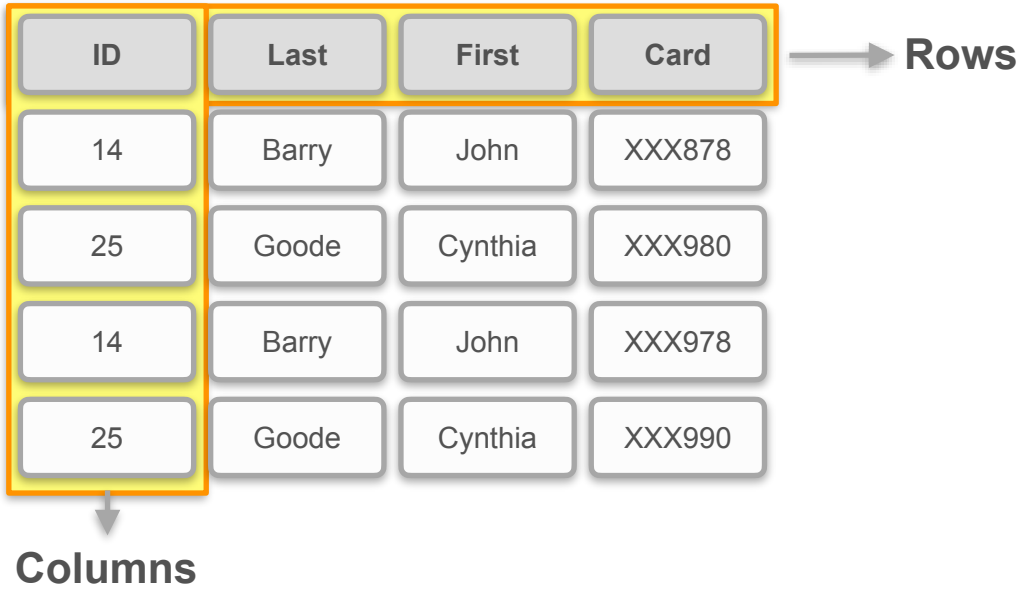
DeepLearning.AI

Introduction to Source Systems

Different Types of Source Systems

Structured Data

Data organized as tables of rows and columns



Video by Adobe Stock (paid license)

Video by Adobe Stock (paid license)

```
import csv
with open('eggs.csv', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
    for row in spamreader:
        print(', '.join(row))
```

Structured Data

Data organized as tables of rows and columns

Semi-Structured Data

Data that is not in tabular form but still has some structure

JavaScript Object Notation (JSON)

A series of key-value pairs

Structured Data

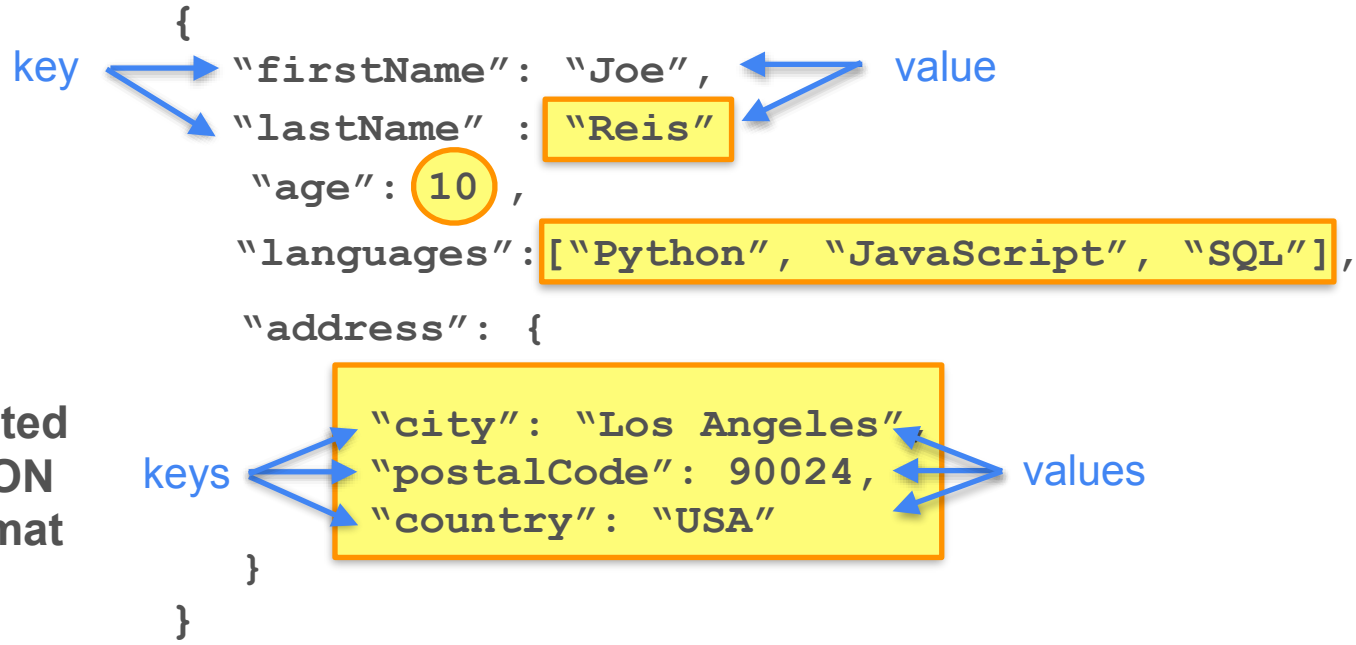
Data organized as tables of rows and columns

Semi-Structured Data

Data that is not in tabular form but still has some structure

JavaScript Object Notation (JSON)

A series of key-value pairs



Nested JSON format

keys

values

Structured Data — Data organized as tables of rows and columns

Semi-Structured Data — Data that is not in tabular form but still has some structure

Unstructured Data — Data that does not have any predefined structure

Text

Video

Audio

Images

- dimensions
- pixel colors

Databases

Structured data
Semi-structured data

Files

Streaming Systems

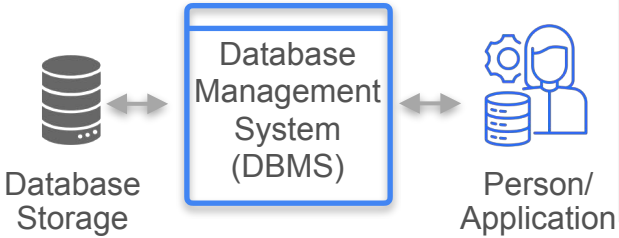
Semi-structured data

Databases

Store data in an organized way

Structured data
Semi-structured data

Create
Read
Update
Delete



Files

Streaming Systems

Semi-structured data

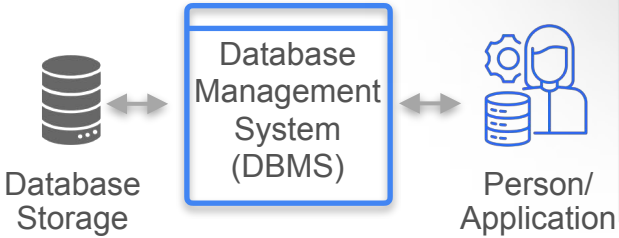
Databases

Store data in an organized way

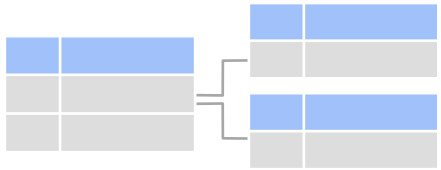
Structured data

Semi-structured data

Create
Read
Update
Delete



Relational databases



Tables with rows and columns

Non-relational (NoSQL) databases

Non-tabular data

Databases

Store data in an organized way

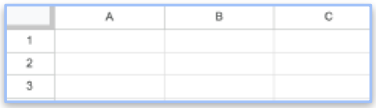
Structured data

Semi-structured data

Create
Read
Update
Delete

Files

Sequence of bytes representing information




```
{
 "firstName": "Joe",
 "lastName" : "Reis",
 "languages":["R", "SQL"],
}
```

Amazon S3

Streaming Systems

Semi-structured data

Databases

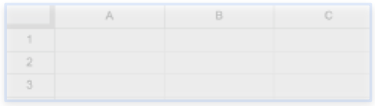
Store data in an organized way

Structured data
Semi-structured data

Create
Read
Update
Delete

Files

Sequence of bytes representing information

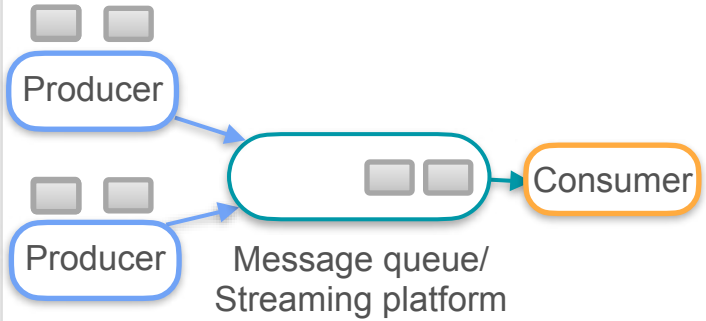


```
{
 "firstName": "Joe",
 "lastName" : "Reis",
 "languages":["R", "SQL"],
}
```


Streaming Systems

Continuous flow of data

Semi-structured data



Databases

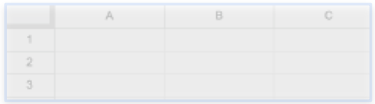
Store data in an organized way

Structured data
Semi-structured data

Create
Read
Update
Delete

Files

Sequence of bytes representing information

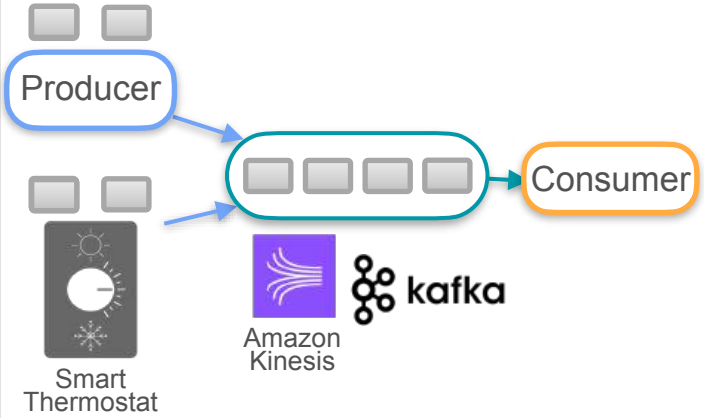


```
{
 "firstName": "Joe",
 "lastName" : "Reis",
 "languages":["R", "SQL"],
}
```


Streaming Systems

Continuous flow of data

Semi-structured data



Databases

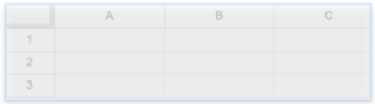
Store data in an organized way

Structured data
Semi-structured data

Create
Read
Update
Delete

Files

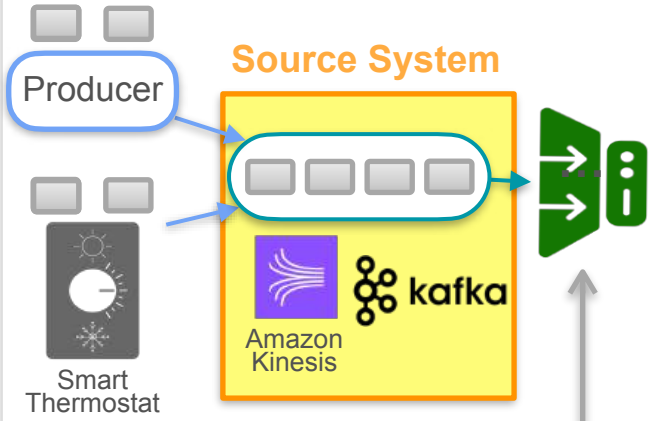
Sequence of bytes representing information



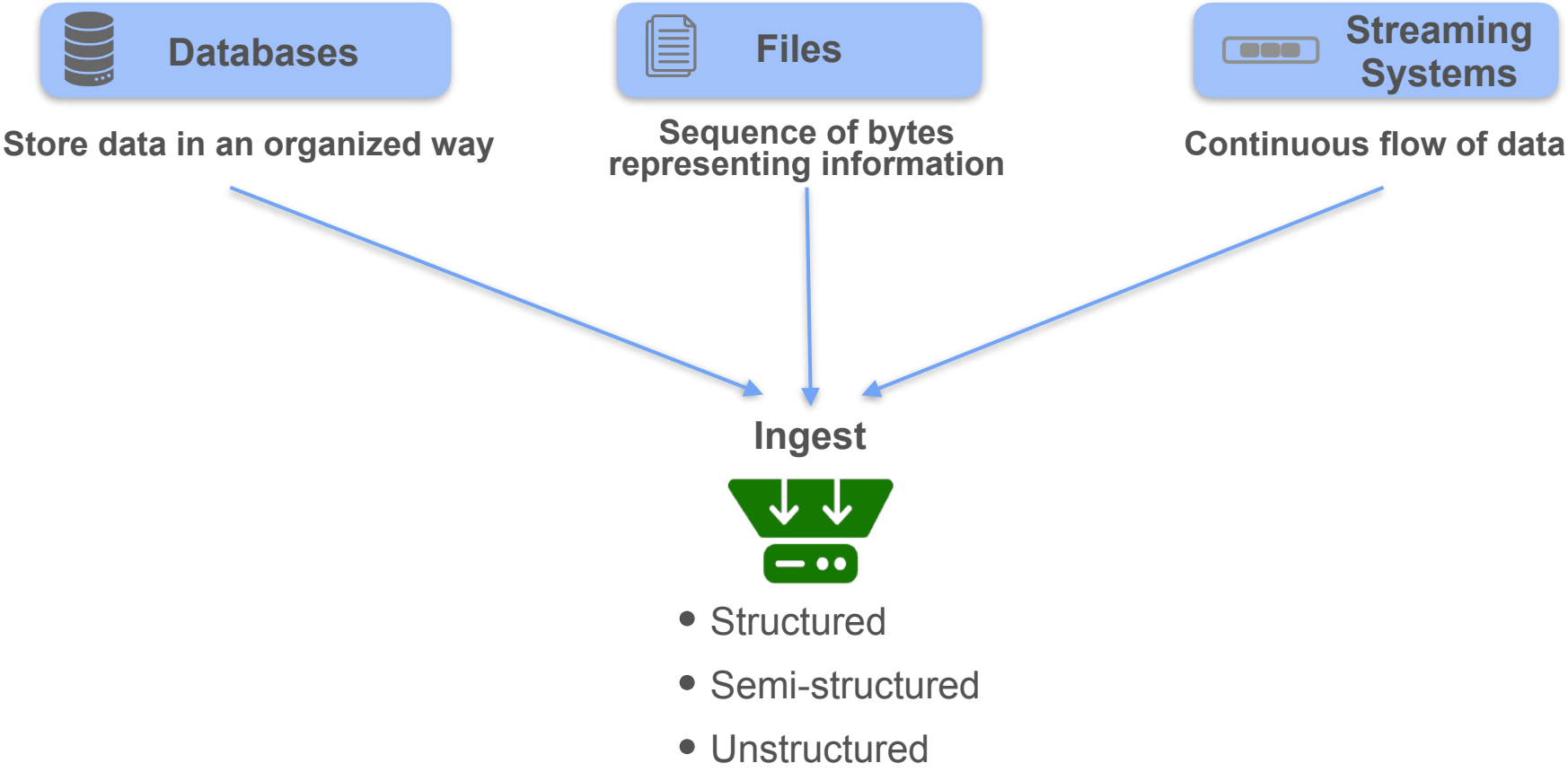
Streaming Systems

Continuous flow of data

Semi-structured data



Your ingestion pipeline starts here

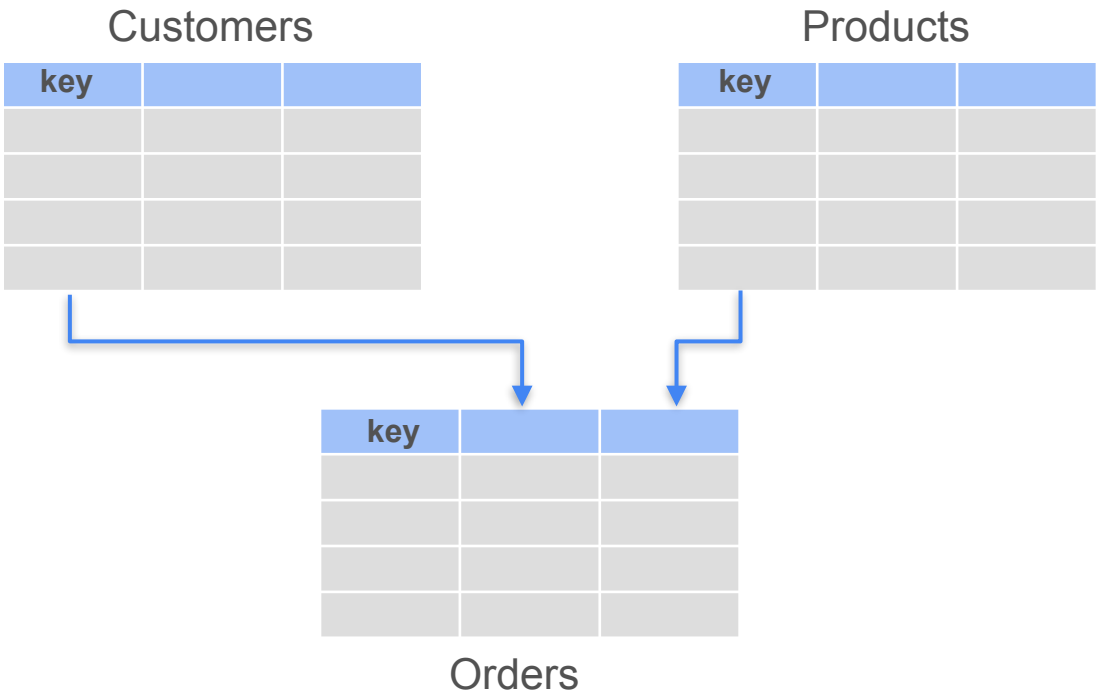

Databases
Store data in an organized way
Files
Sequence of bytes representing information
Streaming Systems
Continuous flow of data
Ingest
Structured
Semi-structured
Unstructured

DeepLearning.AI

Introduction to Source Systems

Relational Databases

Relational Databases



- Reduce redundancy
- Make data easier to manage

Relational Databases

One big table for everything!

name	address	phone	date_time	amount	brand	SKU	description
Jane Doe	74th Street	12345678	12/08/2024	700	ABC	B32	Blender
Jane Doe	74th Street	12345678	12/08/2024	99	XYZ	i56	Iron
Jane Doe	74th Street	12345678	12/08/2024	100	GHJ	k70	Kettle

Jane Doe

Relational Databases

One big table for everything!

name	address	phone	date_time	amount	brand	SKU	description
Jane Doe	74th Street	12345678	12/08/2024	700	ABC	B32	Blender
Jane Doe	74th Street	12345678	12/08/2024	99	XYZ	i56	Iron
Jane Doe	74th Street	12345678	12/08/2024	100	GHJ	k70	Kettle
Mary Ann	19th Avenue	98765432	13/08/2024	899	STU	w40	Washer
John Ken	1st Link	36891623	14/08/2024	899	STU	w40	Washer
Ivy Tan	67th Street	98639513	15/08/2024	899	STU	w40	Washer

Relational Databases

One big table for everything!

Inconsistency

name	address	phone	date_time	amount	brand	SKU	description
Jane Doe	11th Avenue	12345678	12/08/2024	700	ABC	B32	Blender
Jane Doe	11th Avenue	12345678	12/08/2024	99	XYZ	i56	Iron
Jane Doe	74th Street	12345678	12/08/2024	100	GHJ	k70	Kettle
Mary Ann	19th Avenue	98765432	13/08/2024	899	STU	w31	Washer
John Ken	1st Link	36891623	14/08/2024	899	STU	w31	Washer
Ivy Tan	67th Street	98639513	15/08/2024	899	STU	w40	Washer

Jane Doe

now lives on 11th Avenue

SKU

now w31

Inconsistency

Relational Databases

Single customer

Single product

Customers

id	first_name	last_name	age	address
1	Jane	Doe	24	11th Ave.
2	Mary	Ann	65	19th Ave.
3	John	Ken	27	1st Link
4	Ivy	Tan	18	67th St.

Products

id	brand	SKU	description
1	ABC	b32	Blender
2	XYZ	i56	Iron
3	GHJ	k70	Kettle
4	STU	w31	Washer

Orders

id	customer_id	product_id	date_time	purchase_amount

Database schema

Relational Databases

Keys

Primary key:
uniquely identifies each row in a table

Customers

id	first_name	last_name	age	address
1	Jane	Doe	24	11th Ave.
2	Mary	Ann	65	19th Ave.
3	John	Ken	27	1st Link
4	Ivy	Tan	18	67th St.

Products

id	brand	SKU	description
1	ABC	b32	Blender
2	XYZ	i56	Iron
3	GHJ	k70	Kettle
4	STU	w31	Washer

Orders

id	customer_id	product_id	date_time	purchase_amount
1	1	1	12/08/2024	700
2	1	2	12/08/2024	99
3	1	3	12/08/2024	100
4	2	4	13/08/2024	899
5	3	4	14/08/2024	899

Foreign key:
references the primary key of the customer table

Database schema

Relational Databases

Customers

id	first_name	last_name	age	address
1	Jane	Doe	24	11th Ave.
2	Mary	Ann	65	19th Ave.
3	John	Ken	27	1st Link
4	Ivy	Tan	18	67th St.

Products

id	brand	SKU	description
1	ABC	b32	Blender
2	XYZ	i56	Iron
3	GHJ	k70	Kettle
4	STU	w31	Washer

Orders

id	customer_id	product_id	date_time	purchase_amount
1	1	1	12/08/2024	700
2	1	2	12/08/2024	99
3	1	3	12/08/2024	100
4	2	4	13/08/2024	899
5	3	4	14/08/2024	899

Database schema

Each row in a table has to follow the same column structure:
same sequence of columns and data types

Relational Databases

Customers

id	first_name	last_name	age	address
1	Jane	Doe	24	11th Ave.
2	Mary	Ann	65	19th Ave.
3	John	Ken	27	1st Link
4	Ivy	Tan	18	67th St.

Products

id	brand	SKU	description
1	ABC	b32	Blender
2	XYZ	i56	Iron
3	GHJ	k70	Kettle
4	STU	w31	Washer

Orders

id	customer_id	product_id	date_time	purchase_amount
1	1	1	12/08/2024	700
2	1	2	12/08/2024	99
3	1	3	12/08/2024	100
4	2	4	13/08/2024	899
5	3	4	14/08/2024	899
6	1	4	15/08/2024	899

Relational Databases

One big table for everything!

name	address	phone	date_time	amount	brand	SKU	description
Jane Doe	74th Street	12345678	12/08/2024	700	ABC	B32	Blender
Jane Doe	74th Street	12345678	12/08/2024	99	XYZ	i56	Iron
Jane Doe	74th Street	12345678	12/08/2024	100	GHJ	k70	Kettle
Mary Ann	19th Avenue	98765432	13/08/2024	899	STU	w40	Washer
John Ken	1st Link	36891623	14/08/2024	899	STU	w40	Washer
Ivy Tan	67th Street	98639513	15/08/2024	899	STU	w40	Washer

One Big Table (OBT) approach: use cases that need faster processing

Relational Databases

Relational Database Management System (RDBMS)

Software layer that sits on top of a relational database to manage and interact with the data.

MySQL

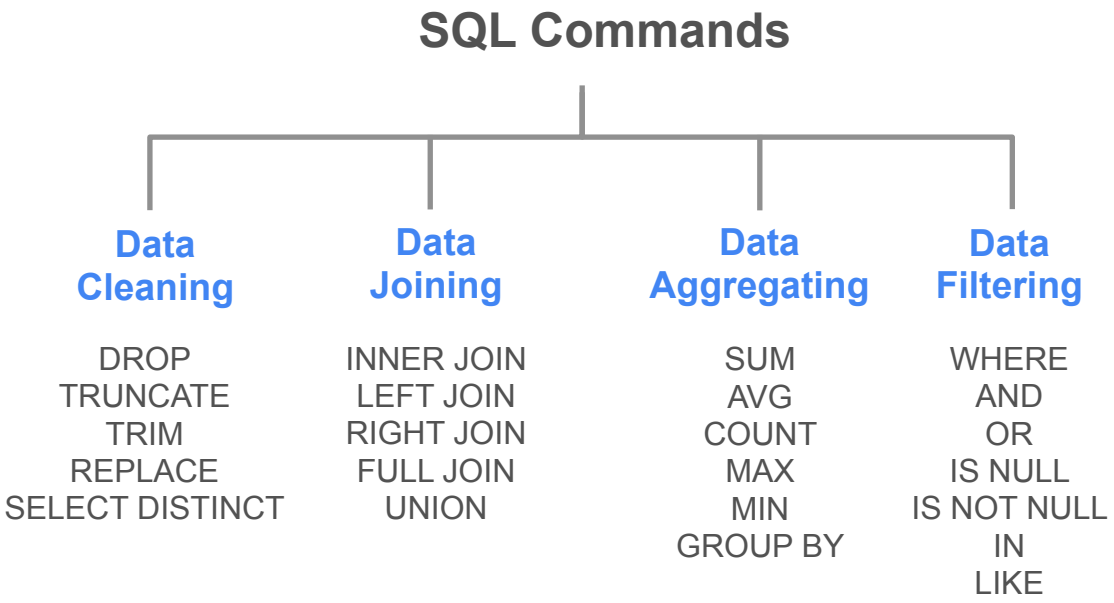
ORACLE DATABASE

PostgreSQL

Microsoft SQL Server

Structured Query Language (SQL)

Relational Databases



DeepLearning.AI

Introduction to Source Systems

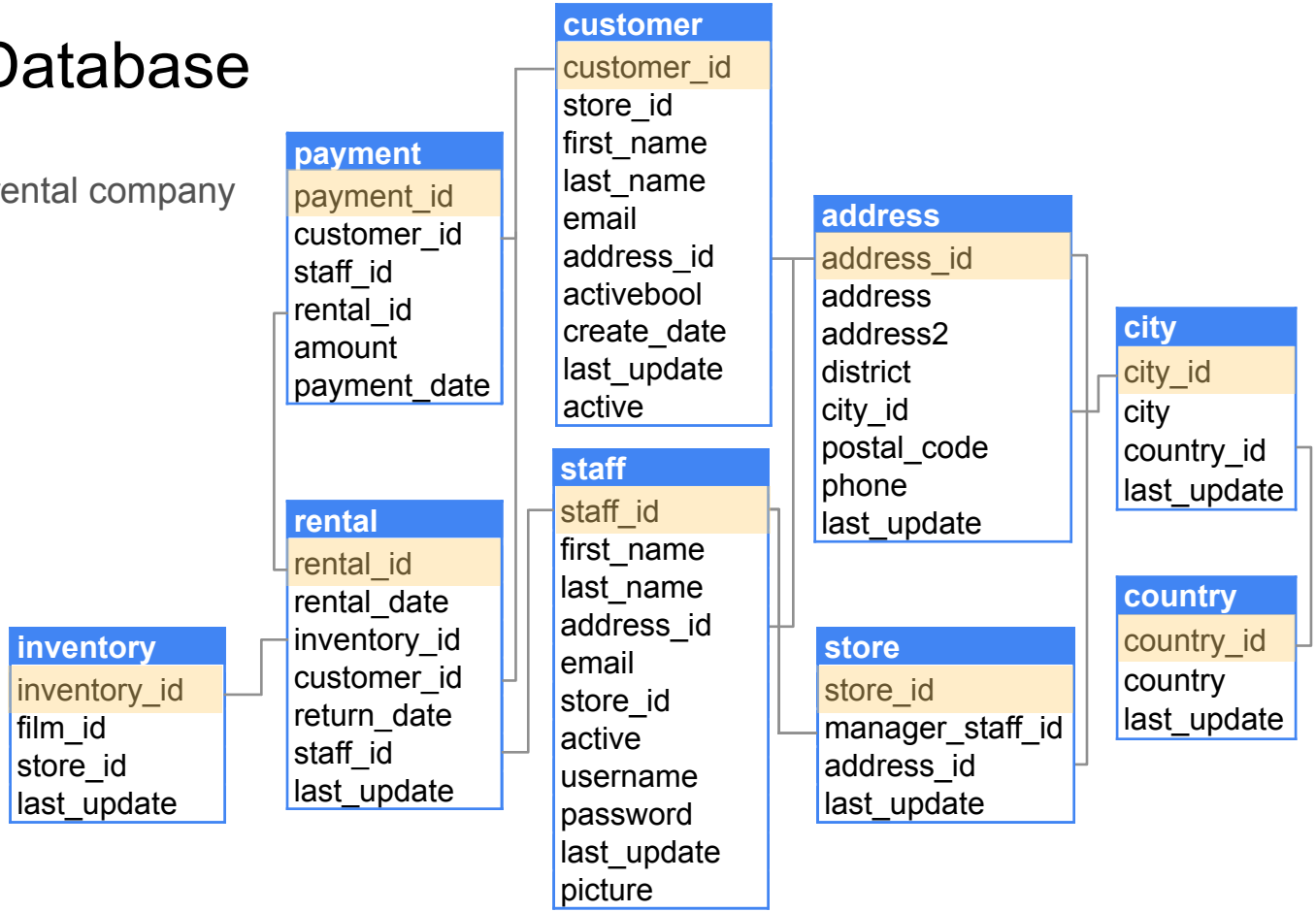
SQL Queries

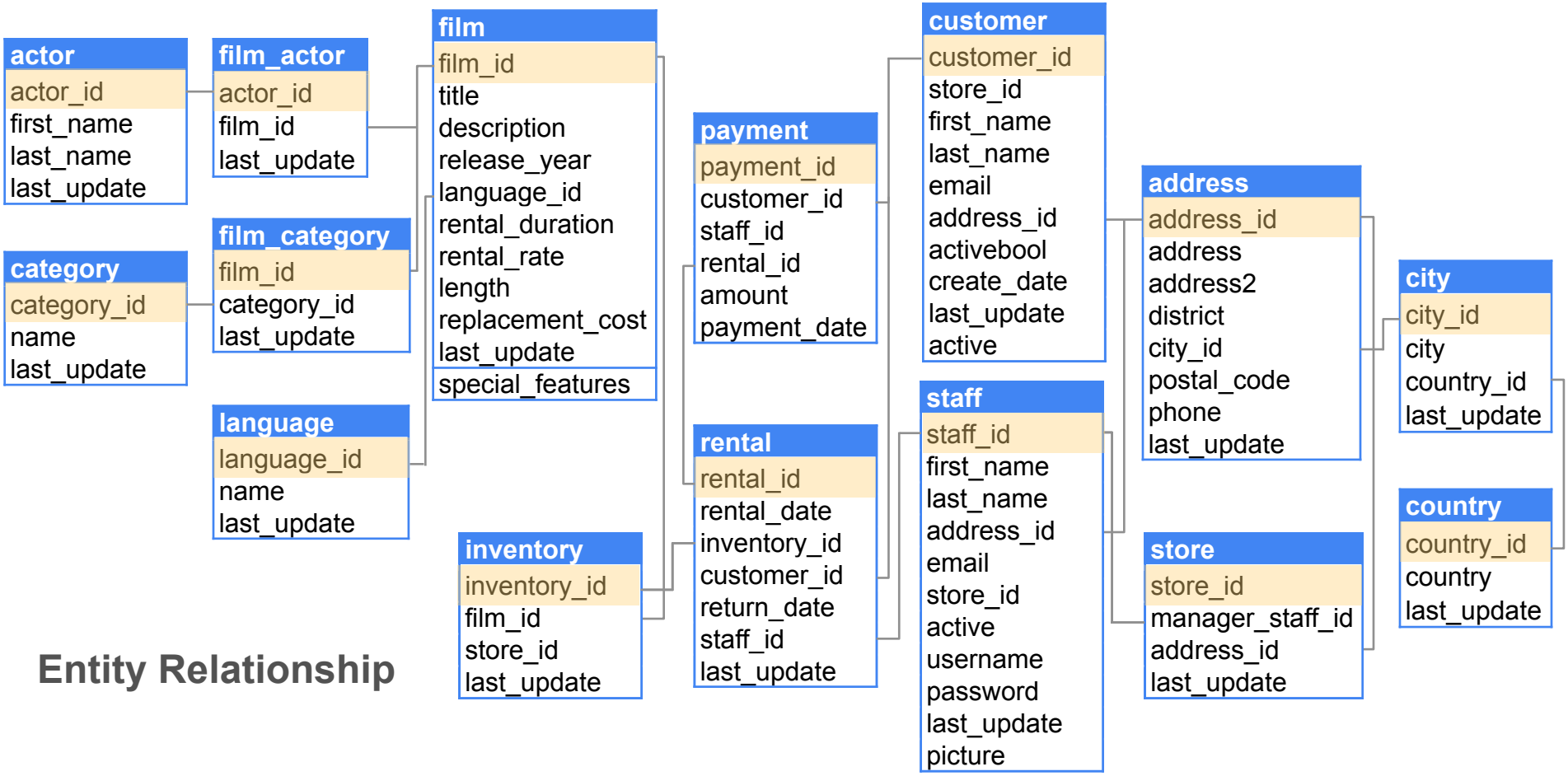
The Relational Database

- Database for a fictitious DVD rental company called **Rentio**
- Database schema

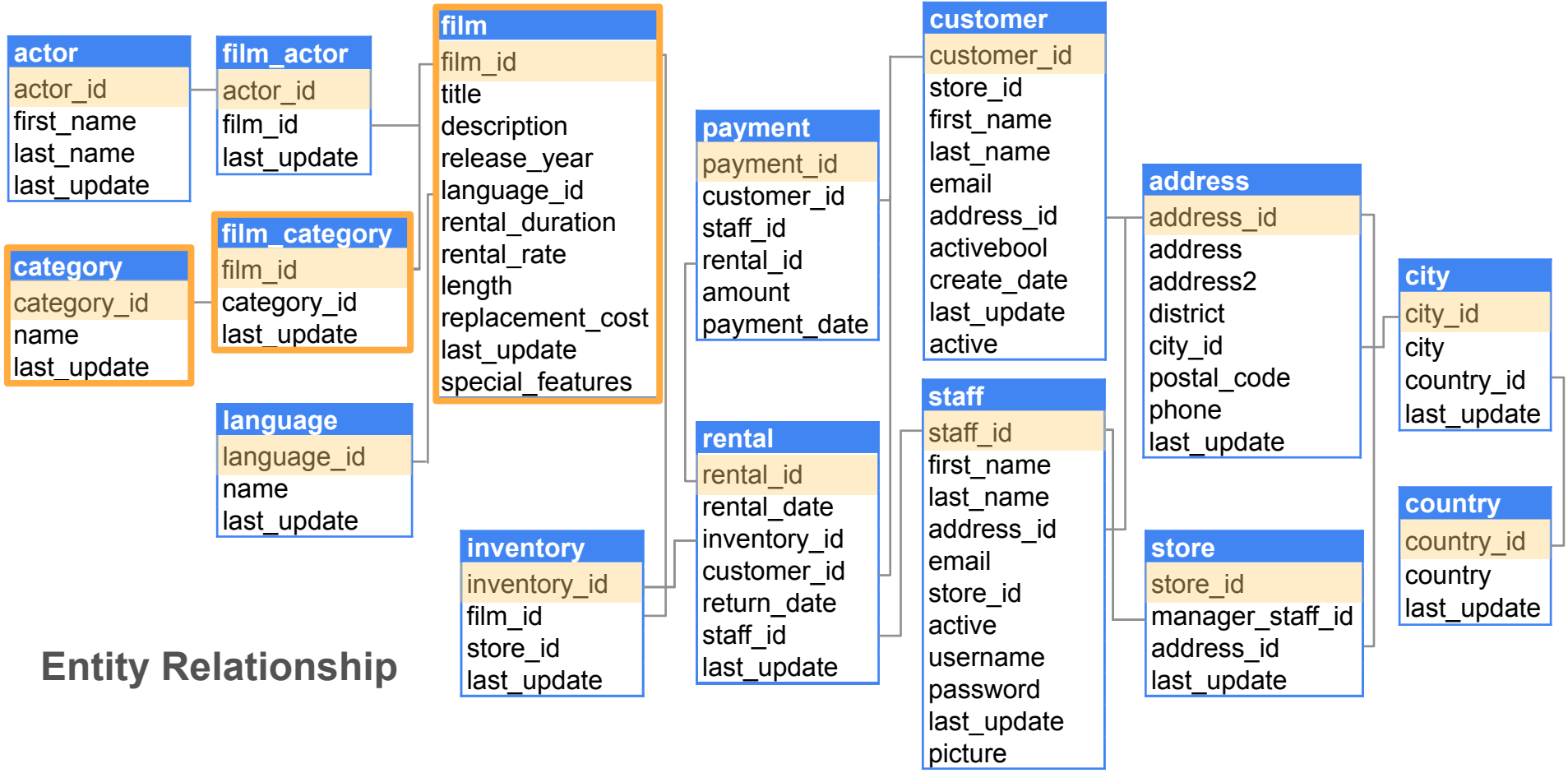
SQL queries

Answer business questions

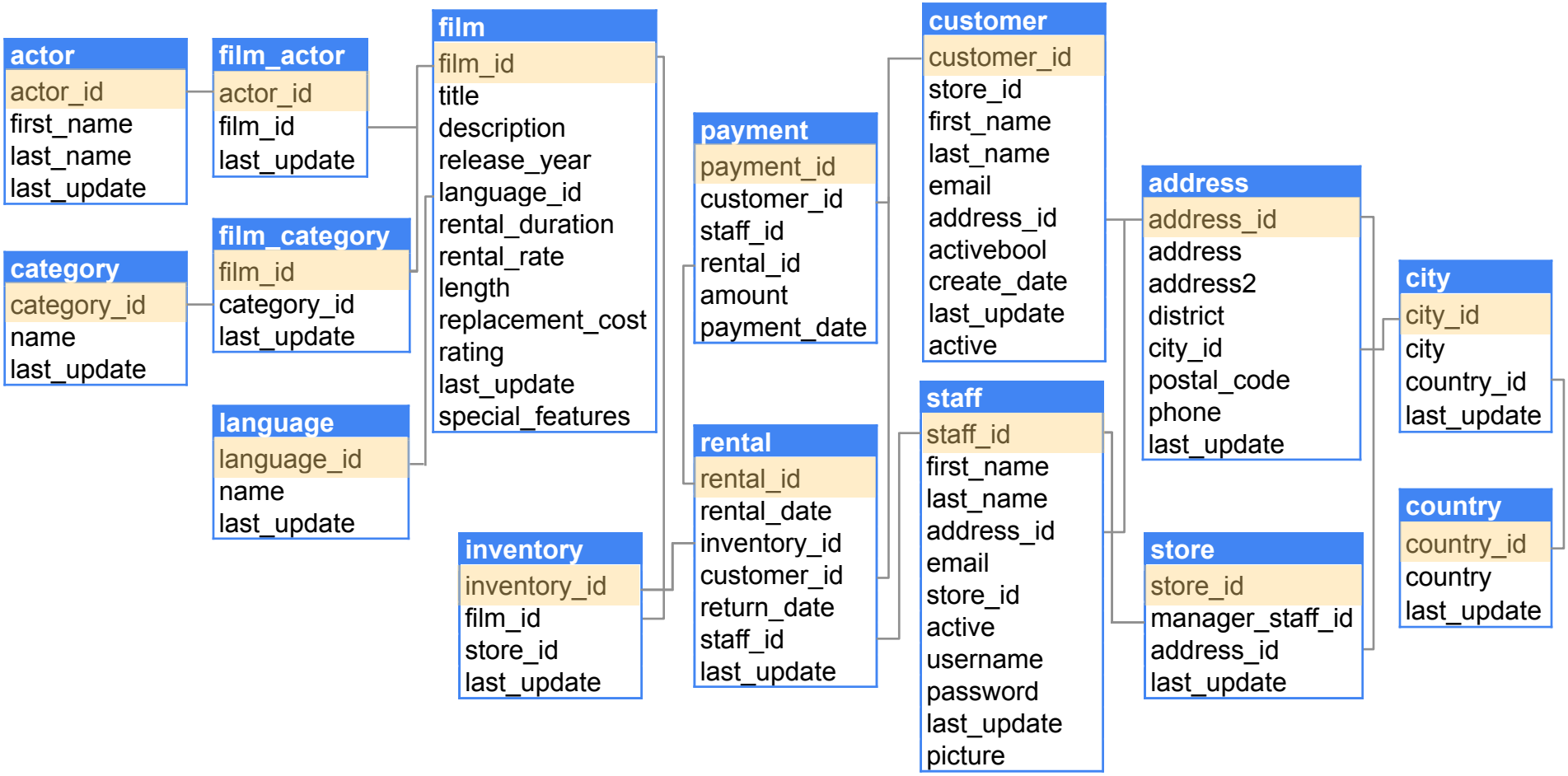


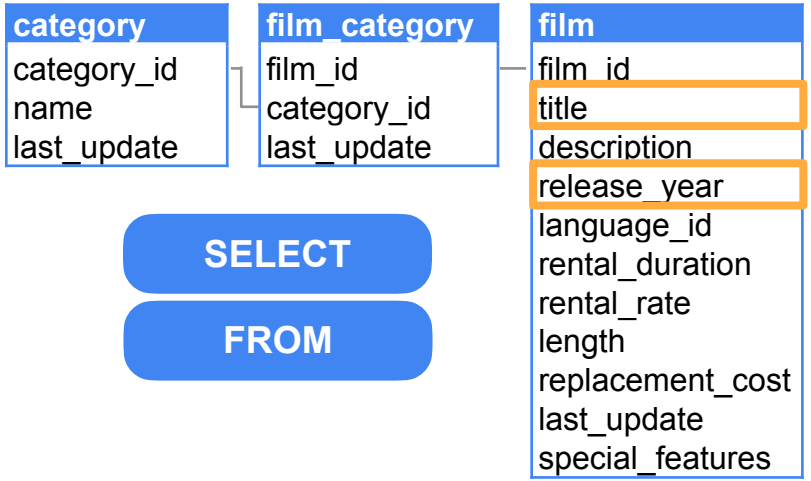


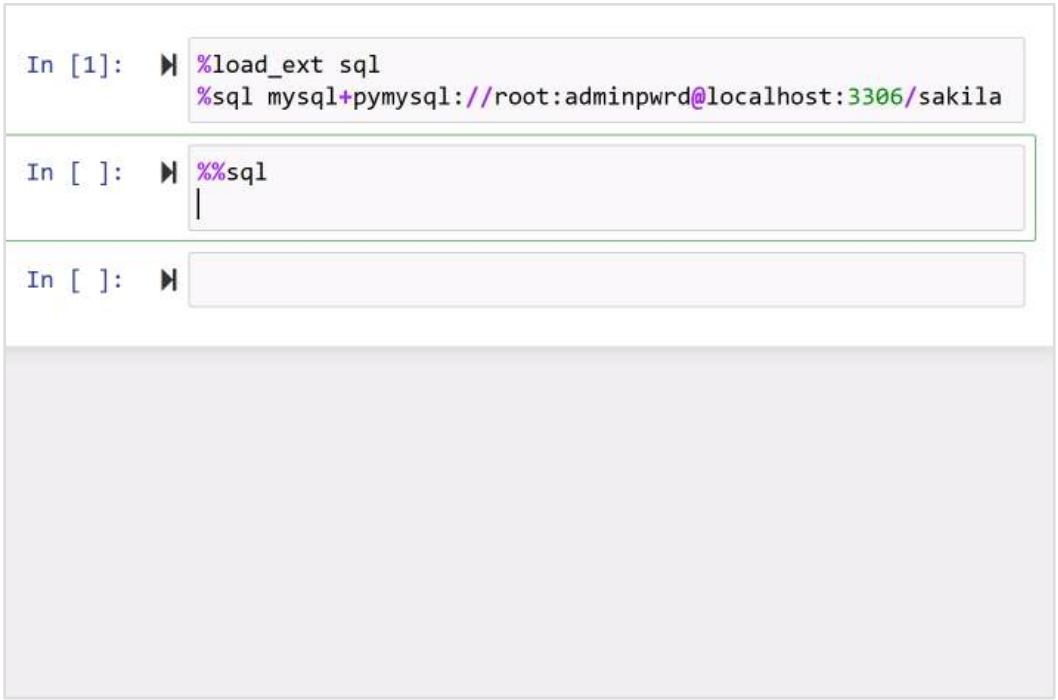
Entity Relationship

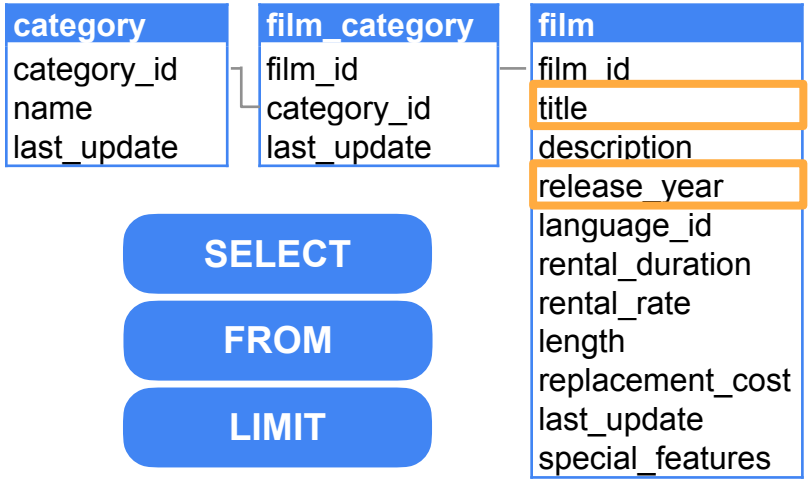


Entity Relationship


actor
actor_id
first_name
last_name
last_update
film_actor
actor_id
film_id
last_update
film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
customer
customer_id
store_id
first_name
last_name
email
address_id
activebool
create_date
last_update
active
payment
payment_id
customer_id
staff_id
rental_id
amount
payment_date
address
address_id
address
address2
district
city_id
postal_code
phone
last_update
city
city_id
city
country_id
last_update
film_category
film_id
category_id
last_update
category
category_id
name
last_update
language
language_id
name
last_update
rental
rental_id
rental_date
inventory_id
customer_id
return_date
staff_id
last_update
staff
staff_id
first_name
last_name
address_id
email
store_id
active
username
password
last_update
picture
inventory
inventory_id
film_id
store_id
last_update
store
store_id
manager_staff_id
address_id
last_update
country
country_id
country
last_update


category
category_id
name
last_update
film_category
film_id
category_id
last_update
film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features
SELECT
FROM


In [1]:
%load_ext sql
%sql mysql+pymysql://root:adminpwrd@localhost:3306/sakila
In [ ]:
%%sql
In [ ]:

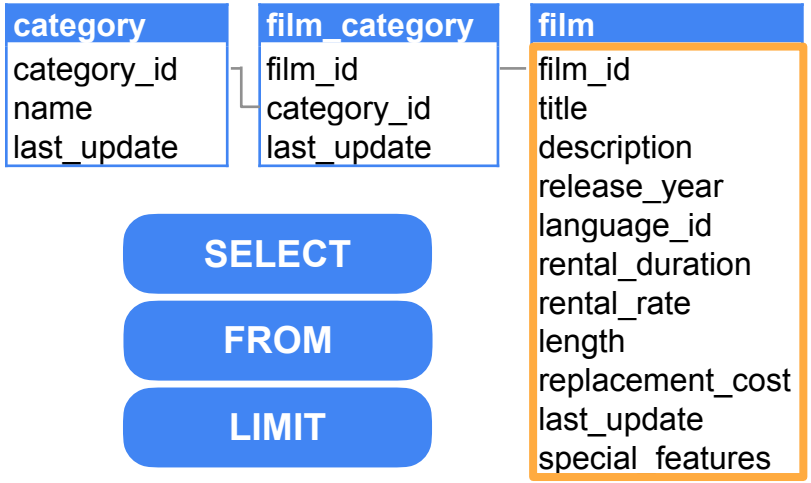


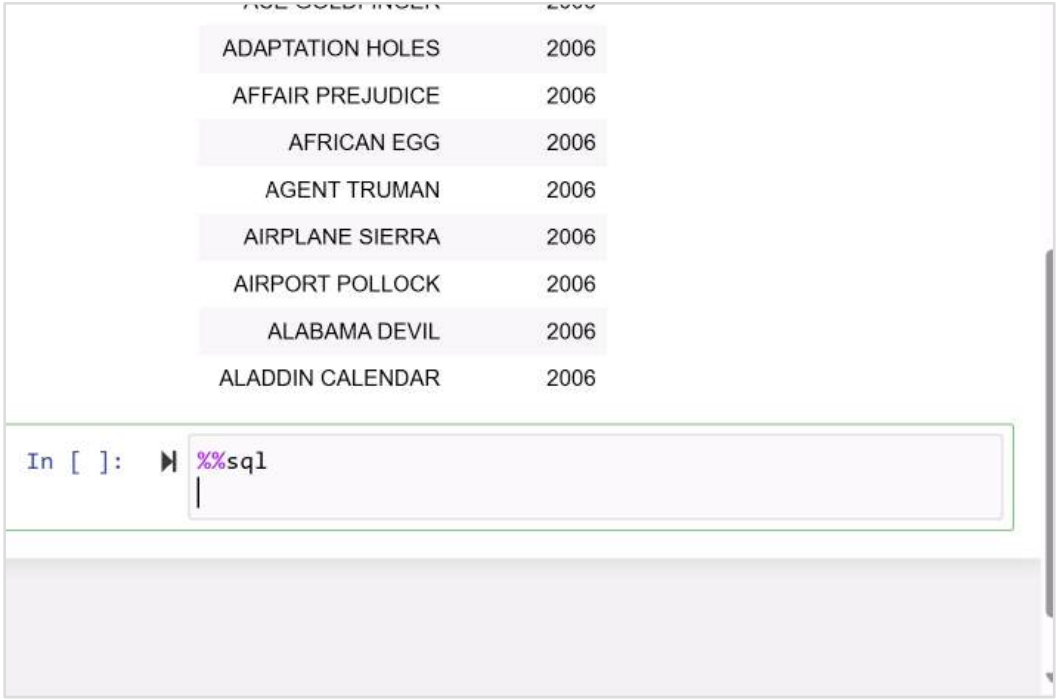

```
In [1]: %load_ext sql
        %sql mysql+pymysql://root:adminpwrd@localhost:3306/sakila

In [2]: %%sql
        SELECT title, release_year
        FROM film
```

```
 * mysql+pymysql://root:***@localhost:3306/sakila
1000 rows affected.
```

Out[2]:

title	release_year
ACADEMY DINOSAUR	2006
ACE GOLDFINGER	2006
ADAPTATION HOLES	2006
AFFAIR PREJUDICE	2006
AFRICAN EGG	2006
AGENT TRUMAN	2006


category
category_id
name
last_update
film_category
film_id
category_id
last_update
film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features
SELECT
FROM
LIMIT


ADAPTATION HOLES 2006
AFFAIR PREJUDICE 2006
AFRICAN EGG 2006
AGENT TRUMAN 2006
AIRPLANE SIERRA 2006
AIRPORT POLLOCK 2006
ALABAMA DEVIL 2006
ALADDIN CALENDAR 2006
In [ ]:
%%sql

category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

SELECT

FROM

WHERE

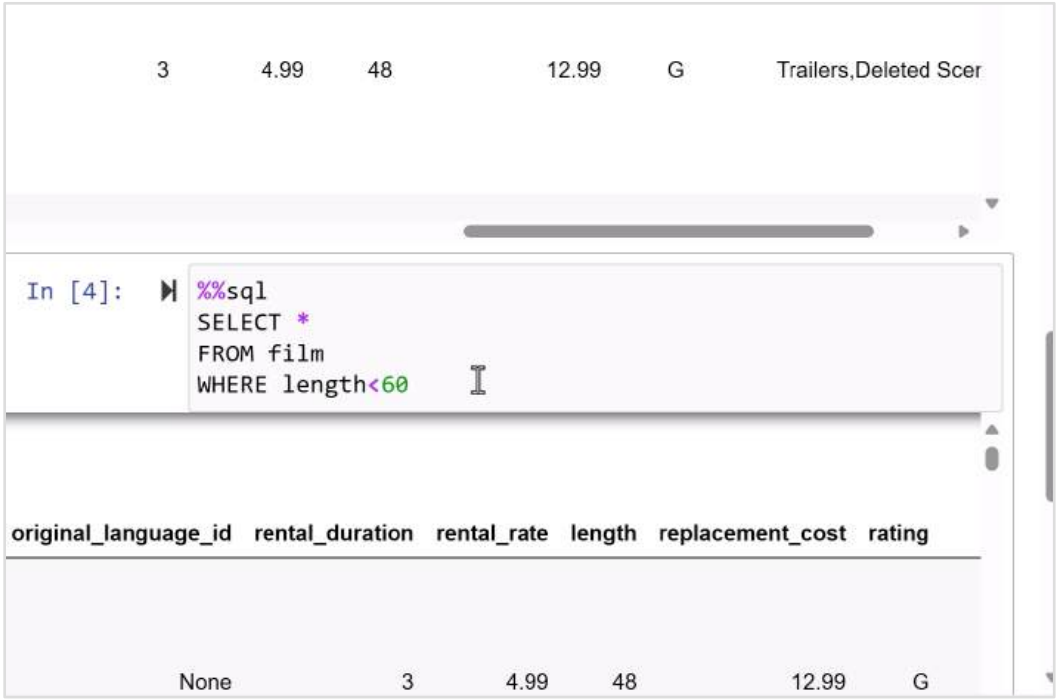
Exploring the films that are less than 60 minutes long.

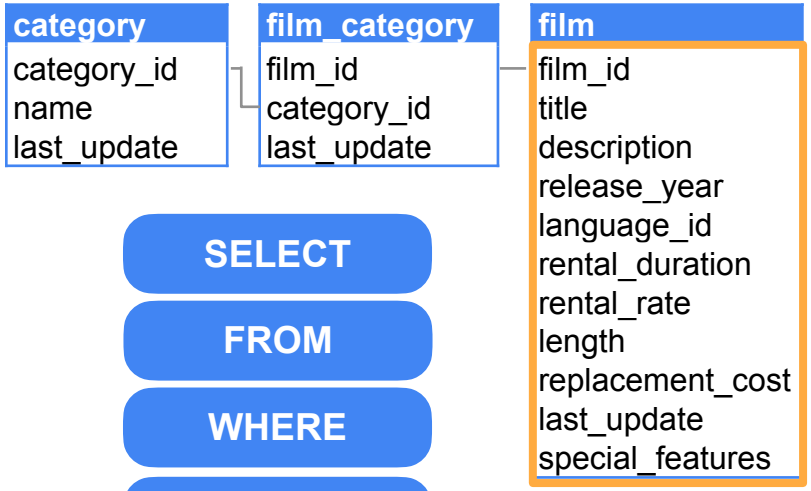
category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- FROM
- WHERE
- ORDER BY

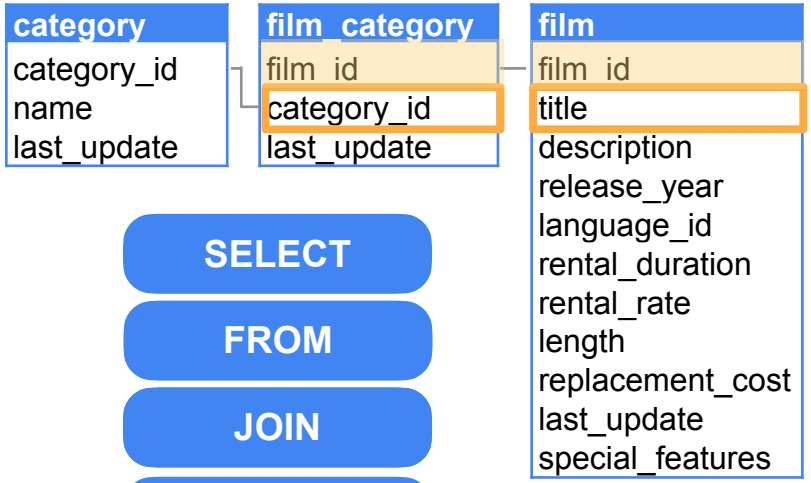




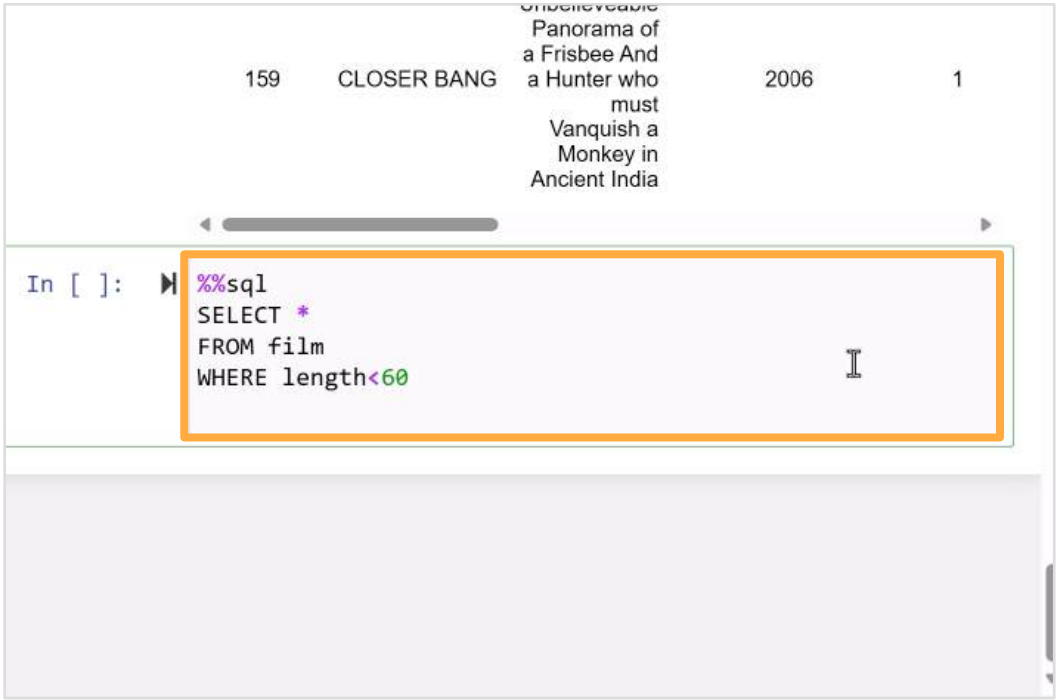
- SELECT
- FROM
- WHERE
- ORDER BY
- LIMIT

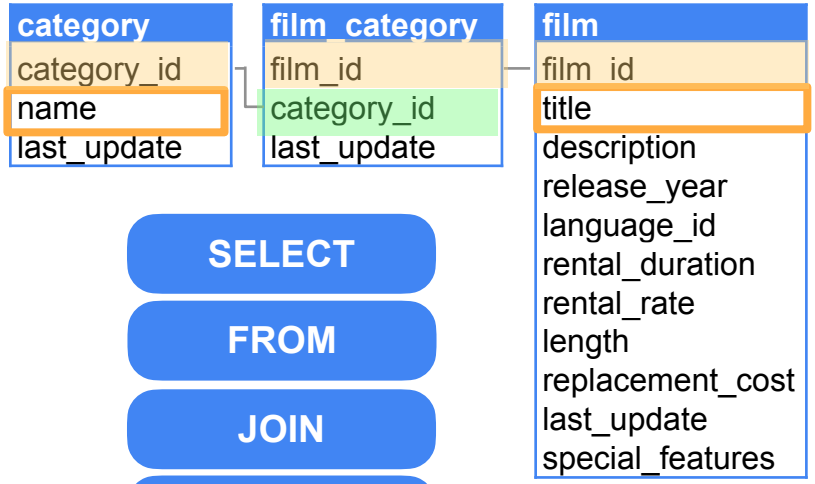
```
                3        4.99        48              12.99      G        Trailers,Deleted Scer

In [5]:  %%sql
         SELECT *
         FROM film
         WHERE length<60
         ORDER BY length
```

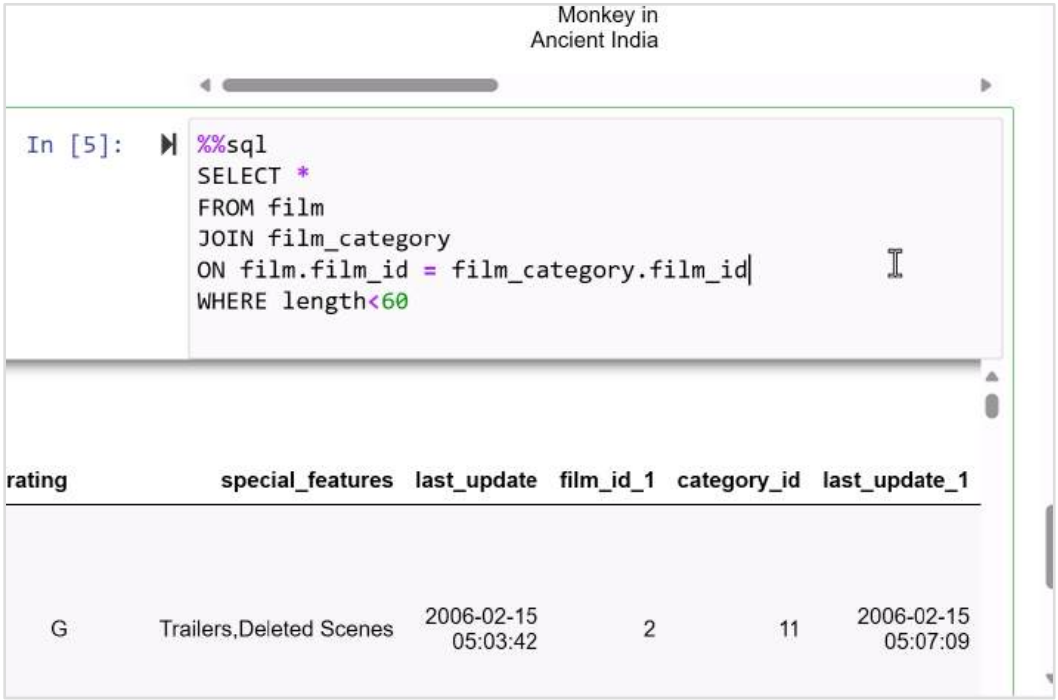
original_language_id	rental_duration	rental_rate	length	replacement_cost	rating



- SELECT
- FROM
- JOIN
- ORDER BY
- LIMIT




category
category_id
name
last_update
film_category
film_id
category_id
last_update
film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features
SELECT
FROM
JOIN
WHERE
ORDER BY
LIMIT

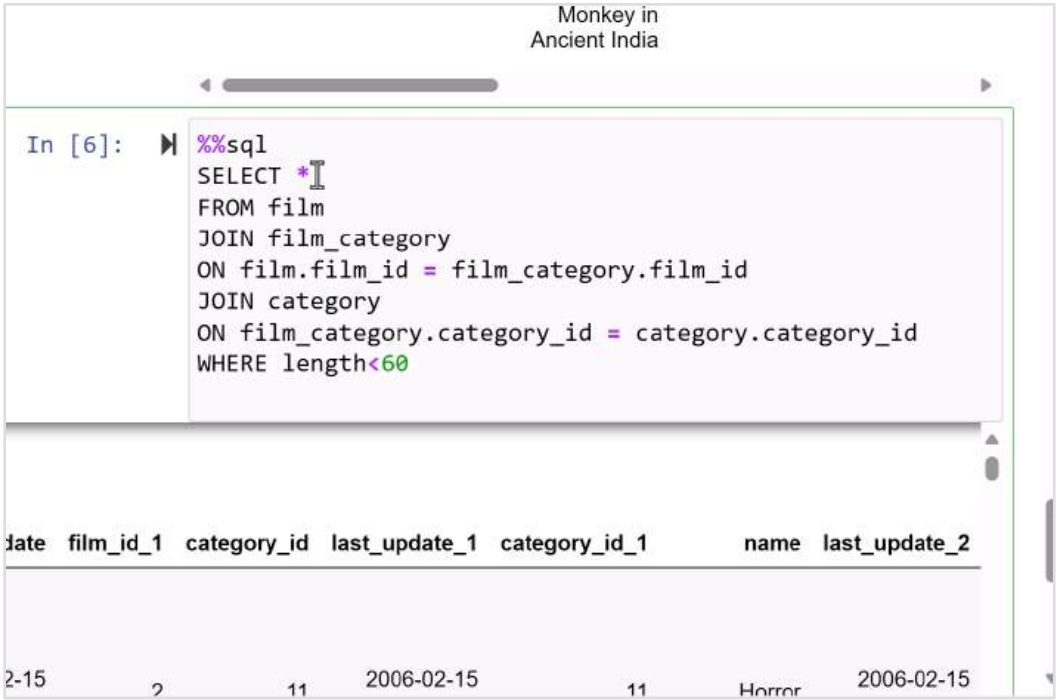

Monkey in
Ancient India
In [5]:
%%sql
SELECT *
FROM film
JOIN film_category
ON film.film_id = film_category.film_id
WHERE length<60
rating
special_features
last_update
film_id_1
category_id
last_update_1
G
Trailers,Deleted Scenes
2006-02-15 05:03:42
2
11
2006-02-15 05:07:09

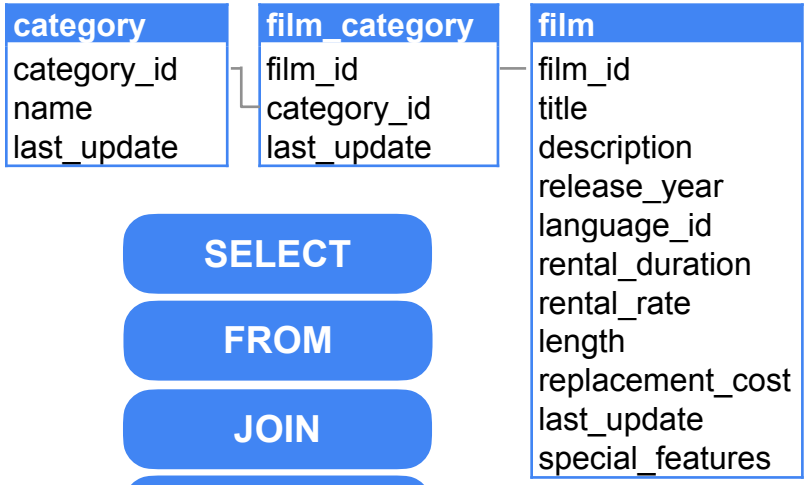
category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- FROM
- JOIN
- WHERE
- ORDER BY
- LIMIT





INNER JOIN

JOIN: combine the records from both tables that have a matching column value specified in the ON statement.

film has a row with film_id = 123
film_category does not have a row with film_id= 123

The row with film_id = 123 will not be in the join results

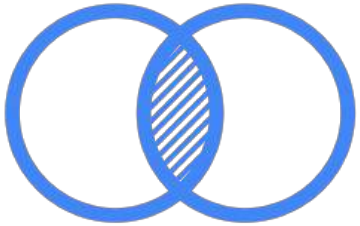
category
category_id
name
last_update

film_category
film_id
category_id
last_update

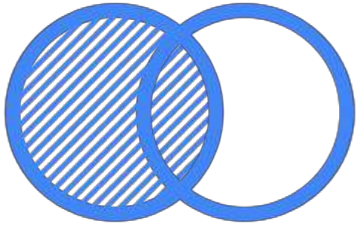
film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- FROM
- JOIN
- WHERE
- ORDER BY
- LIMIT

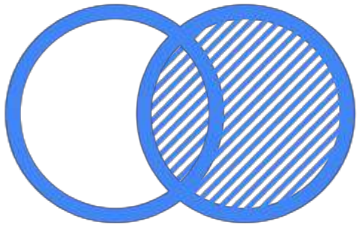
INNER JOIN



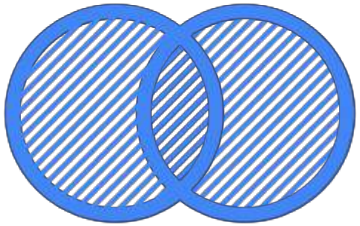
LEFT JOIN



RIGHT JOIN



FULL JOIN



category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- FROM
- JOIN
- WHERE
- GROUP BY
- LIMIT

Monkey in
Ancient India

In [7]:
```
%%sql
SELECT film.title, category.name
FROM film
JOIN film_category
ON film.film_id = film_category.film_id
JOIN category
ON film_category.category_id = category.category_id
WHERE length<60
```

Out[7]:

title	name
ACE GOLDFINGER	Horror
ADAPTATION HOLES	Documentary
AIRPORT POLLOCK	Horror
ALIEN CENTER	Foreign
ALTER VICTORY	Animation

category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- COUNT
- JOIN
- WHERE
- GROUP BY
- ORDER BY
- LIMIT

Monkey in
Ancient India

In [7]:
```
%%sql
SELECT film.title, category.name
FROM film
JOIN film_category
ON film.film_id = film_category.film_id
JOIN category
ON film_category.category_id = category.category_id
WHERE length<60
```

Out[7]:

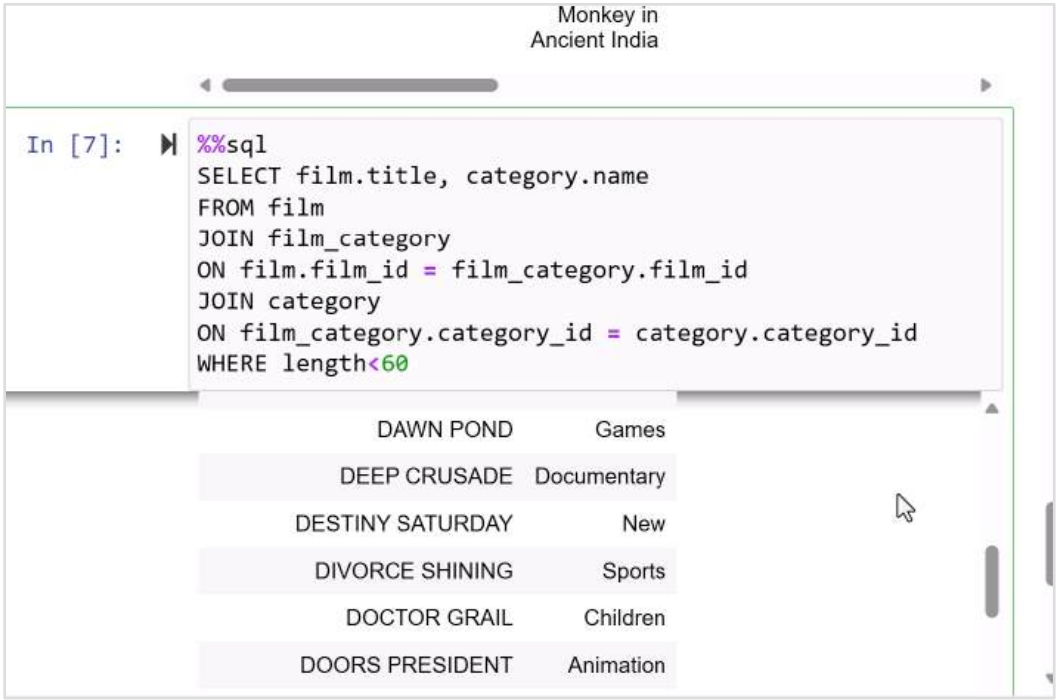
title	name
ACE GOLDFINGER	Horror
ADAPTATION HOLES	Documentary
AIRPORT POLLOCK	Horror
ALIEN CENTER	Foreign
ALTER VICTORY	Animation

category
category_id
name
last_update

film_category
film_id
category_id
last_update

film
film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
last_update
special_features

- SELECT
- COUNT
- FROM
- JOIN
- WHERE
- GROUP BY
- ORDER BY
- LIMIT



Common SQL Commands

- SELECT
- COUNT
- FROM
- JOIN
- WHERE
- GROUP BY
- ORDER BY
- LIMIT

Data Manipulation Operations

- CREATE
- INSERT INTO
- UPDATE
- DELETE

DeepLearning.AI

Introduction to Source Systems

NoSQL Databases

NoSQL Databases

NoSQL

NoSQL Databases

No SQL

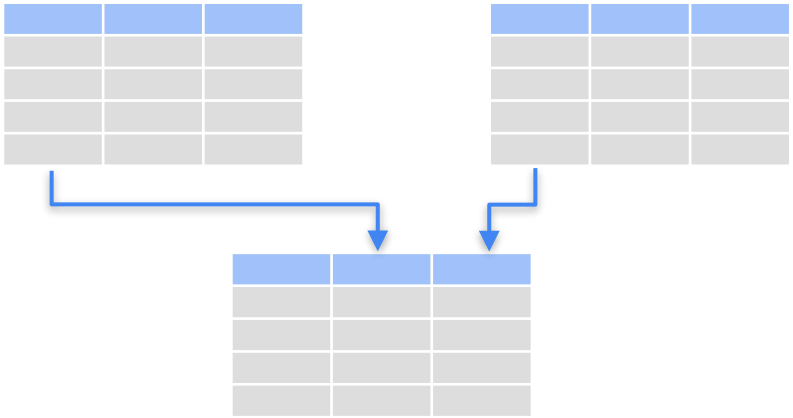
NoSQL Databases

Not Only SQL

Non-Relational Databases

It can still support SQL or SQL-like query languages.

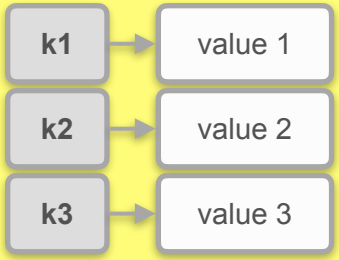
Relational Databases



NoSQL Databases

Non-tabular structures

Key-Value

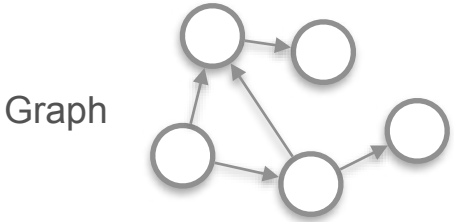


Document

```
{
  "firstName": "Joe",
  "lastName" : "Reis",
  "age": 10,
  "address": {
     "city": "Los Angeles",
     "postalCode": 90024,
     "country": "USA"
  }
}
```

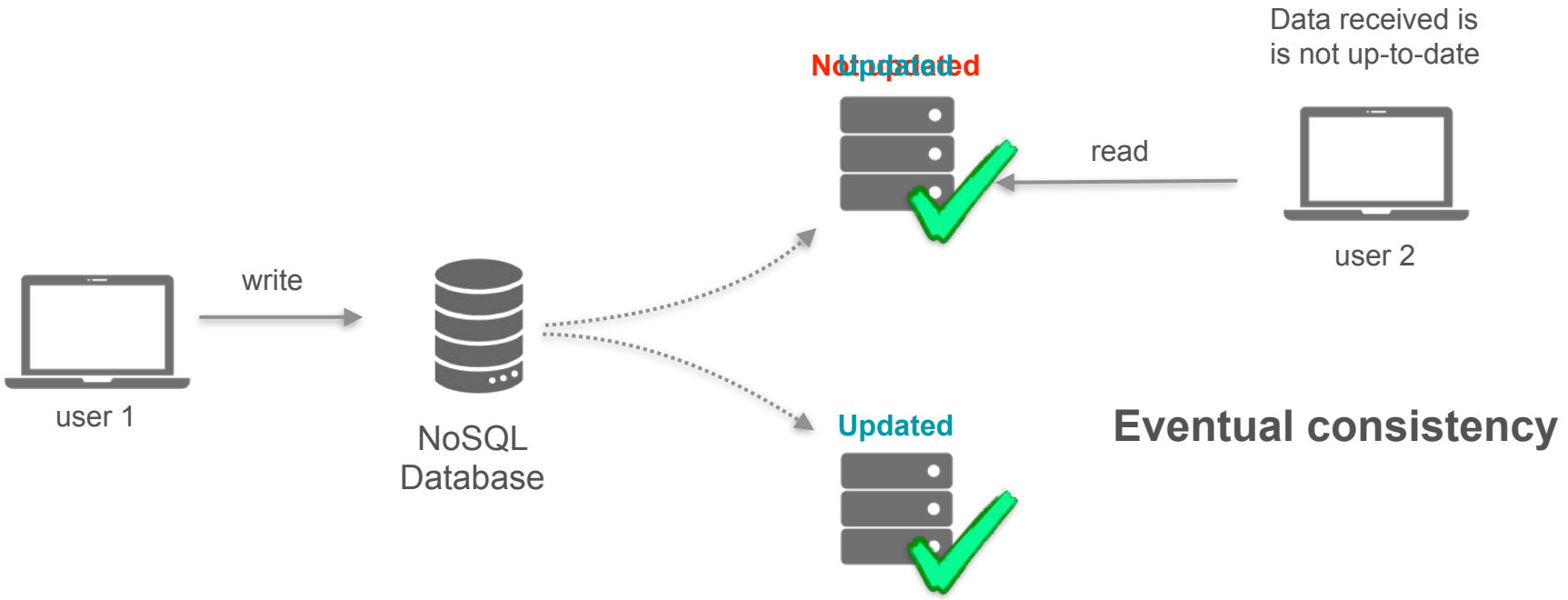
Wide-Column

Graph



- No predefined schemas
- More flexibility when storing your data

Horizontal Scaling



Consistency

NoSQL Databases	Relational Databases
Eventual Consistency	Strong Consistency
• Speed is prioritized • System availability and scalability are important	• Read data only when all nodes have been updated

NoSQL Databases

Not all NoSQL databases guarantee:

ACID compliance

Atomicity

Consistency

Isolation

Durability

Specialized Query Language

Example of NoSQL Data

```
{
  "id": 1,
  "key": "Blender",
  "qty": 6,
  "sku": "b32"
}
```

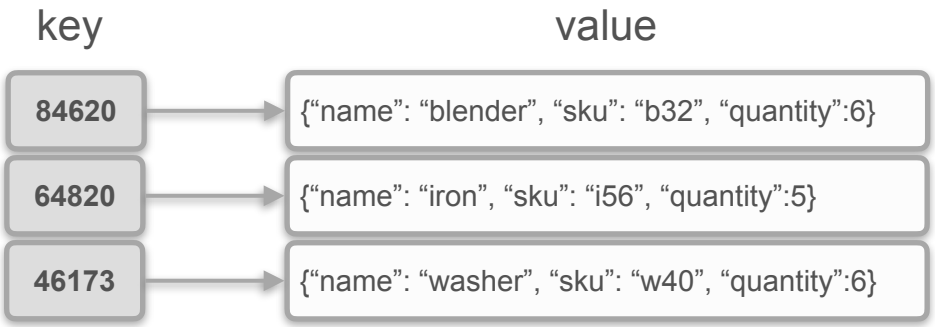
Query

```
db.products.find({qty: {$gt: 4}})
```

Ref: AWS docs

Types of No-SQL Databases

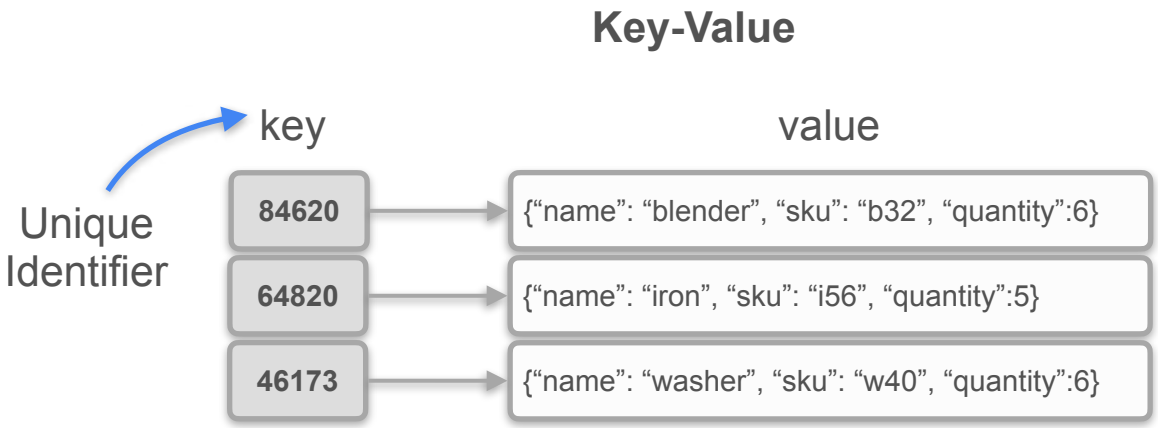
Key-Value



Document

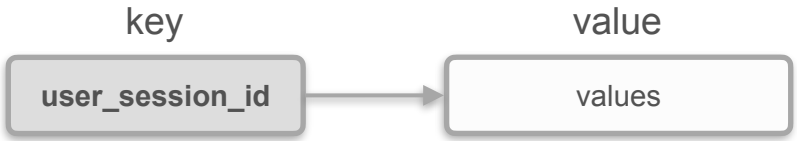
```
{
  “firstName”: “Joe”,
  “lastName” : “Reis”,
  “age”: 10,
  “address”: {
     “city”: “Los Angeles”,
     “postalCode”: 90024,
     “country”: “USA”
  }
}
```

Key-Value Database



Fast lookup: such as caching user session data

- viewing different products
- adding items to the shopping cart
- checking out



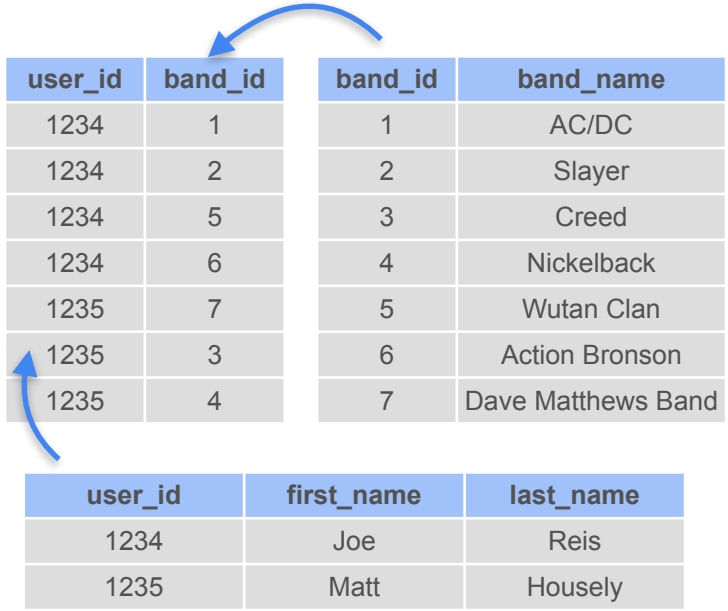
Document Database

Collection *(Like a table)*

Document Database

```
{
 "users" : [
         {
          "id": 1234,
          "name": {
                   "first":"Joe",
                   "last":"Reis"
                   },
          "favorite_bands" : ["AC/DC", "Slayer", "WuTang Clan", "Action Bronson" ]
          },

          {
           "id":1235,
           "name": {
                    "first": "Matt",
                    "last":"Housley"
                   },
           "favorite_bands" : ["Dave Matthews Band", "Creed", "Nickelback"]
          }
       ]
}
```

- Easy to retrieve all the information about a user (locality)
- Document stores don't support joins
- Flexible schema



user_id	band_id
1234	1
1234	2
1234	5
1234	6
1235	7
1235	3
1235	4

band_id	band_name
1	AC/DC
2	Slayer
3	Creed
4	Nickelback
5	Wutan Clan
6	Action Bronson
7	Dave Matthews Band

user_id	first_name	last_name
1234	Joe	Reis
1235	Matt	Housely

Fixed schema

Document Database

Use cases

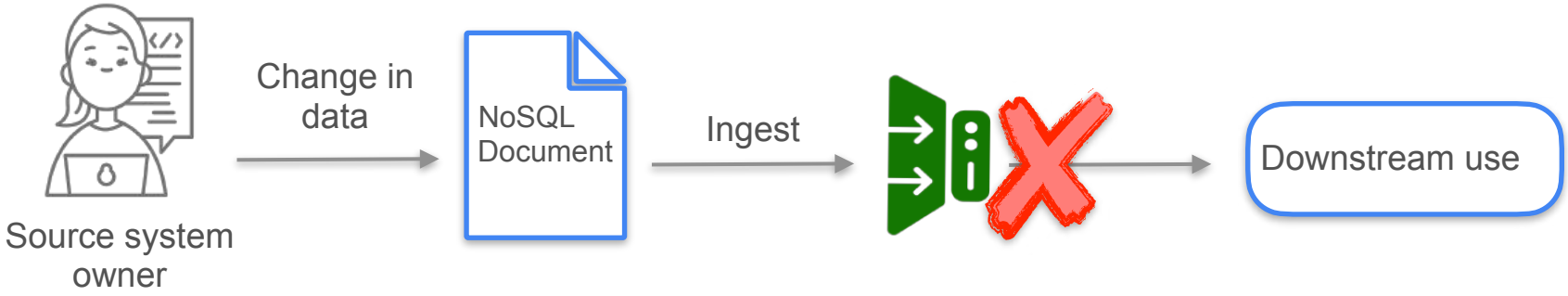
- Content management
- Catalogs
- Sensor readings

```
{
 "iot" : [
          {
           "id": 24,
           "interaction": "some_interaction",
           "device": "my_device",
           "sensor_reading": 34
           }
          ]
}
```

Flexible Schema

Document Database

Document databases become absolute nightmares to manage and query.



DeepLearning.AI

Introduction to Source Systems

Database ACID Compliance

OLTP Systems

OLTP

Online Transaction Processing

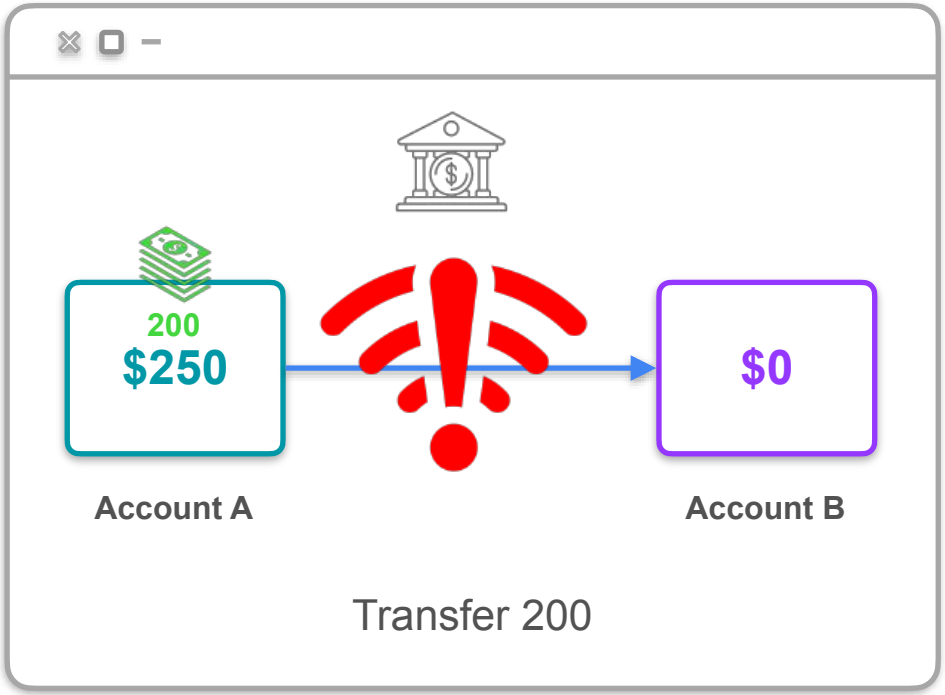
Support very high transaction rates (bank account balances, online orders)

ACID Compliance

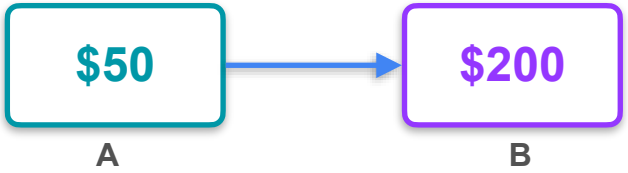
Relational Databases	NoSQL Databases
ACID compliant **A**tomicity **C**onsistency **I**solation **D**urability	Not ACID compliant by default

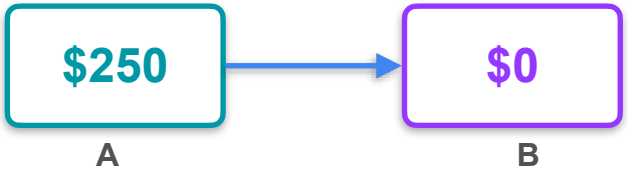
They help ensure transactions are processed reliably and accurately in an OLTP system.

ACID Compliance

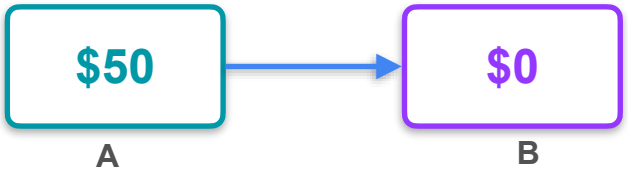


You'd be hoping



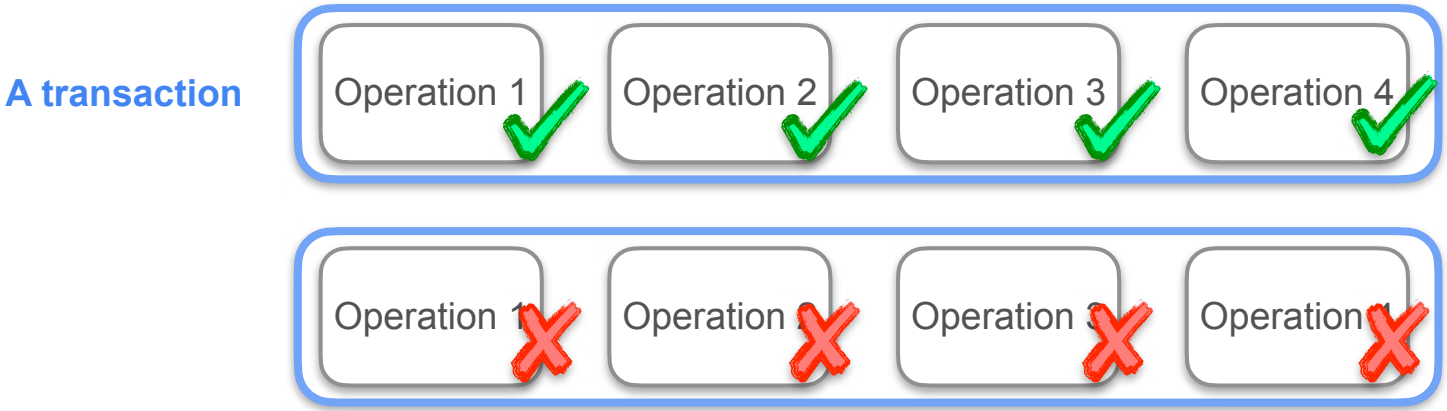


But not



Atomicity

Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.



Atomicity

Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

A transaction: placing an order

Both operations must happen as a single transaction

Atomicity Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

Consistency Any changes to the data made within a transaction follow the set of rules or constraints defined by the database schema.

id	product_name	quantity
1	blender	1

Buy 2 blenders

Transaction

id	product_name	quantity
1	blender	-1

Rule: stock level ≥ 0

Atomicity Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

Consistency Any changes to the data made within a transaction follow the set of rules or constraints defined by the database schema.

ACID compliance

Atomicity

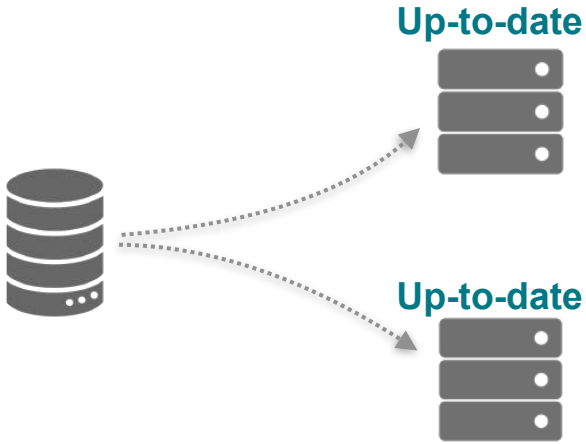
Consistency

Isolation

Durability

Strong Consistency

All nodes provide the same up-to-date



Atomicity Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

Consistency Any changes to the data made within a transaction follow the set of rules or constraints defined by the database schema.

Isolation Each transaction is executed independently in sequential order.

id	product_name	quantity
1	blender	5

Transaction

Buy 5 blenders

Transaction

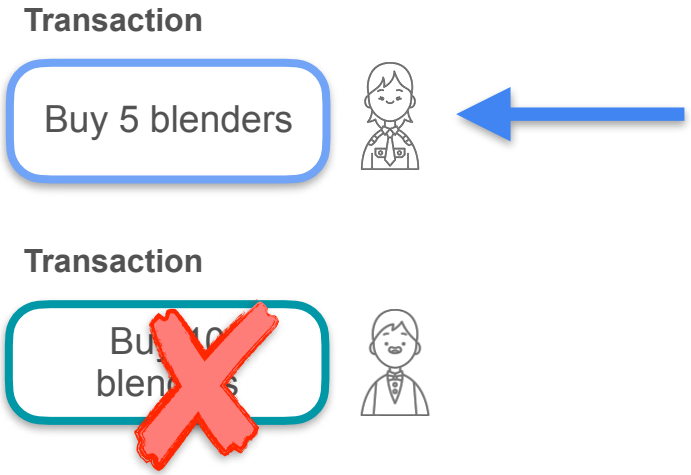
Buy 5 blenders

Atomicity Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

Consistency Any changes to the data made within a transaction follow the set of rules or constraints defined by the database schema.

Isolation Each transaction is executed independently in sequential order.

id	product_name	quantity
1	blender	5



Atomicity Atomicity ensures that transactions are **atomic**, treated as a single, indivisible unit.

Consistency Any changes to the data made within a transaction follow the set of rules or constraints defined by the database schema.

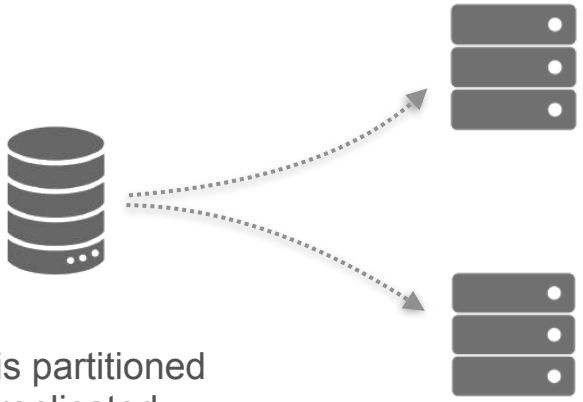
Isolation Each transaction is executed independently in sequential order.

Durability Once a transaction is completed, its effects are permanent and will survive any subsequent system failures.

Essential for maintaining the reliability of the database

ACID Compliance

The ACID principles guarantee that a database will maintain a consistent picture of the world.



Data is partitioned or replicated

Strong Consistency

- Data is consistent across the entire network
- Key feature of relational databases that ensures ACID

DeepLearning.AI

Introduction to Source Systems

Lab Walkthrough - Interacting with Amazon DynamoDB NoSQL Database

Interacting with Amazon DynamoDB

Amazon DynamoDB

Apply some **C**reate, **R**ead, **U**pdate and **D**elete (**CRUD**) operations

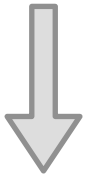
In this video,

- Overview of DynamoDB features
- Data you will work on
- DynamoDB methods that you will use to apply **CRUD** operations

Amazon DynamoDB

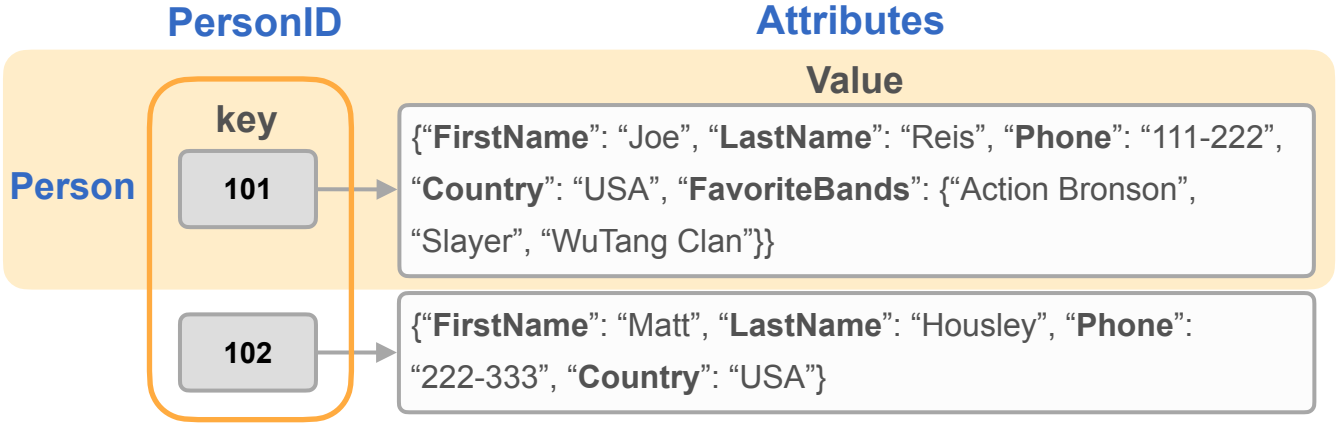
Key-value Database

Key-value Items



Table

- Row: attributes of one item
- Uniquely identified by the item's key.
- **Simple Primary Key:** partition key
- **Composite Primary Key:**



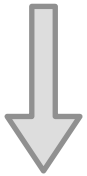
Simple primary key

Key: PersonID	Attributes				
101	FirstName	LastName	Phone	Country	FavoriteBands
	Joe	Reis	111-222	USA	{"Action Bronson", "Slayer", "WuTang Clan"}
102	FirstName	LastName	Phone	Country	
	Matt	Housley	222-333	USA	

Amazon DynamoDB

Key-value Database

Key-value Items



Table

- Row: attributes of one item
- Uniquely identified by the item's key.
- **Simple Primary Key:** partition key
- **Composite Primary Key:** partition key & sort key

Composite Primary Key		Attributes				
Partition Key	**Sort Key**					
OrderID	ItemNum					
1234	Item1	Price	Quantity	ProductType	ISBN	Title
		10	1	Book	45679	Data
1234	Item2	Price	Quantity	ProductType	Brand	Color
		50	1	Bike	AZY	Black
1235	Item1	Price	Quantity	ProductCode		
		23	4	23697		
1235	Item2	Price	Quantity	ProductType	Brand	
		1200	2	Laptop	XYZ	

Schema-less: Each item can have its own distinct attributes.

Simple primary key

Key: PersonID	Attributes				
101	FirstName	LastName	Phone	Country	FavoriteBands
	Joe	Reis	111-222	USA	{"Action Bronson", "Slayer", "WuTang Clan"}
102	FirstName	LastName	Phone	Country	
	Matt	Housley	222-333	USA	

Interacting with Amazon DynamoDB


Table


Table


Table


Table

Interact with the tables using Python

Boto3

AWS Software Development Kit (SDK) for Python
Allows you to create and configure AWS services using Python

Boto3 1.34.144 documentation

Search

Feedback

Do you have a suggestion to improve this website or boto3? Give us feedback.

Boto3 documentation

You use the AWS SDK for Python (Boto3) to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented API as well as low-level access to AWS services.

Note

Documentation and developers tend to refer to the AWS SDK for Python as "Boto3," and this documentation often does so as well.

Quickstart

Interacting with Amazon DynamoDB


Table


Table


Table


Table

Interact with the tables using Python

Boto3

AWS Software Development Kit (SDK) for Python
Allows you to create and configure AWS services using Python

```
import boto3

client = boto3.client('dynamodb')
```

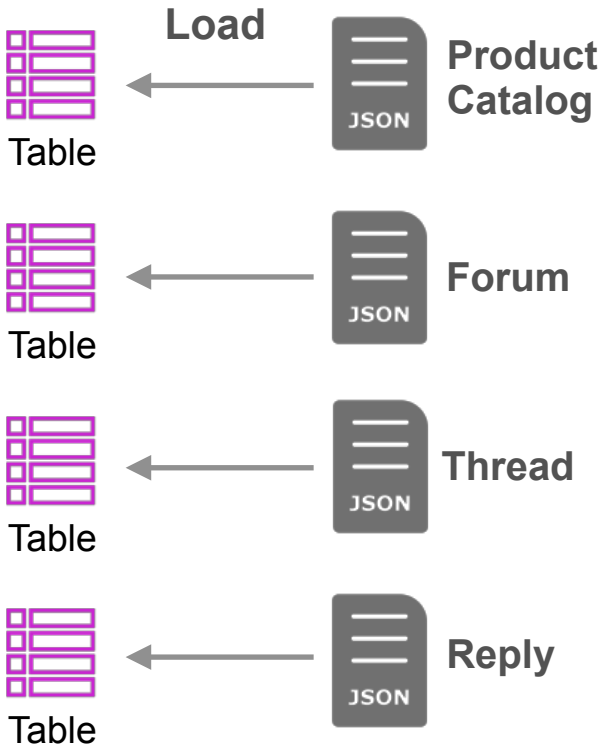
Create	`create_table`

Read	`scan`
	`get_item`
	`query`

Update	`put_item`
	`write_batch_items`
	`update_item`

Delete	`delete_item`

Data



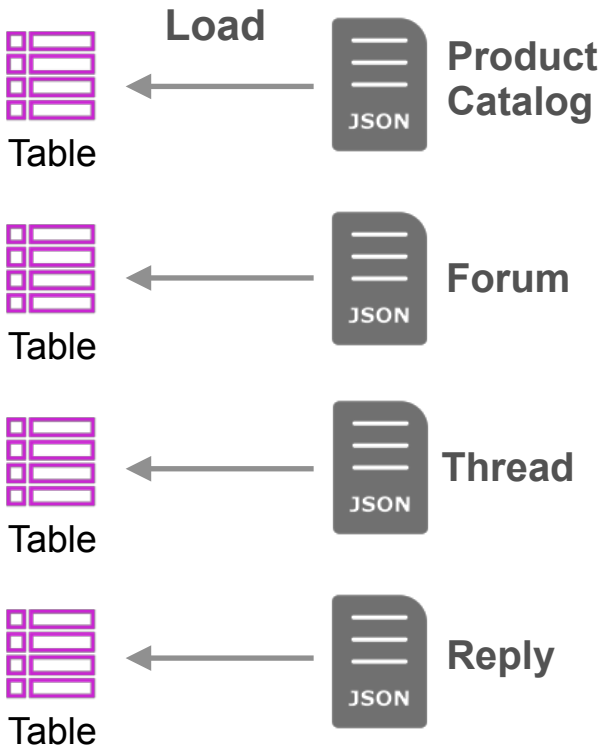
- Information about some products sold on Amazon
- *ID*: simple primary key

- Information about some AWS forums
- Each forum: total number of threads, messages and views
- *Name*: simple primary key

- Information about each forum thread
- Each thread: subject, message, total number of views and replies
- *ForumName* & *Subject*: composite primary key

- Information about each thread reply
- Each reply: time, reply message, user, ID (Forum and thread subject)
- *ID* & *ReplyDateTime:* composite primary key

Data


Load
Table
JSON
Product Catalog
Table
JSON
Forum
Table
JSON
Thread
Table
JSON
Reply
Table

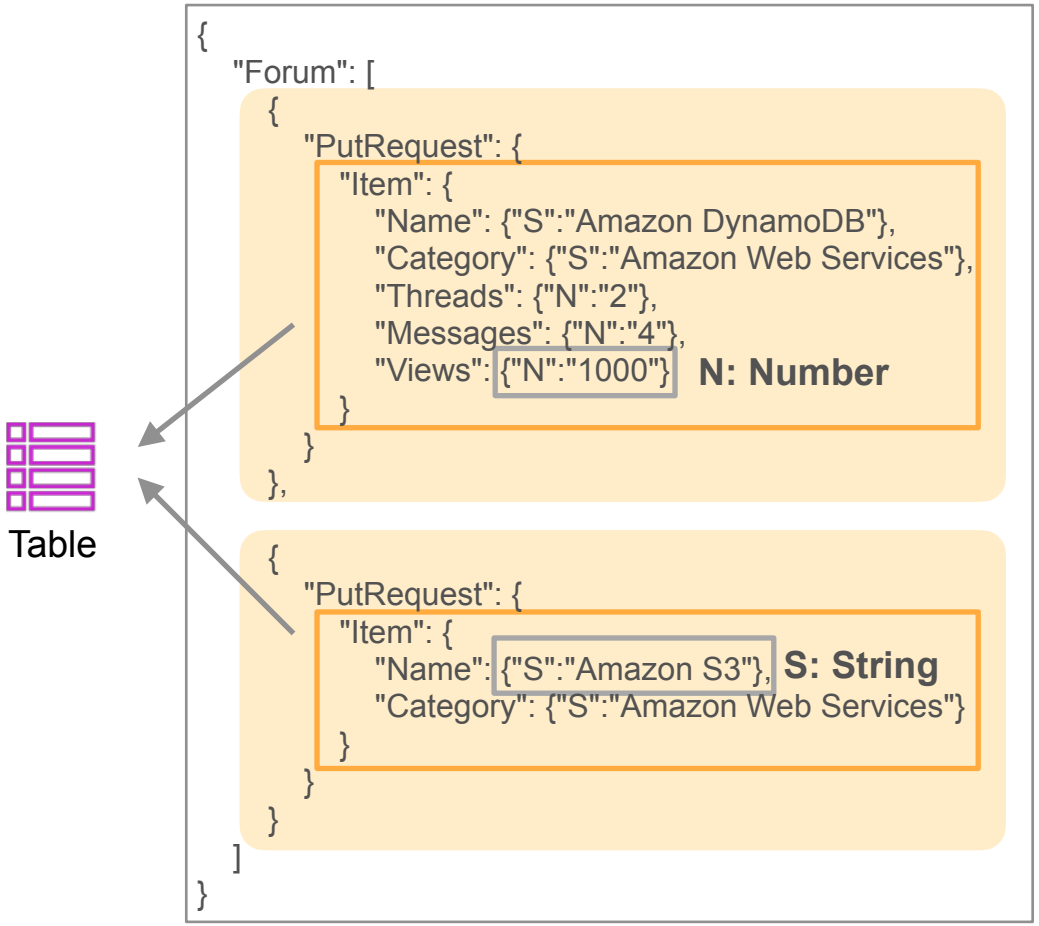
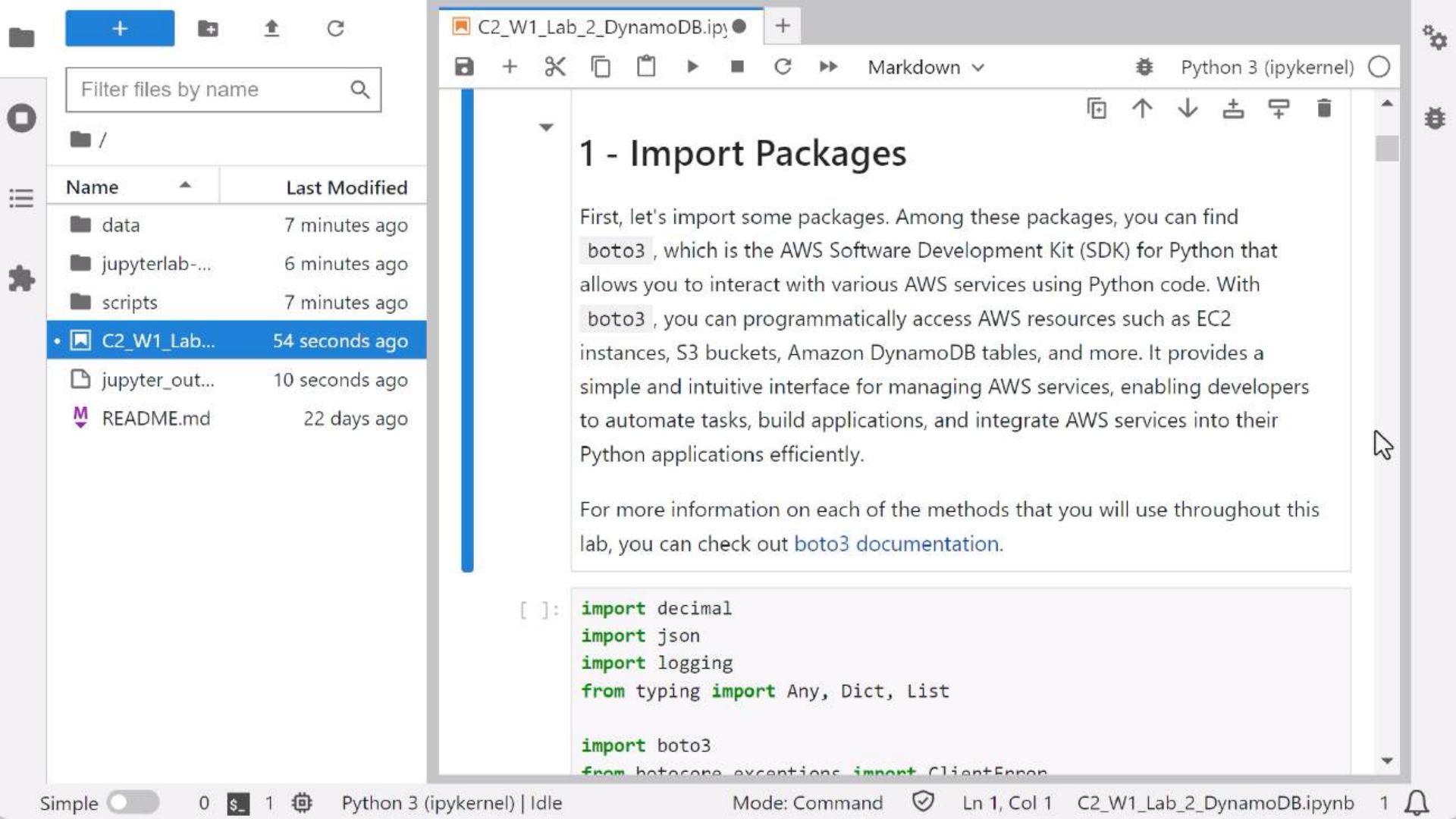


Table
{
"Forum": [
{
"PutRequest": {
"Item": {
"Name": {"S":"Amazon DynamoDB"},
"Category": {"S":"Amazon Web Services"},
"Threads": {"N":"2"},
"Messages": {"N":"4"},
"Views": {"N":"1000"}
}
}
},
N: Number
{
"PutRequest": {
"Item": {
"Name": {"S":"Amazon S3"},
"Category": {"S":"Amazon Web Services"}
}
}
},
S: String
]
}


Filter files by name
/
Name
Last Modified
data
7 minutes ago
jupyterlab-...
6 minutes ago
scripts
7 minutes ago
C2_W1_Lab...
54 seconds ago
jupyter_out...
10 seconds ago
README.md
22 days ago
C2_W1_Lab_2_DynamoDB.ipy
Markdown
Python 3 (ipykernel)
1 - Import Packages
First, let's import some packages. Among these packages, you can find boto3, which is the AWS Software Development Kit (SDK) for Python that allows you to interact with various AWS services using Python code. With boto3, you can programmatically access AWS resources such as EC2 instances, S3 buckets, Amazon DynamoDB tables, and more. It provides a simple and intuitive interface for managing AWS services, enabling developers to automate tasks, build applications, and integrate AWS services into their Python applications efficiently.
For more information on each of the methods that you will use throughout this lab, you can check out boto3 documentation.
[ ]:
import decimal
import json
import logging
from typing import Any, Dict, List
import boto3
Simple
0
1
Python 3 (ipykernel) | Idle
Mode: Command
Ln 1, Col 1
C2_W1_Lab_2_DynamoDB.ipynb
1

DeepLearning.AI

Introduction to Source Systems

Object Storage

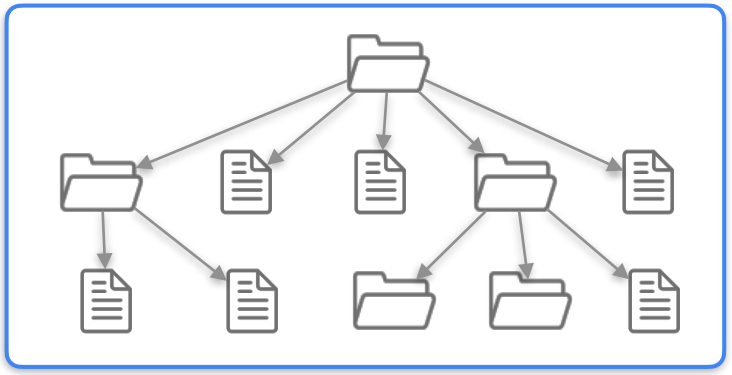
Object Storage

files

Object Storage

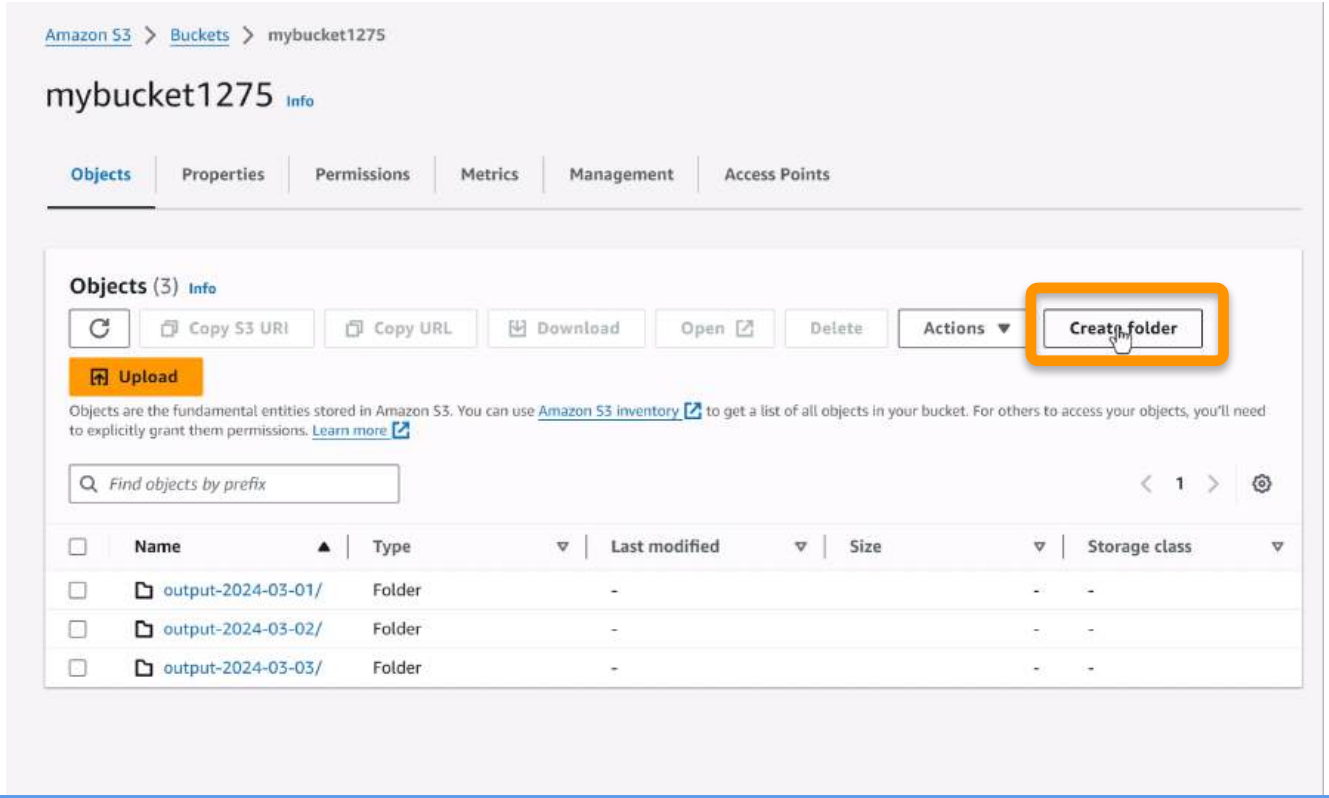
No hierarchy!

Traditional File System Hierarchy



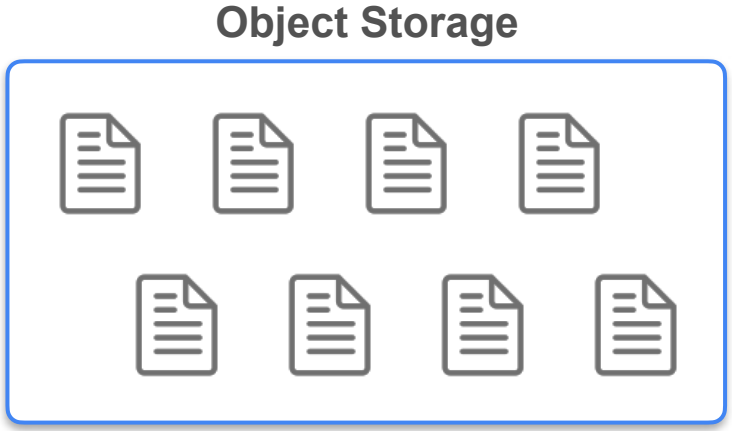
Object Storage

Amazon S3



Object Storage

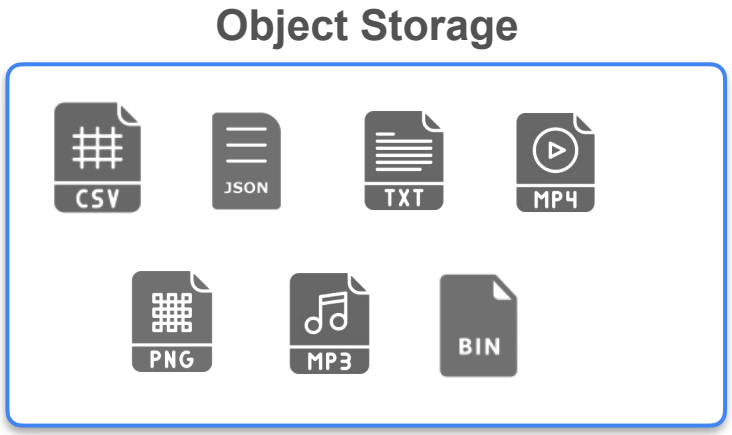
files



No hierarchy!

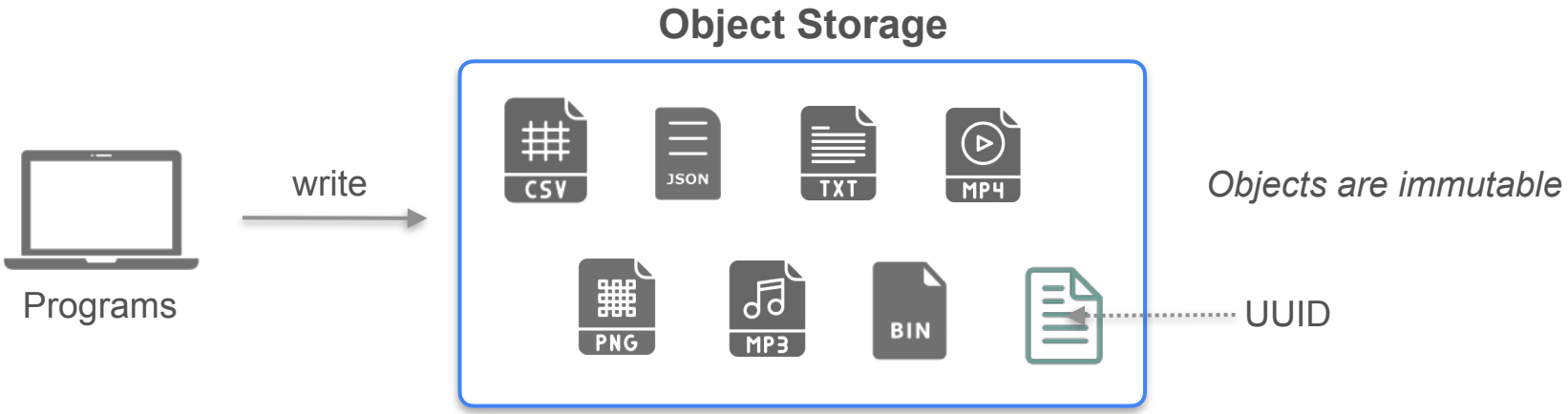
Object Storage

files



- Storing semi-structured and unstructured data
- Serving data for training machine learning models

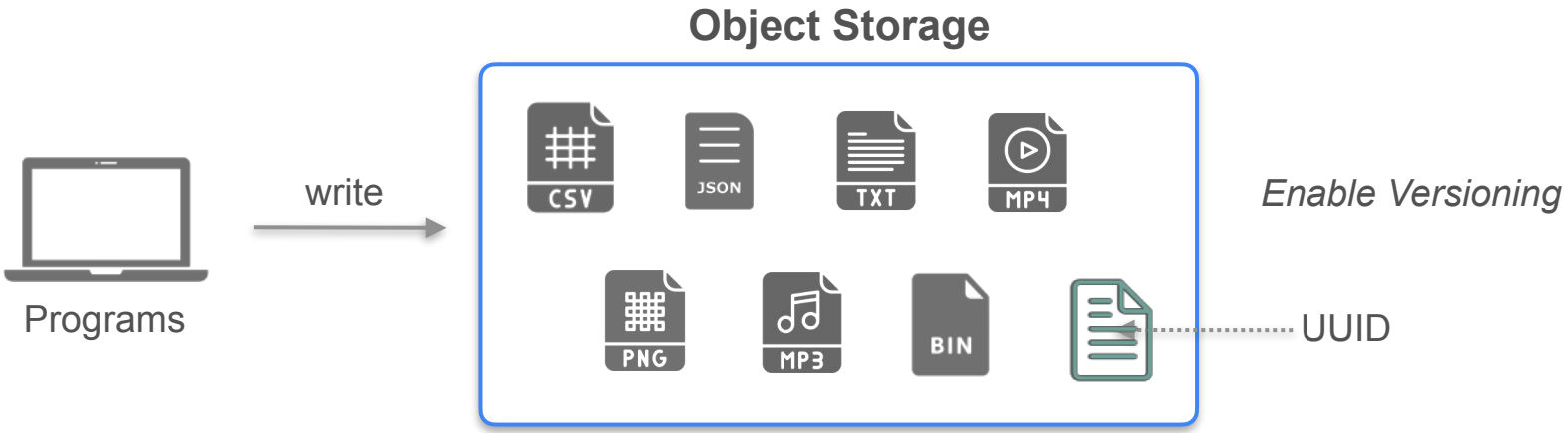
Object Storage



For each object,

- Universal Unique Identifier or UUID (key)
- Metadata: creation date, file type, owner

Object Storage



For each object,

- Universal Unique Identifier or UUID (key)
- Metadata: creation date, file type, owner, version

Why Use Object Storage?

- Store files of various data formats without a specific file system structure
- Easily scale out to provide virtually limitless storage space
- Replicate data across several availability zones

Amazon S3

99.999999999% : data durability

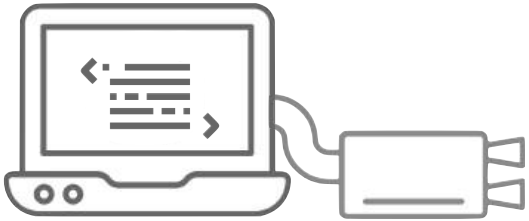
- Cheaper than other storage options

DeepLearning.AI

Introduction to Source Systems

Logs

Logs



Software Application

Logs

```
01-01-2025:10.30   67945   success   user added a product x to their cart
01-01-2025:10.32   38910   fail      invalid values typed for product quantity
01-01-2025:10.38   17462   fail      customer table corrupted
```

Exhaust / Byproduct

Monitoring or Debugging a system

- User activity:
  - Signing in
  - navigating to a particular page
- An update to a database
- An error from a procedure

Log

An append-only sequence of records ordered by time, capturing information about events that occur in systems.

Rich data source

Web Server Logs
Detailed user activity data

Database System Logs

Security System Logs

Downstream use cases

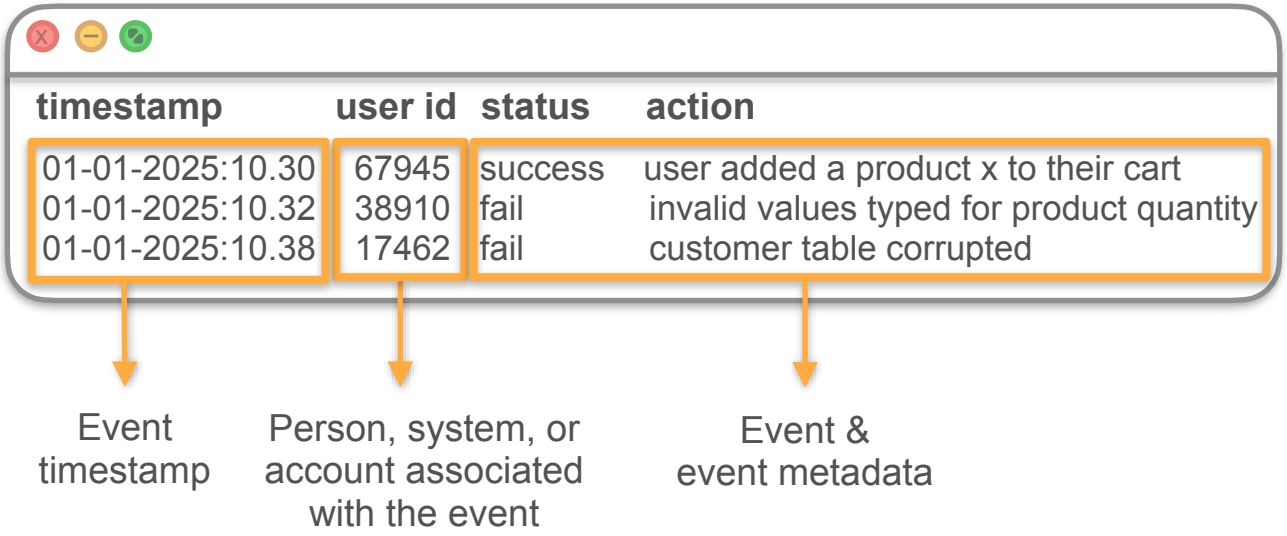
Analysis of user behavior patterns

Track changes in source database

Machine Learning *anomaly detection*

Log

An append-only sequence of records ordered by time, capturing information about events that occur in systems.



Log

An append-only sequence of records ordered by time, capturing information about events that occur in systems.

timestamp	user id	status	action
01-01-2025:10.30	67945	success	user added a product x to their cart
01-01-2025:10.32	38910	fail	invalid values typed for product quantity
01-01-2025:10.38	17462	fail	customer table corrupted

```
{
  "user id": 67945,
  "action": "user added a product x to their cart",
  "status": "success",
  "time-stamp": 01-01-2025:10.30
}
```

[00101011 11000101 11001001 11000101 110001001]

user id	action	status	timestamp
67945	user added a product x to their cart	success	01-01-2025:10.30
38910	invalid values typed for product quantity	fail	01-01-2025:10.32
17462	customer table corrupted	fail	01-01-2025:10.38

Log Levels

A tag to categorize the event (log level)

- “debug”
- “info”
- “warn”
- “error”
- “fatal”

user id	action	status	timestamp	level
67945	user added a product x to their cart	success	01-01-2025:10.30	Info
38910	invalid values typed for product quantity	fail	01-01-2025:10.32	error
17462	customer table corrupted	fail	01-01-2025:10.38	fatal

DeepLearning.AI

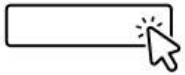
Introduction to Source Systems

Streaming Systems

Terminology

Event

Something that happened in the world or a change to the state of a system.



User clicking on a link

Sensor measuring a temperature change

Message

P**Event**er

Stream

Data: record of events

Terminology

Event

Something that happened in the world or a change to the state of a system.

User clicking on a link

Sensor measuring a temperature change

Message

A record of information about an event.

Message
Event Details
Event Metadata
Event Timestamp

Producer

Event Event Event Event

Stream

Terminology

Event

Something that happened in the world or a change to the state of a system.

User clicking on a link

Sensor measuring a temperature change

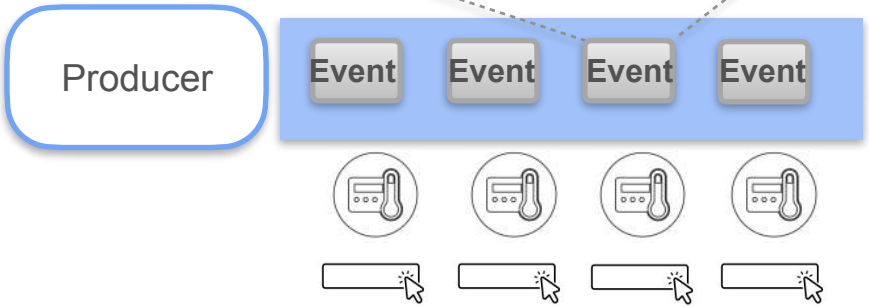
Message

A record of information about an event.

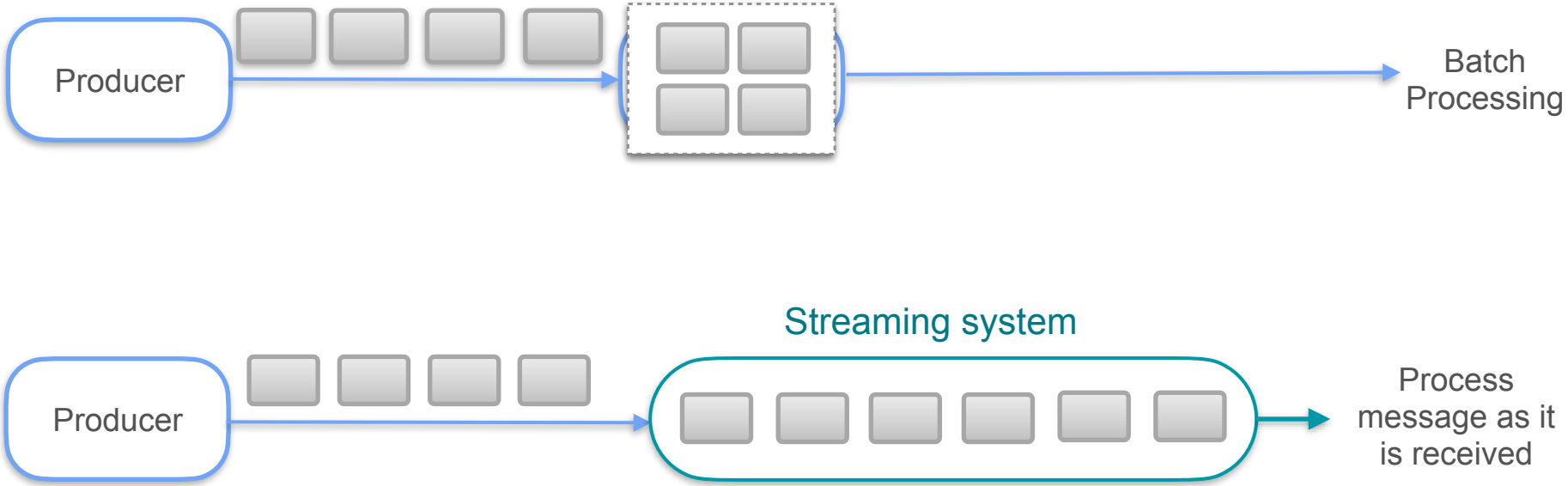
Message
Event Details
Event Metadata
Event Timestamp

Stream

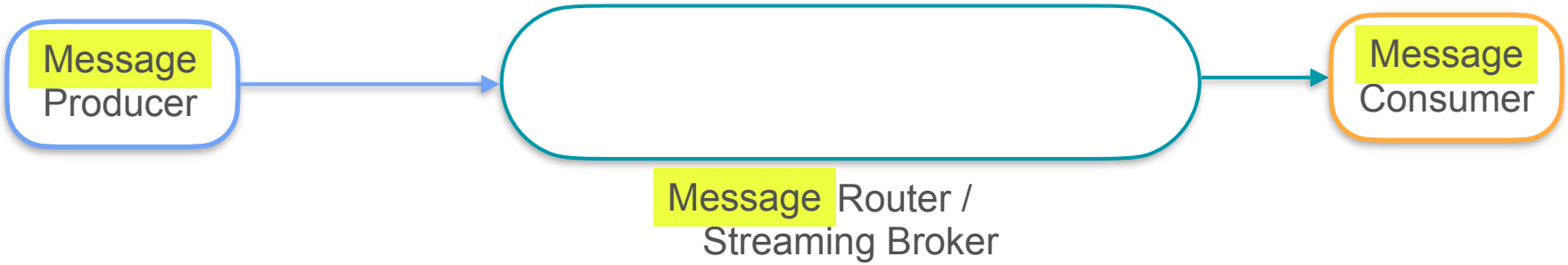
A sequence of messages.



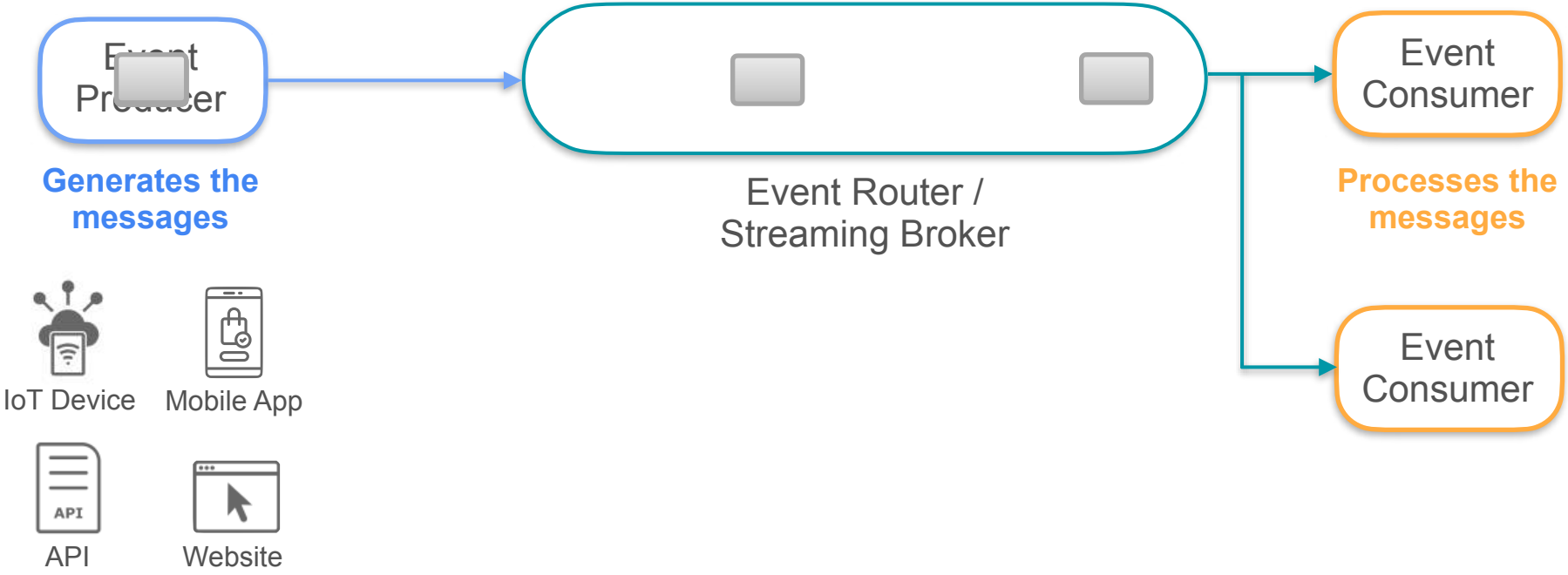
Stream Processing



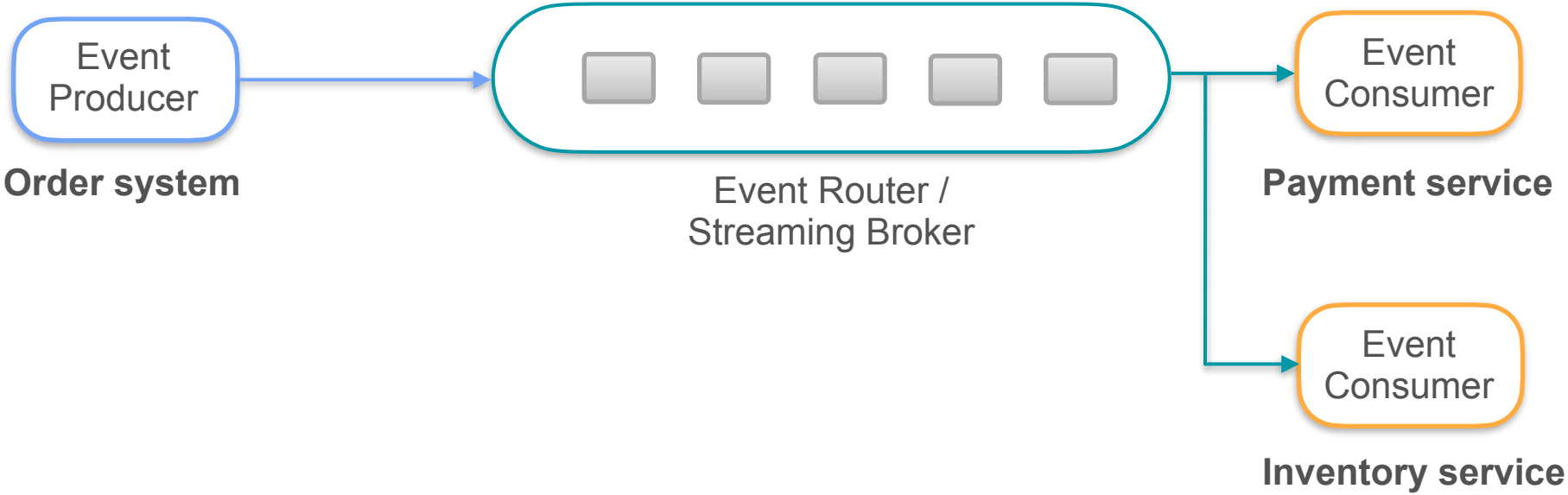
Streaming System



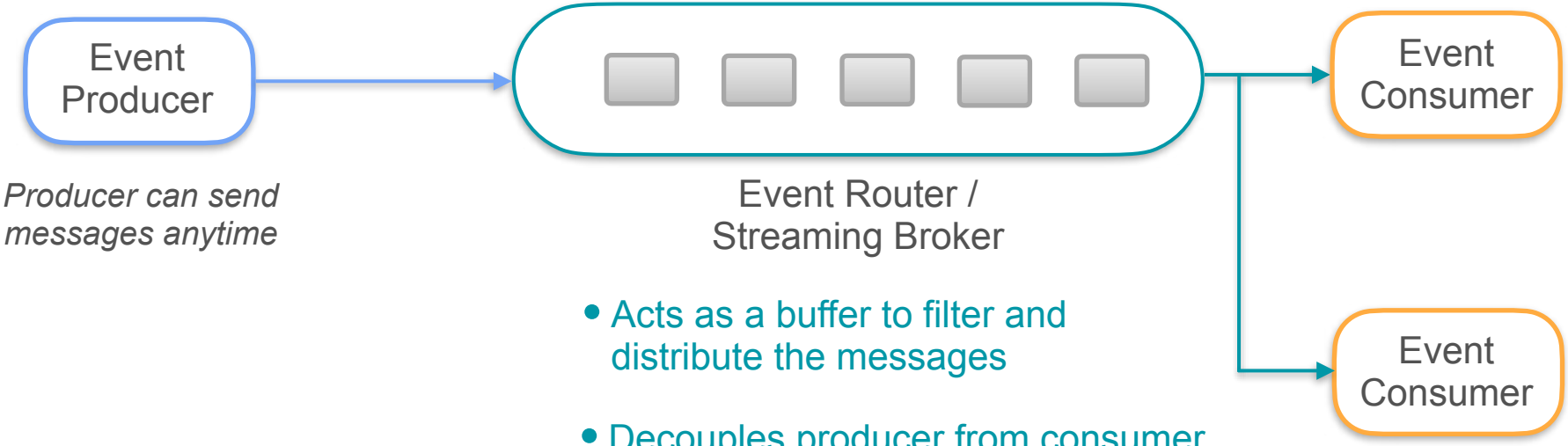
Streaming System



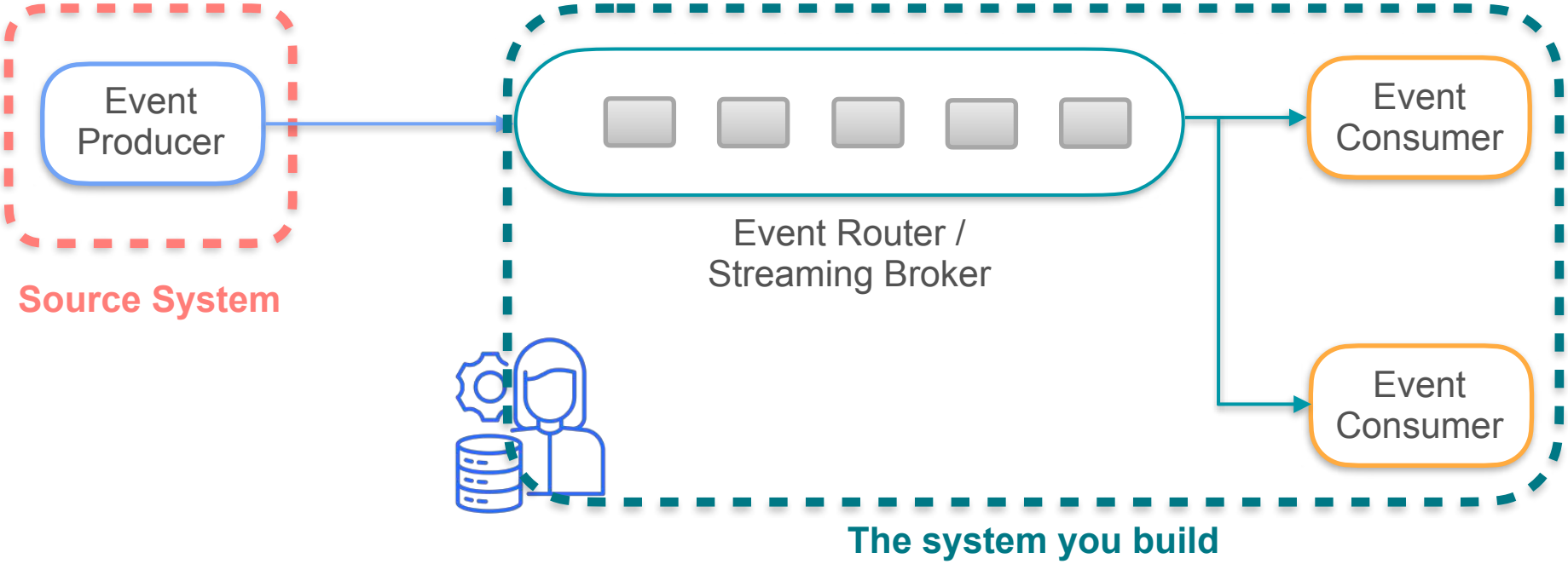
Streaming System



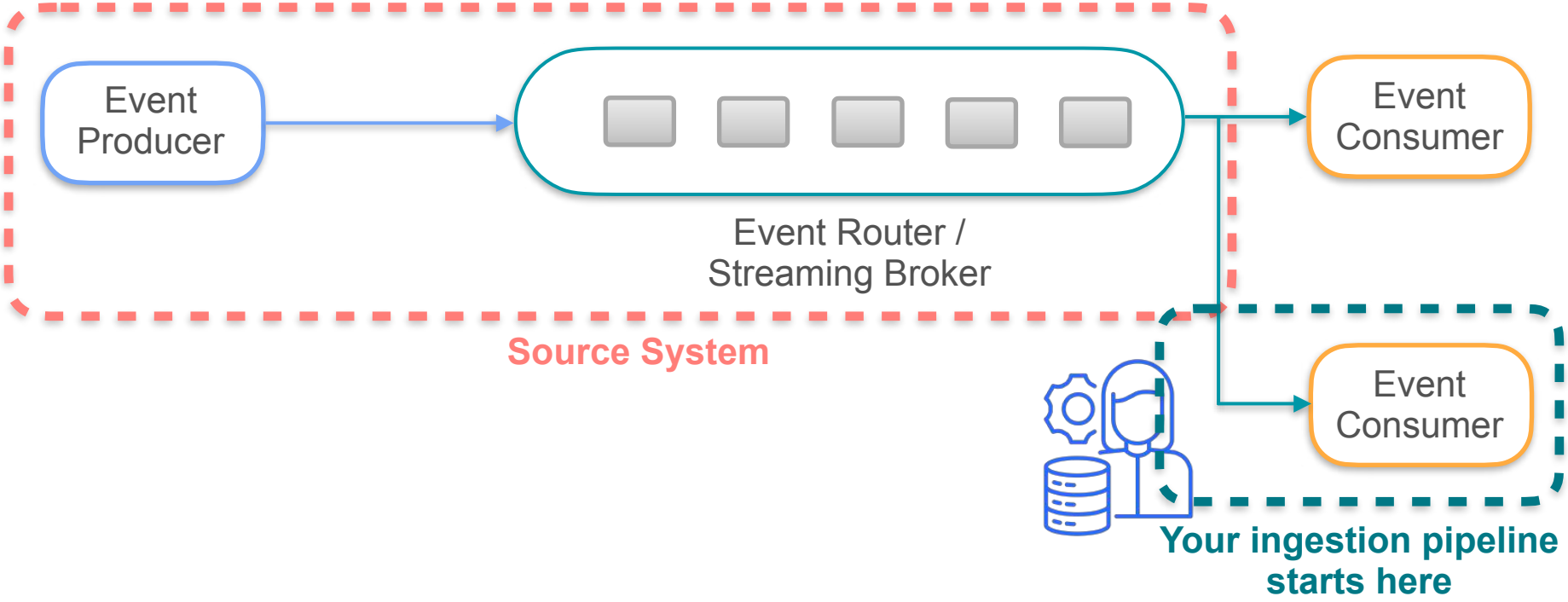
Streaming System



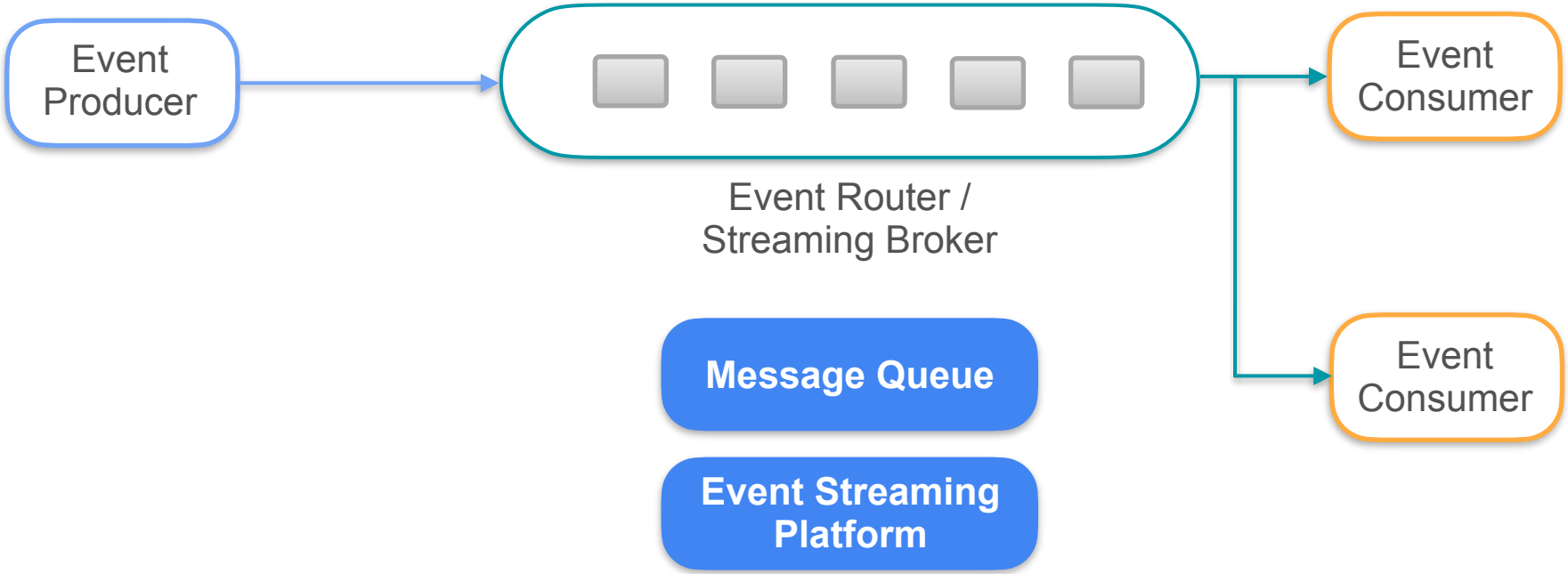
Your Data System



Your Data System

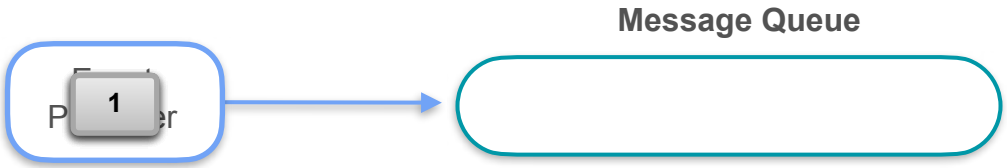


Streaming System



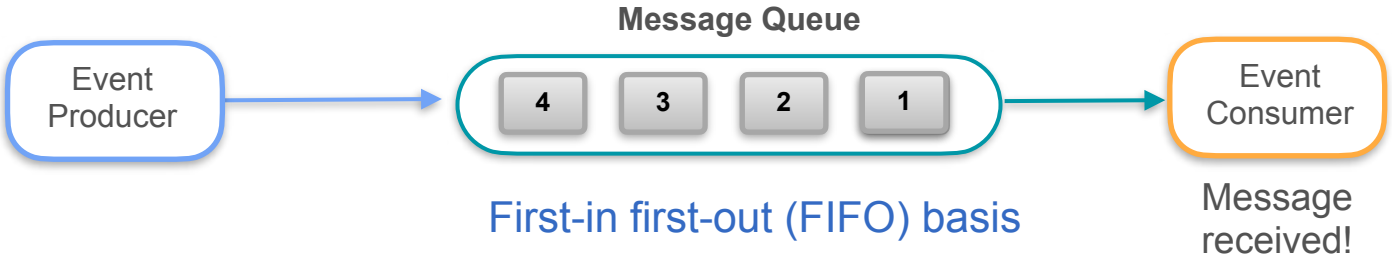
Message Queue

A queue/buffer that accumulates messages



Message Queue

A queue/buffer that accumulates messages
and delivers those messages to consumers asynchronously.



Message Queue

A queue/buffer that accumulates messages
and delivers those messages to consumers asynchronously.

First-in first-out (FIFO) basis

Message Queue

A queue/buffer that accumulates messages
and delivers those messages to consumers asynchronously.

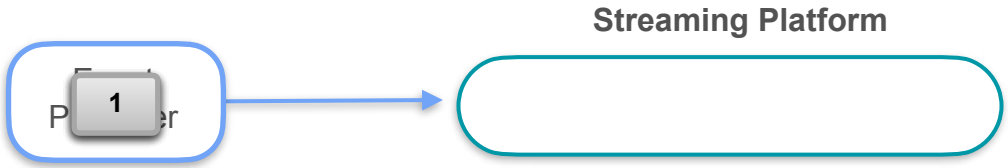
First-in first-out (FIFO) basis

Temporary storage

Amazon Simple Queue Service (Amazon SQS)

Event Streaming Platform

Log: Append-only record of events

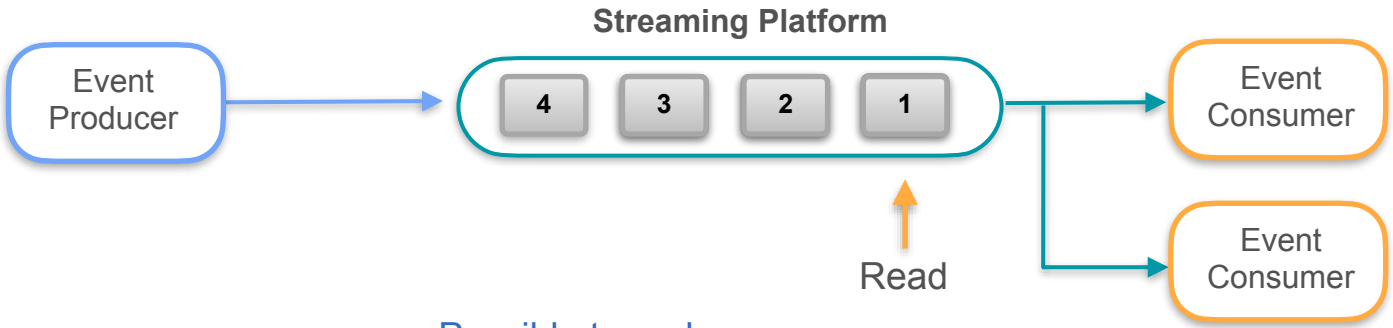


kafka

Amazon Kinesis Data Streams

Event Streaming Platform

Log: Append-only record of events



Possible to replay or reprocess any events in the log

kafka

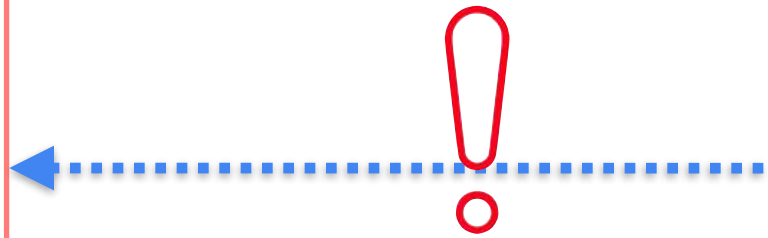
Amazon Kinesis Data Streams

DeepLearning.AI

Interacting with Source Systems

Lesson Overview

Connecting to Source Systems



- Improper identity and access managements (IAM) definitions
- Broken networking configurations
- Wrong set of access credentials

Data Engineer

Lesson's Plan

1. Ways of connecting to source systems

2. IAM roles and permissions

Key to controlling and managing access to cloud-based data sources

Role

Permissions

Lesson's Plan

1. Ways of connecting to source systems

2. IAM roles and permissions

Key to controlling and managing access to cloud-based data sources

Role

Permissions

3. Basics of networking

VPCs and Subnets, Gateways, Routing, Security groups

Lesson's Plan

1. Ways of connecting to source systems

2. IAM roles and permissions

Key to controlling and managing access to cloud-based data sources

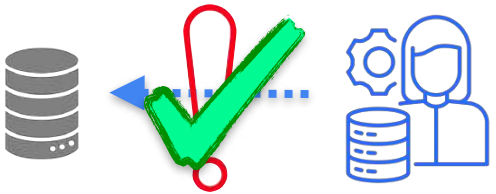
Role

Permissions

3. Basics of networking

VPCs and Subnets, Gateways, Routing, Security groups

Real world scenario



4. Lab exercise: put your skills to the test

Your job: troubleshoot and figure out the cause of the problem

DeepLearning.AI

Interacting with Source Systems

Connecting to Source Systems

Connecting to Source Systems

```
def create_client():
        dynamodb_client = boto3.client("dynamodb")

        return dynamodb_client
```

boto3: AWS Software Development Kit (SDK) for Python

Connecting to Source Systems

Running this command in Cloud9 IDE

```
mysql --host=<MySQLEndpoint> --user=<DatabaseUserName> --password=<Password> --port=3306
```

Programmatic Way

SDK (boto3)

IDE (Cloud 9)

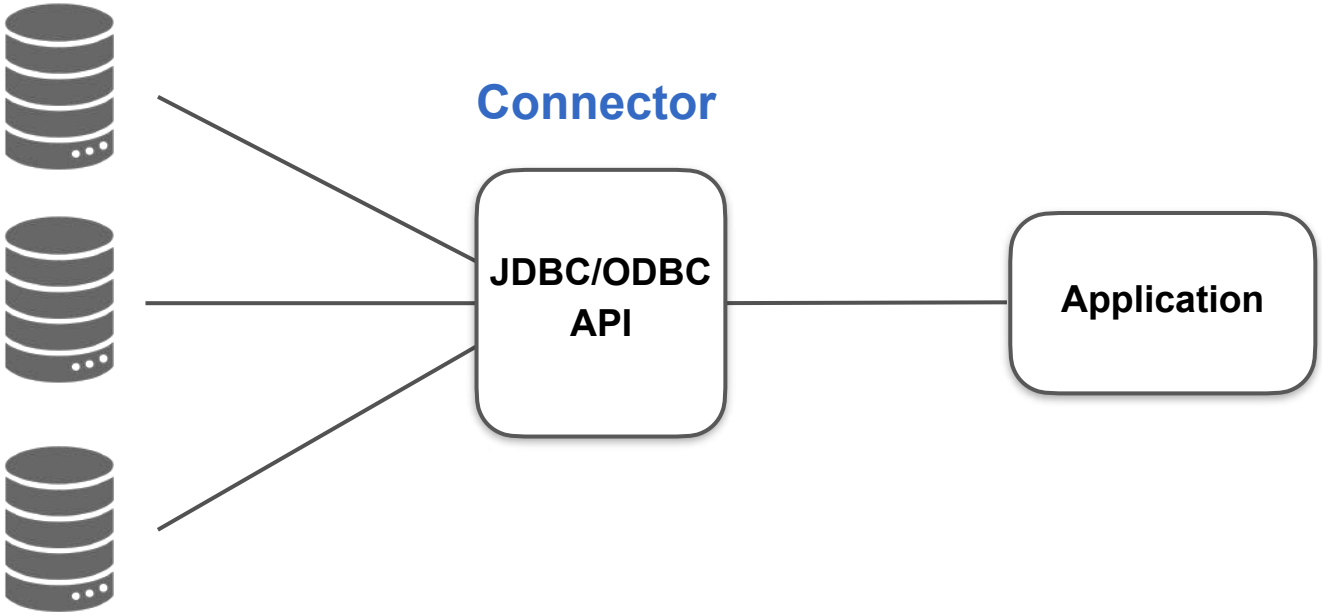
connect_to_source.py

```
1  import pymysql
2  import boto3
3
4  ENDPOINT="............"
5  PORT="3306"
6  USER="jane_doe"
7  REGION="us-east-1"
8  DBNAME="mydb"
9
10 #gets the credentials from .aws/credentials
11 session = boto3.Session(profile_name='default')
12 client = session.client('rds')
```

Jupyter Notebook

```
import pymysql
import boto3

ENDPOINT="............"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')
```

API Connector



DeepLearning.AI

Interacting with Source Systems

Basics of IAM and Permissions

Security on the Cloud

Encryption Methods

Identity and Access Management (IAM)

Networking Protocols

We're only human:

The #1 root cause of cloud data breaches is human error

- Insecure storage of passwords
- IAM misconfigurations

Mistakes

Confidential data

Public S3 Bucket

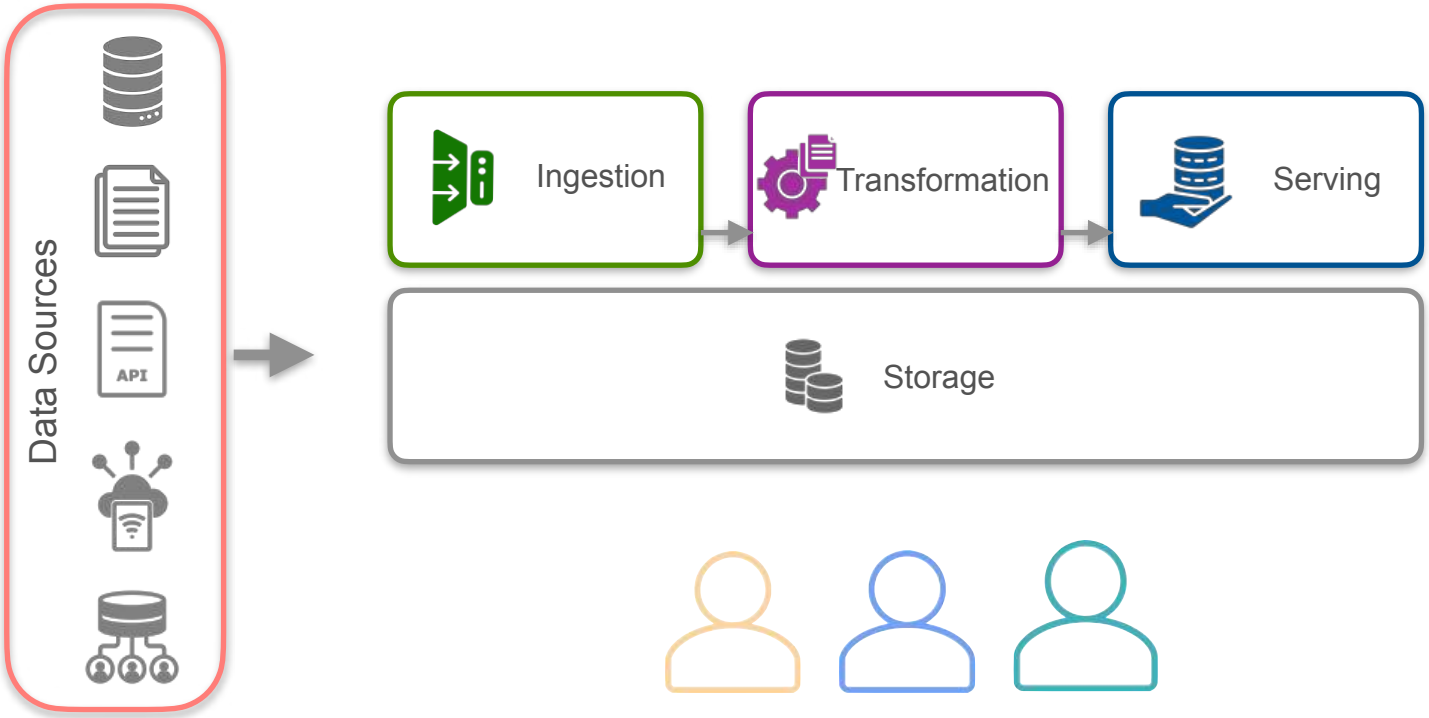
Admin access

IAM

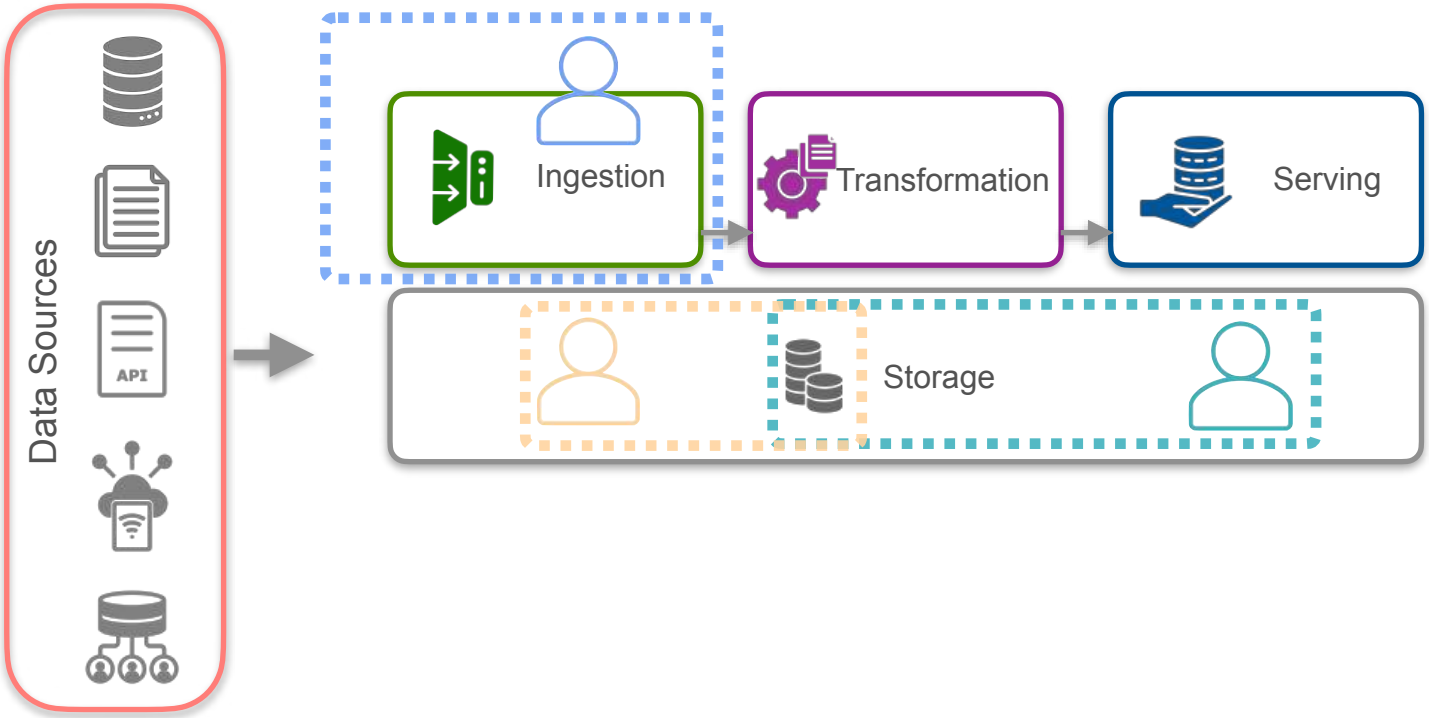
IAM is a framework for managing permissions.

Permissions define which actions an identity can perform on a specific set of resources

Principle of Least Privilege



Principle of Least Privilege



Principle of Least Privilege

AWS IAM

AWS Identity and Access Management (IAM)

AWS IAM

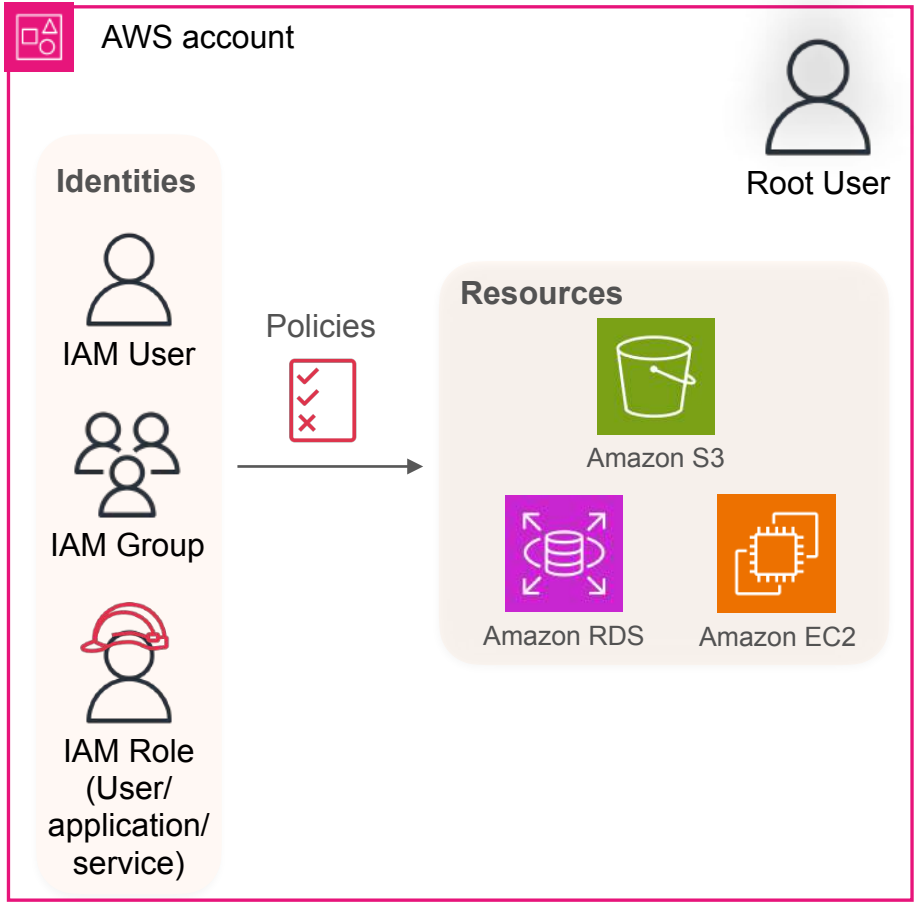
Root User — Has unrestricted access to all resources

IAM User — Has specific permissions to certain resources

- Username & password
- Access key

IAM Group — A collection of users that inherit the same permission from the group policy

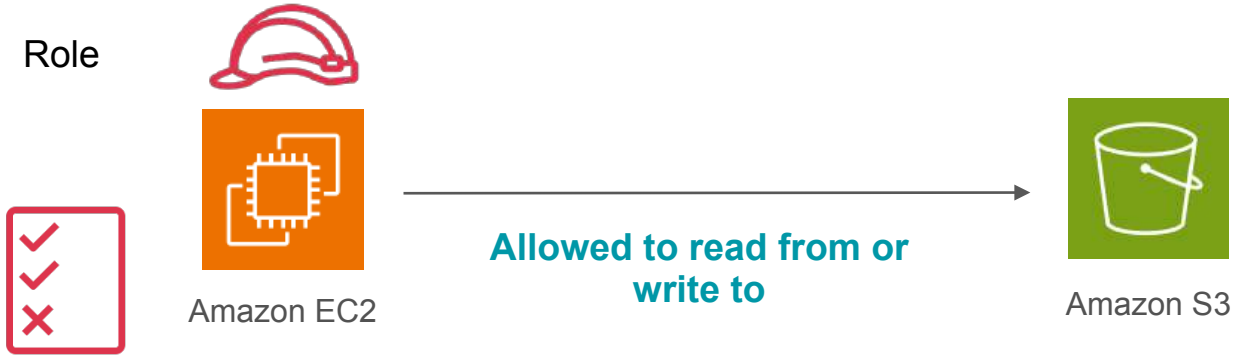
IAM Role — A user, application, or service that's been granted temporary permissions



AWS IAM


Role
Amazon EC2
Not allowed to read
from or write to
Amazon S3

AWS IAM



More secure than storing long-term user credentials within the EC2 configurations

Check if credentials have expired!

IAM Policies

permission to access the specified S3 buckets

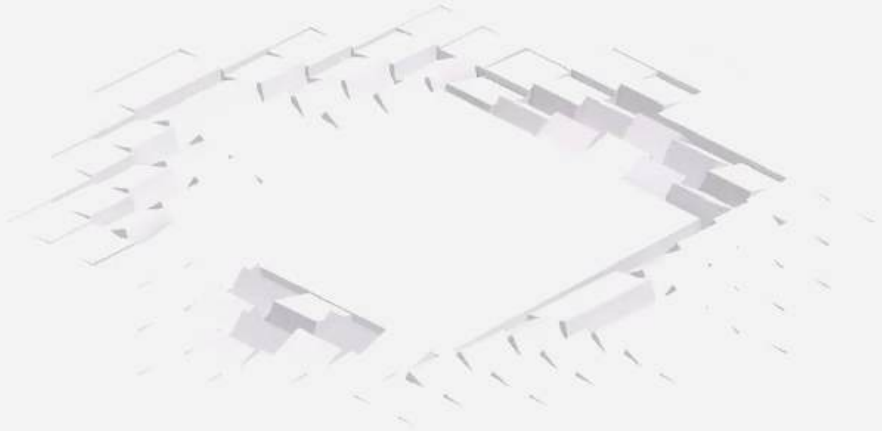
permission to access the AWS Glue job

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      “Sid”: “S3AccessDLAIBucket”,
      “Action”: [
                “s3:List*”,
                “s3:Get*”
      ],
      "Effect":"Allow",
      “Resource”: [
                “arn:aws:s3:::dlai-data-engineering”,
                “arn:aws:s3:::dlai-data-engineering/*”
      ]
    },
    {
      “Sid”: “GlueMgmt”,
      “Action”: [
                “glue:*”
      ],
      "Effect":"Allow",
      “Resource”: [
                “arn:aws:glue:*:*:catalog”,
                “arn:aws:glue:*:*:*/de-c1w2*”
      ]
    }
}
```

DeepLearning.AI

Interacting with Source Systems

Basics of Networking



AWS Cloud

What does cloud in “cloud computing” mean?

The “cloud” is made up of very real physical data centers that are spread out around the world.

Each dot represents a region

Screenshot from AWS Global Infrastructure (2023)

AWS Cloud

Resources are replicated across availability zones to ensure that your systems keep working even if a data center goes down.

AWS Cloud

Region considerations:

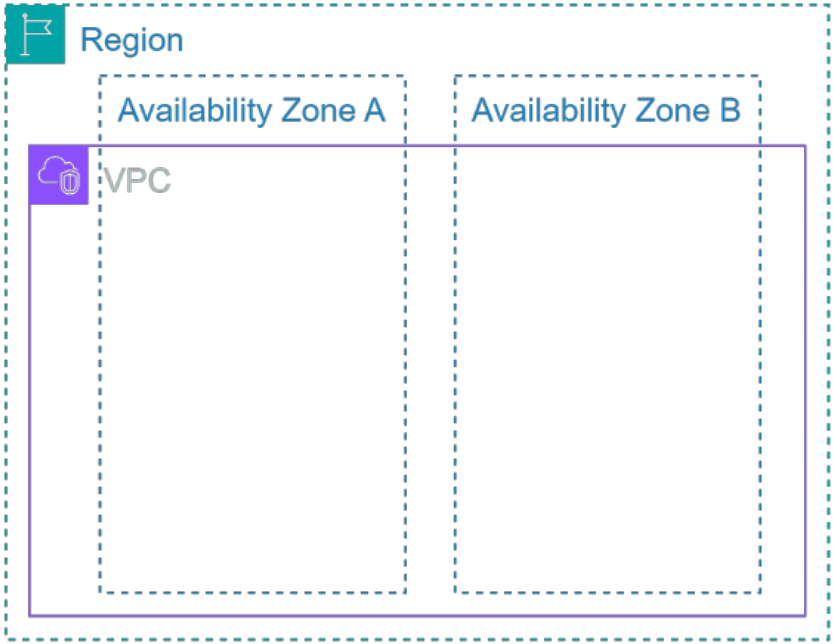
- Legal compliance
- Latency: *the closer your end users are to the region, the lower the latency*
- Availability: *the more availability zones, the better you will be able to recover from a disaster*
- Cost

AWS Cloud

Region considerations:

- Legal compliance
- Latency
- Availability
- Cost

Virtual Private Cloud

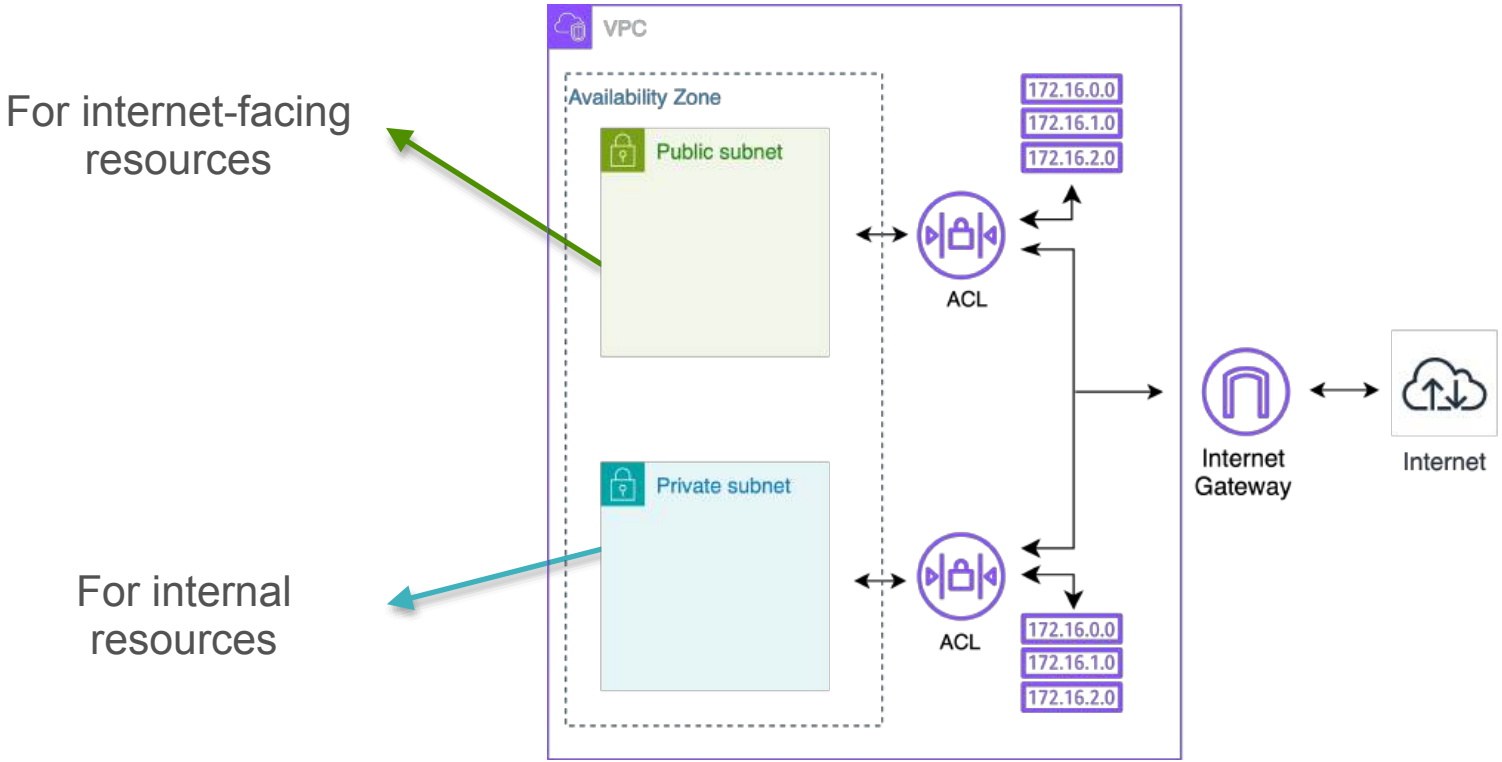


Virtual Private Cloud (VPC)

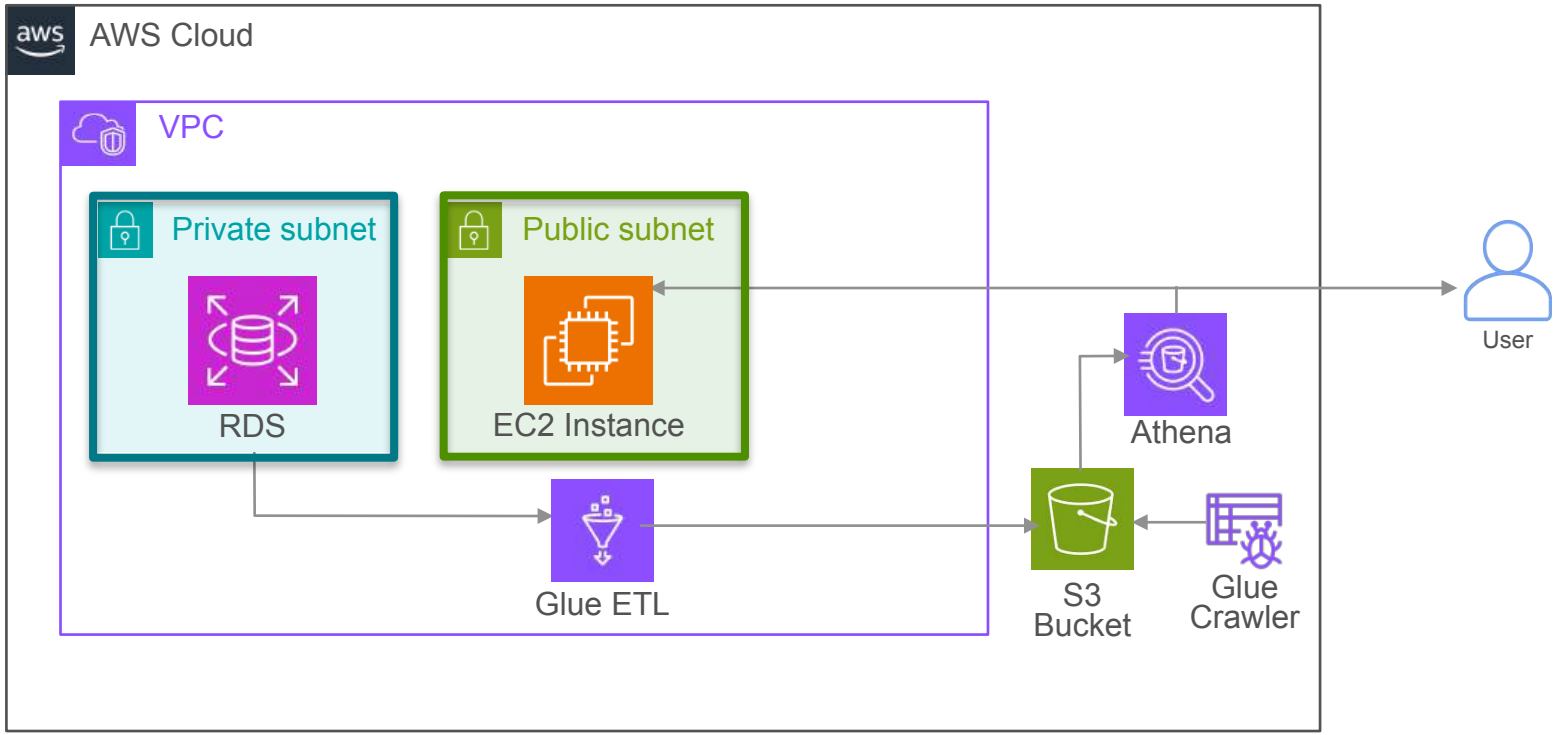
Smaller networks that span multiple availability zones

Virtual Private Cloud

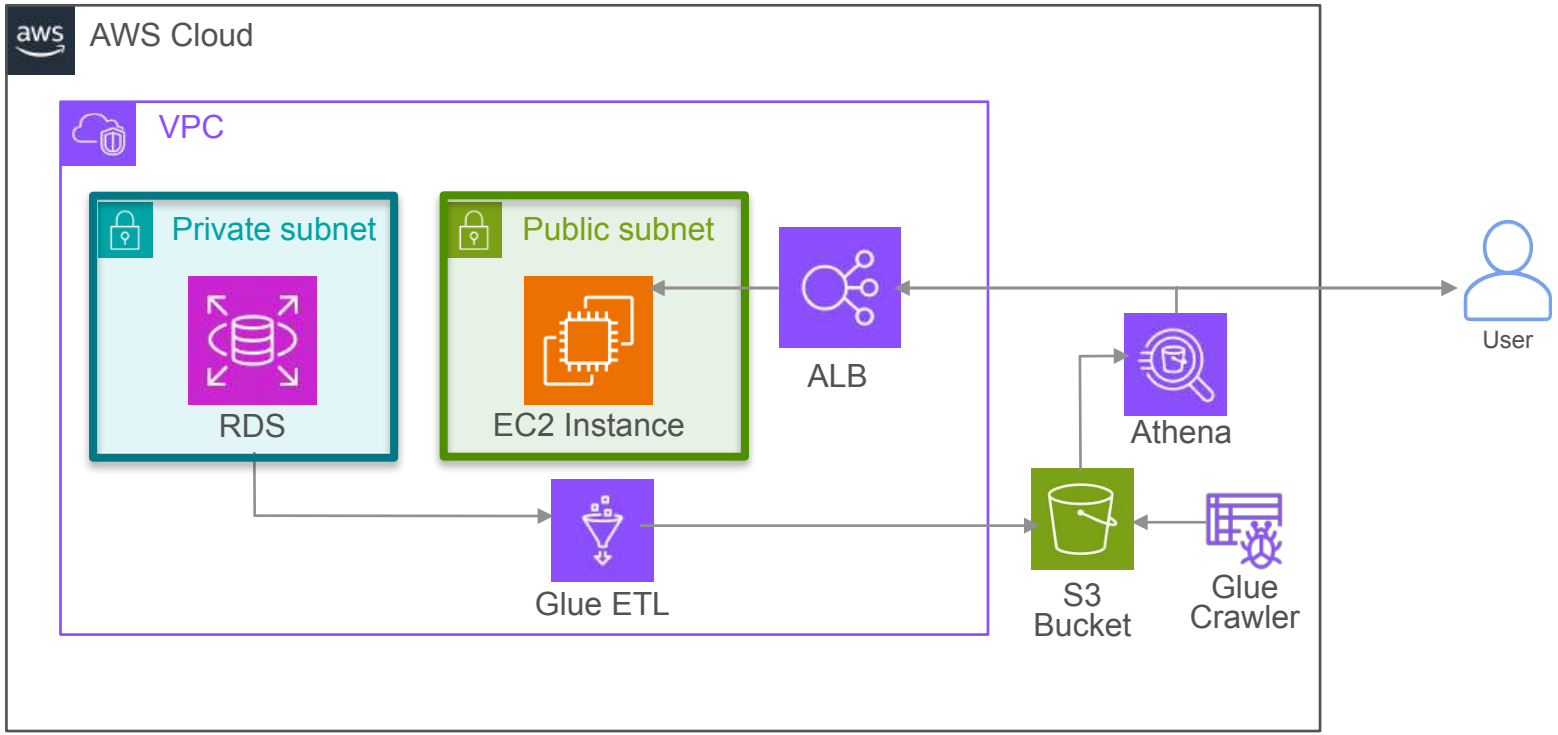
Virtual Private Cloud



Virtual Private Cloud



Virtual Private Cloud

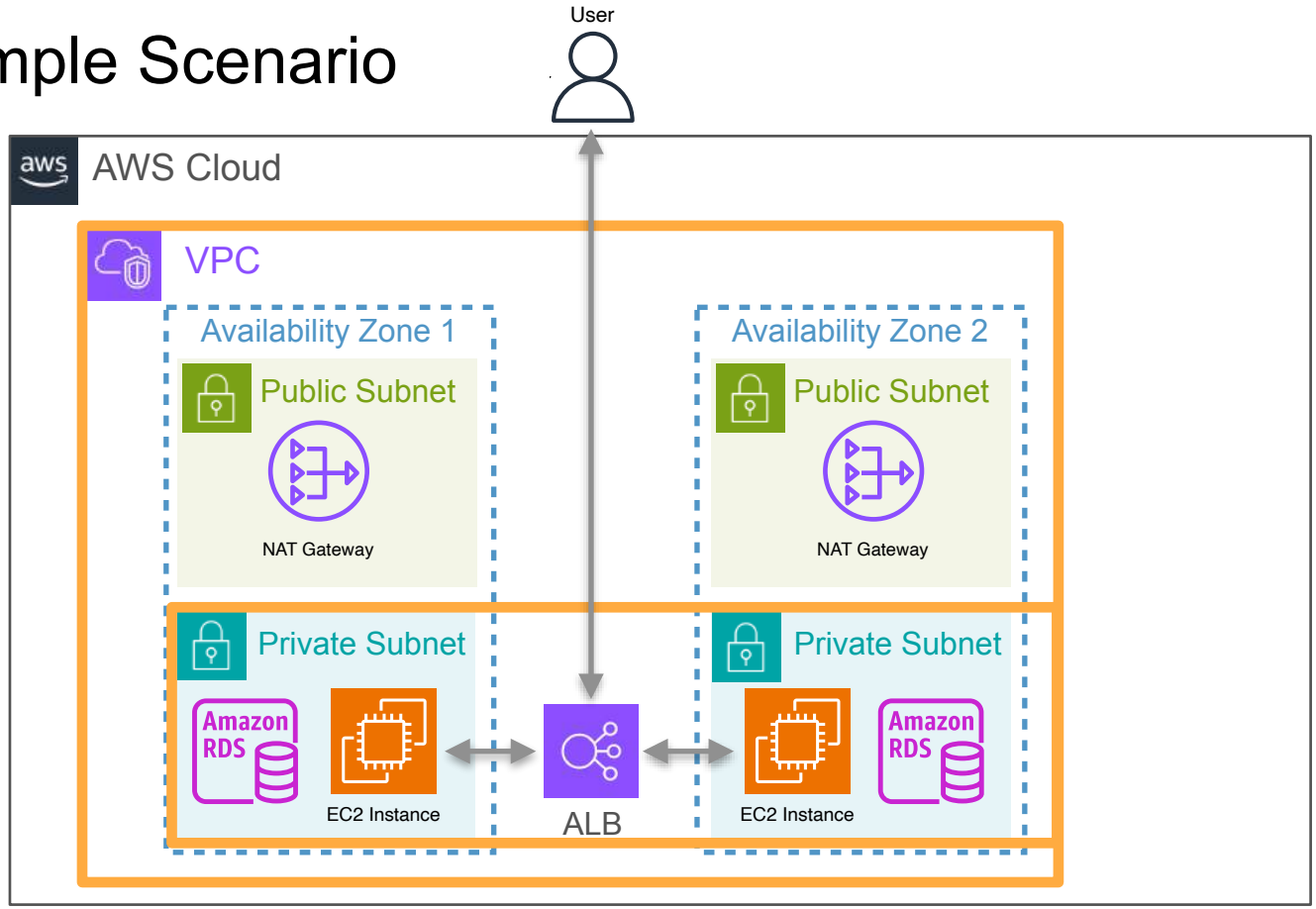


DeepLearning.AI

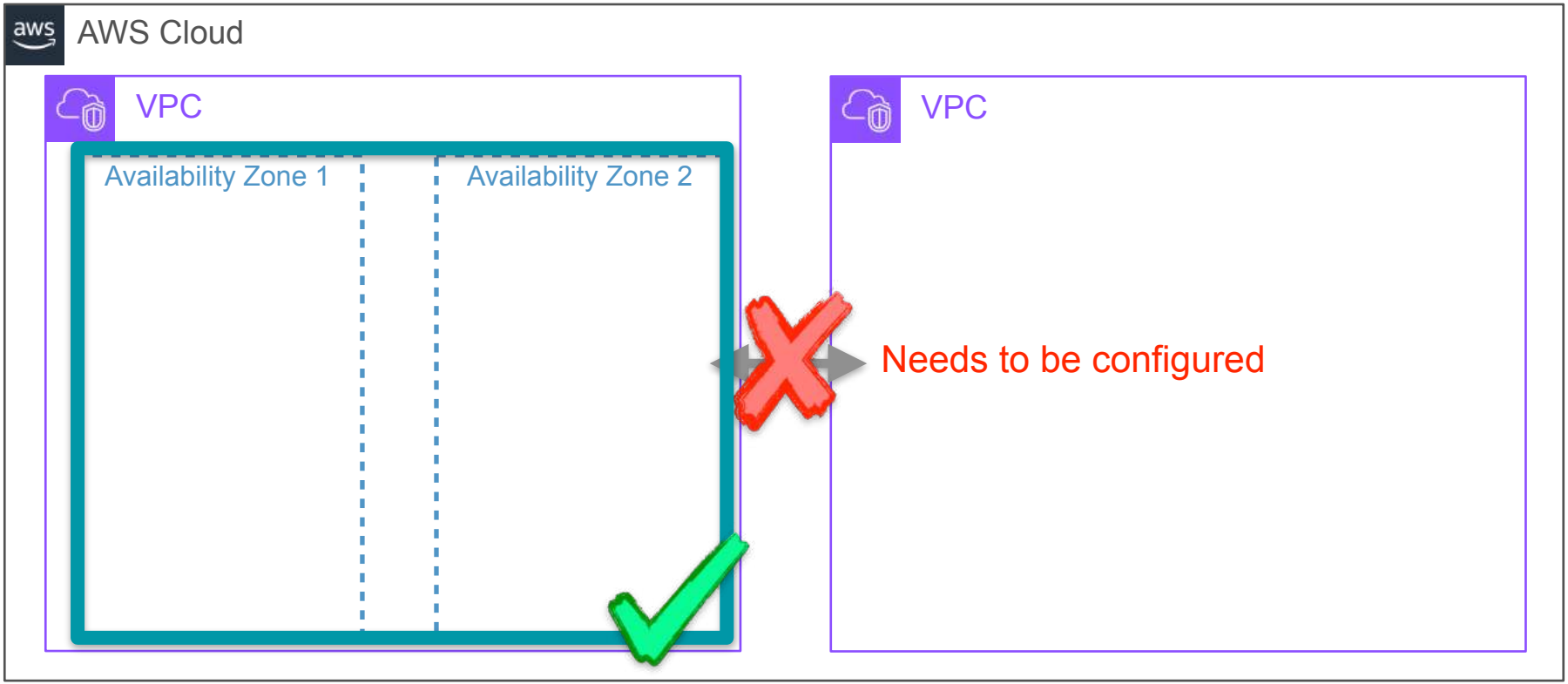
Interacting with Source Systems

AWS Networking - VPCs & Subnets

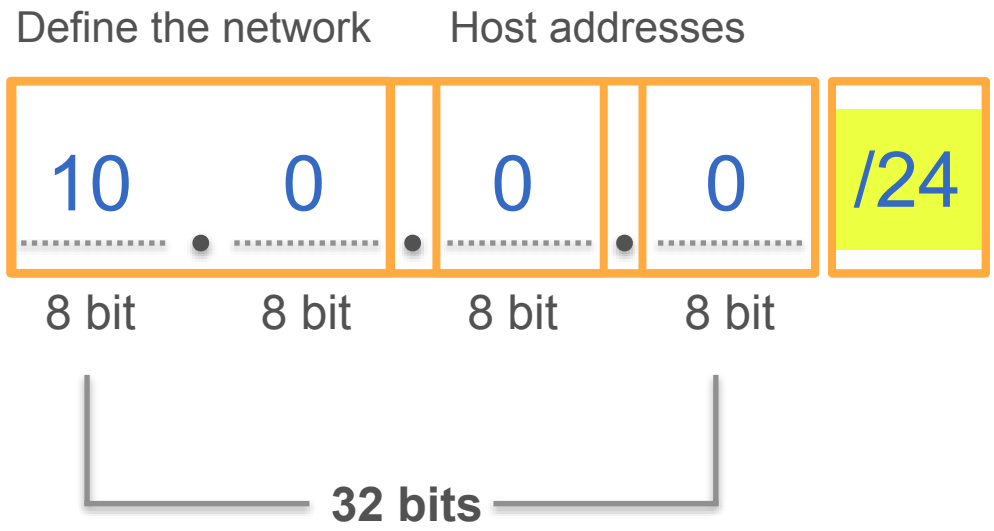
Example Scenario



AWS Networking - VPCs & Subnets



AWS Networking - VPCs & Subnets

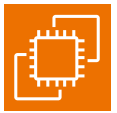


16 bits

prefix length

How many bits used for the network part of the address

10.0._._

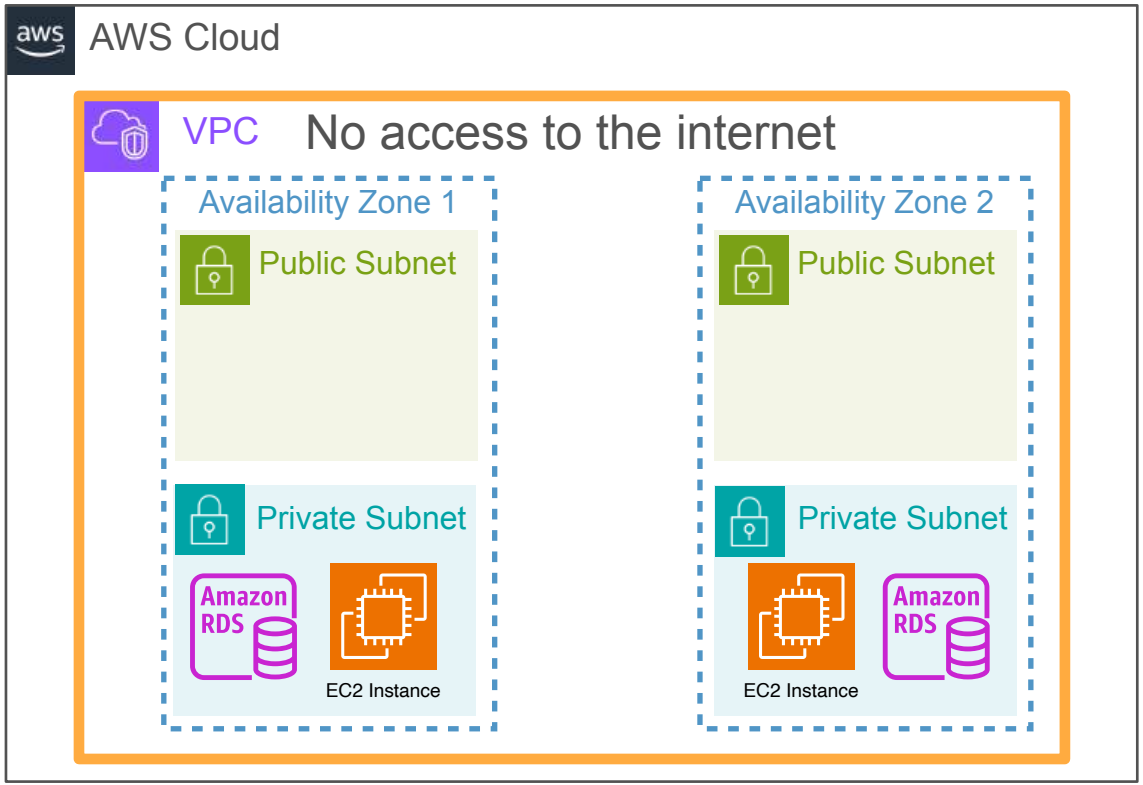


EC2 Instance

10.0._._

0 to 255

AWS Networking - VPCs & Subnets

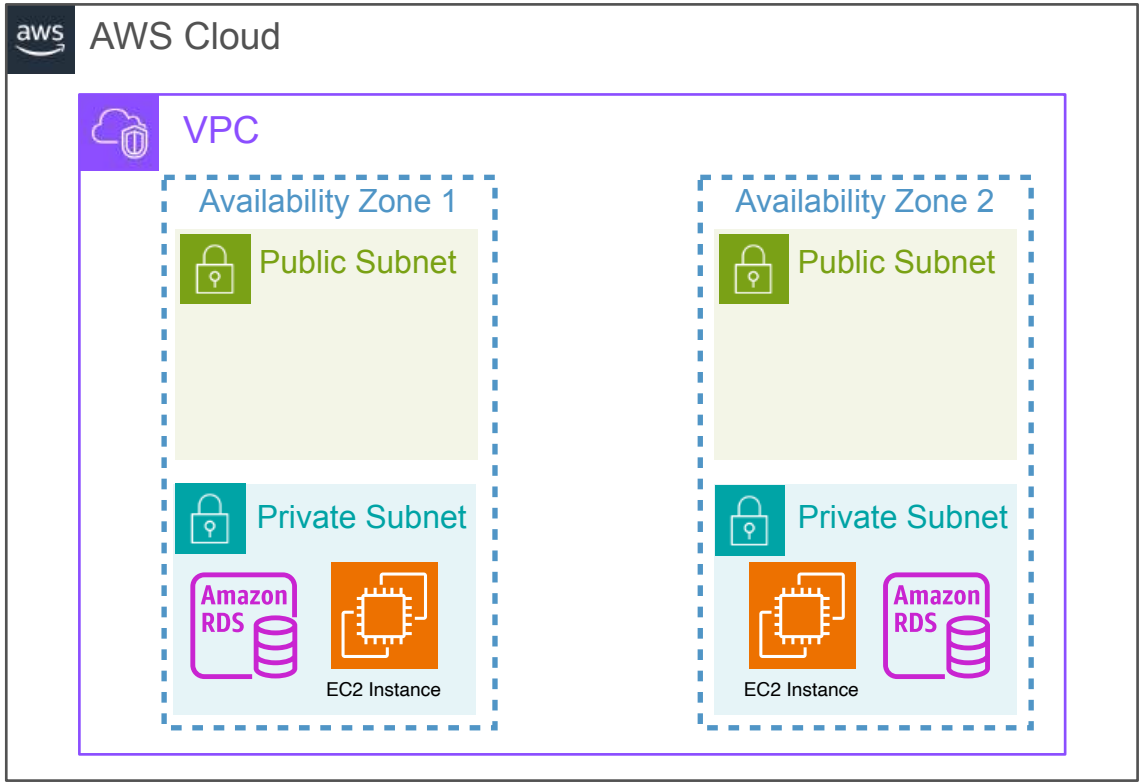


Closed network

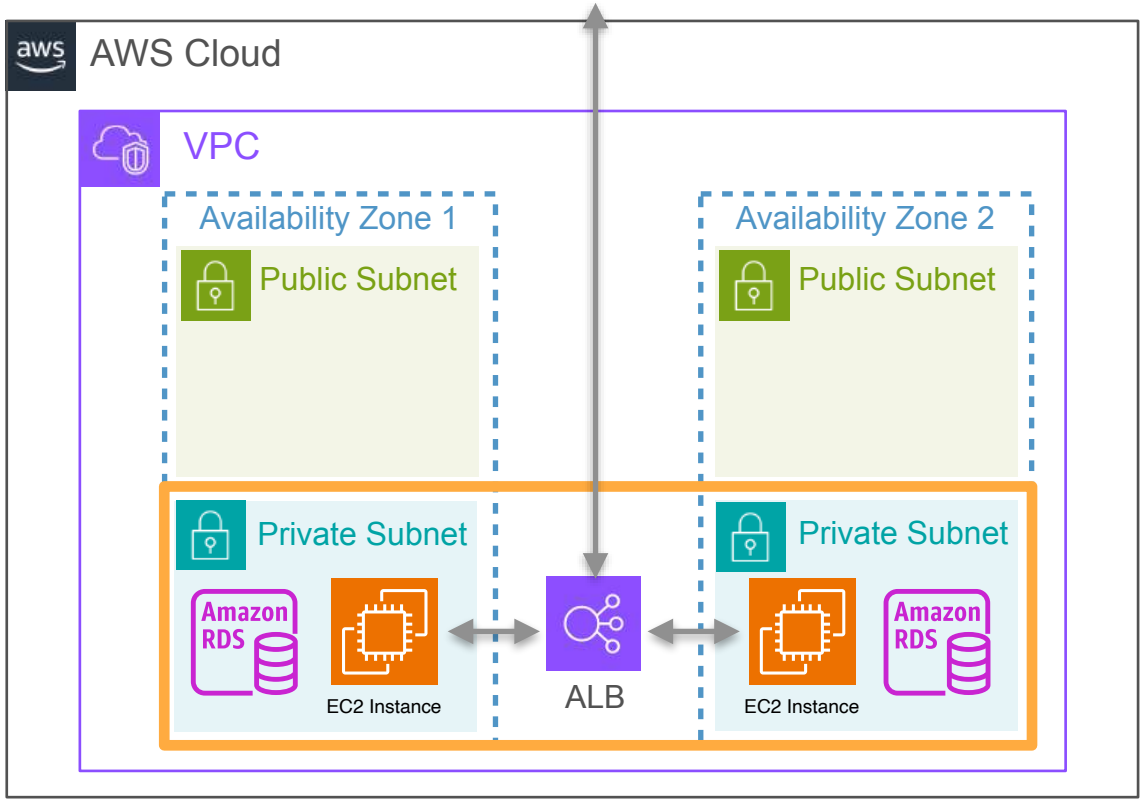
DeepLearning.AI

Interacting with Source Systems

AWS Networking -
Internet Gateways & NAT Gateways

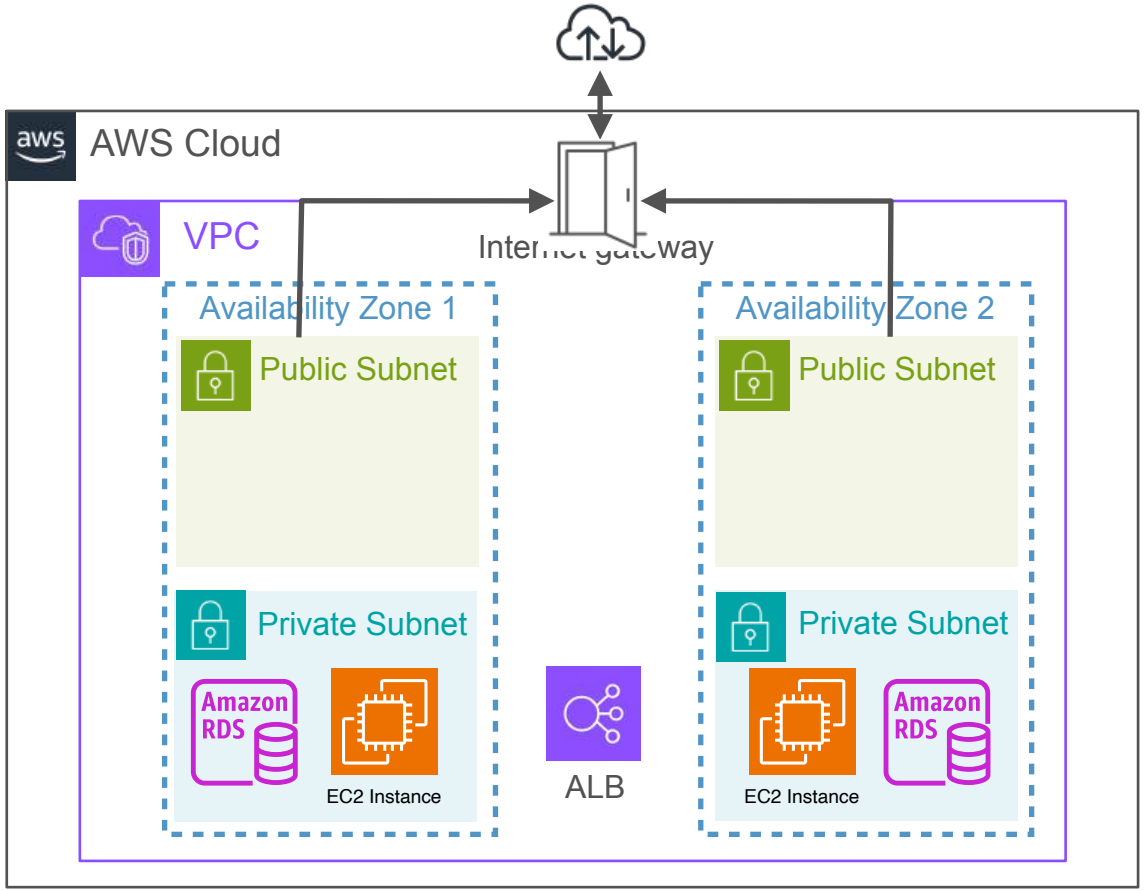

AWS Cloud
VPC
Availability Zone 1
Public Subnet
Private Subnet
Amazon RDS
EC2 Instance
Availability Zone 2
Public Subnet
Private Subnet
EC2 Instance
Amazon RDS

Example Scenario



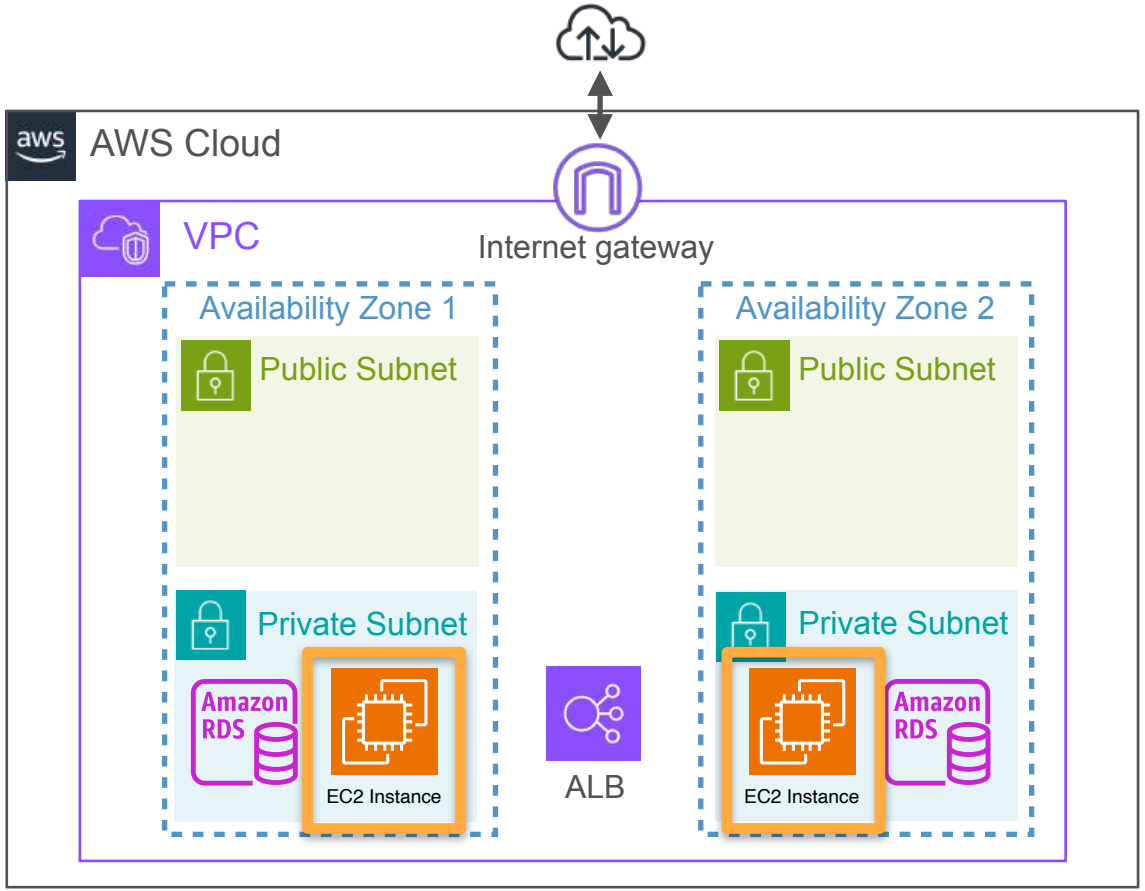
Considerations

1. Applications running on EC2 need to occasionally download updates from resources on the internet
   - Upgrades, patching
2. Need a way to submit requests to the application running on the EC2 instance



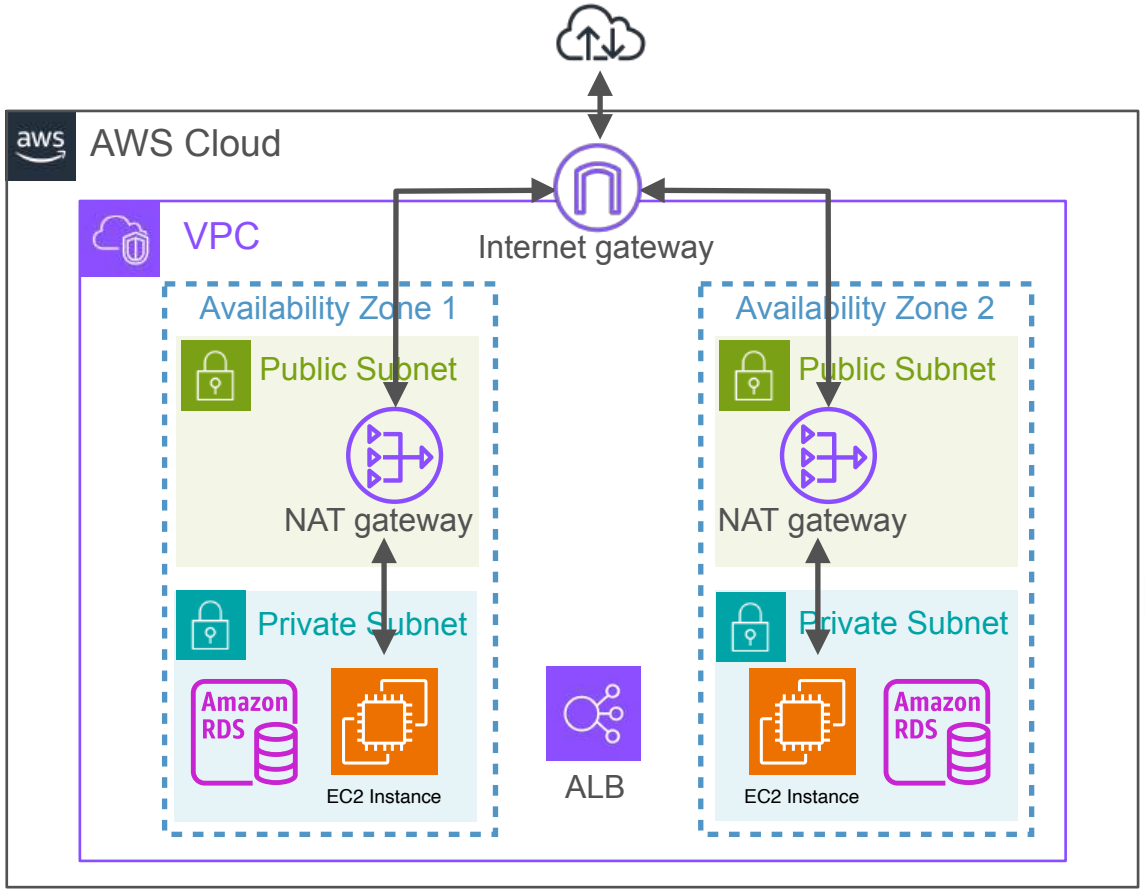
Considerations

1. Applications running on EC2 need to occasionally download updates from resources on the internet
   - Upgrades, patching
2. Need a way to submit requests to the application running on the EC2 instance



Considerations

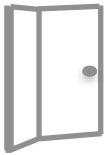
1. Applications running on EC2 need to occasionally download updates from resources on the internet
   - Upgrades, patching
2. Need a way to submit requests to the application running on the EC2 instance



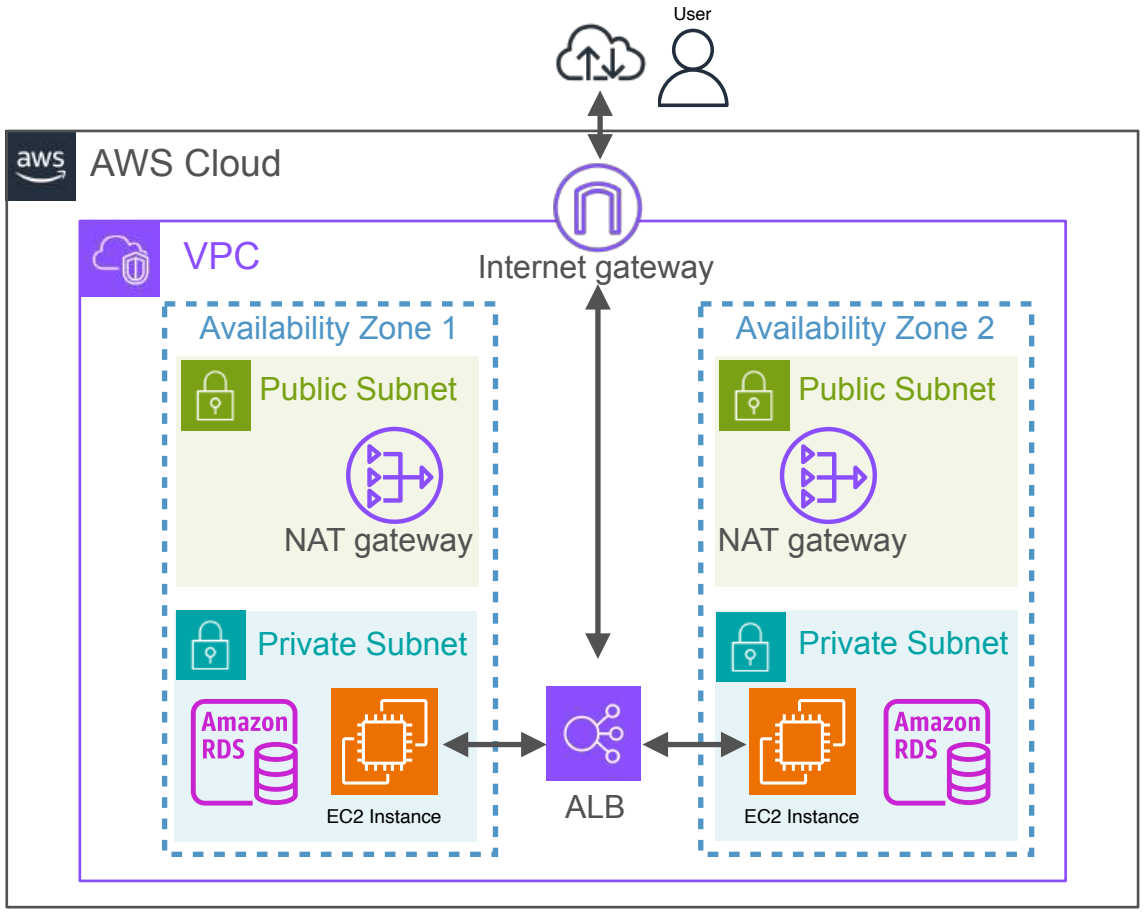
Considerations

1. Applications running on EC2 need to occasionally download updates from resources on the internet
   - Upgrades, patching
2. Need a way to submit requests to the application running on the EC2 instance

NAT Gateway **Network Address Translation Gateway**



- Allows resources in a private subnet to connect to the internet or other AWS services
- Prevents the internet from initiating connections with those resources

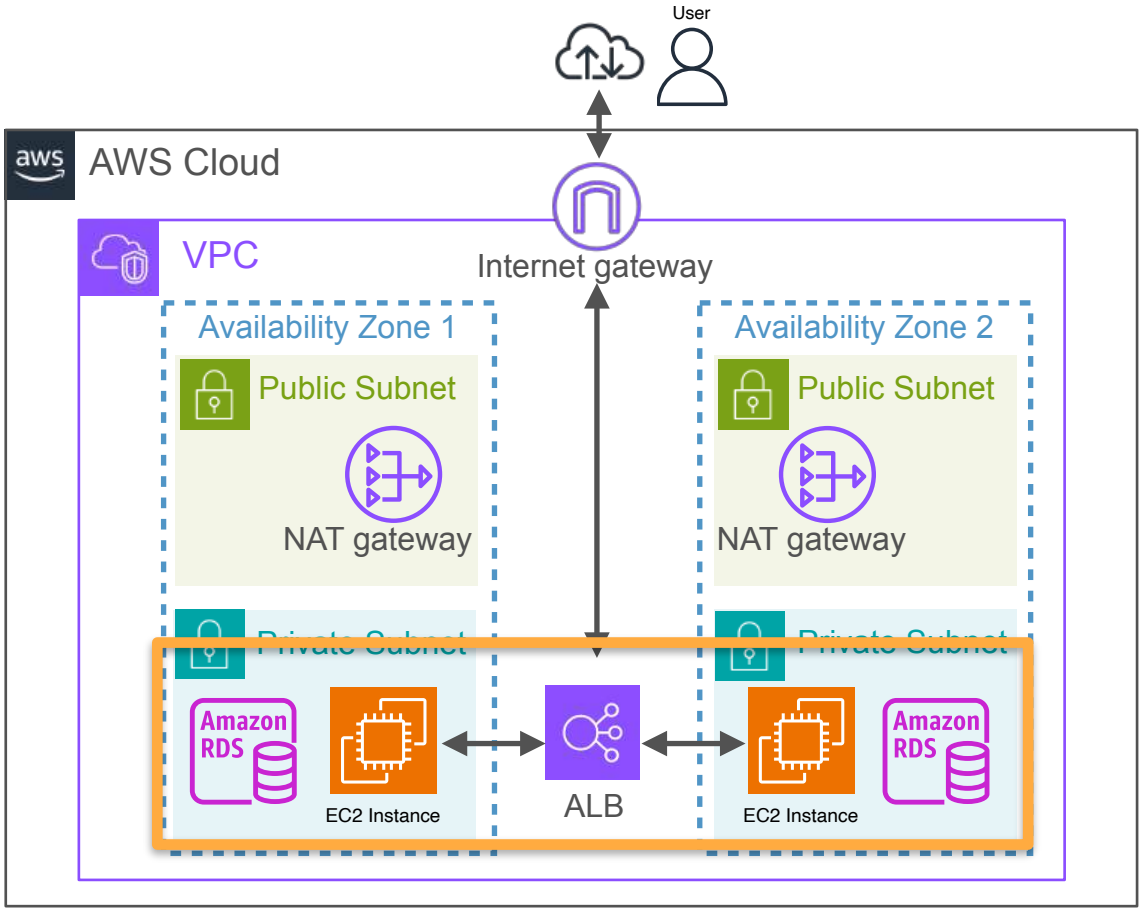


Considerations

1. Applications running on EC2 need to occasionally download updates from resources on the internet
   - Upgrades, patching
2. Need a way to submit requests to the application running on the EC2 instance

ALB:

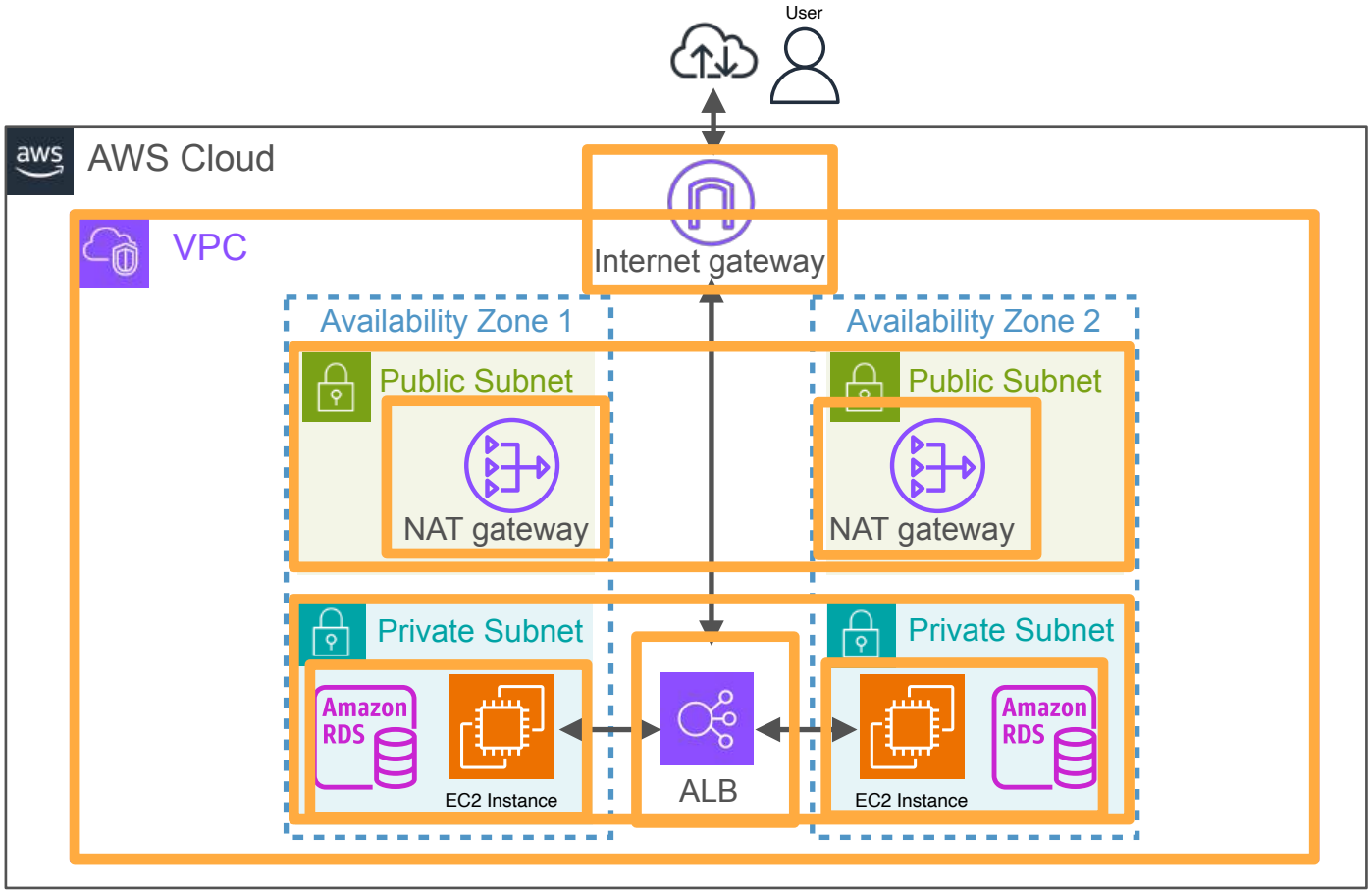
- Distributes incoming application traffic across multiple backend targets
- Handles the load and ensures the application remains responsive and available
- Keeps those EC2 instances private


User
AWS Cloud
VPC
Internet gateway
Availability Zone 1
Public Subnet
NAT gateway
Availability Zone 2
Public Subnet
NAT gateway
Amazon RDS
EC2 Instance
ALB
EC2 Instance
Amazon RDS

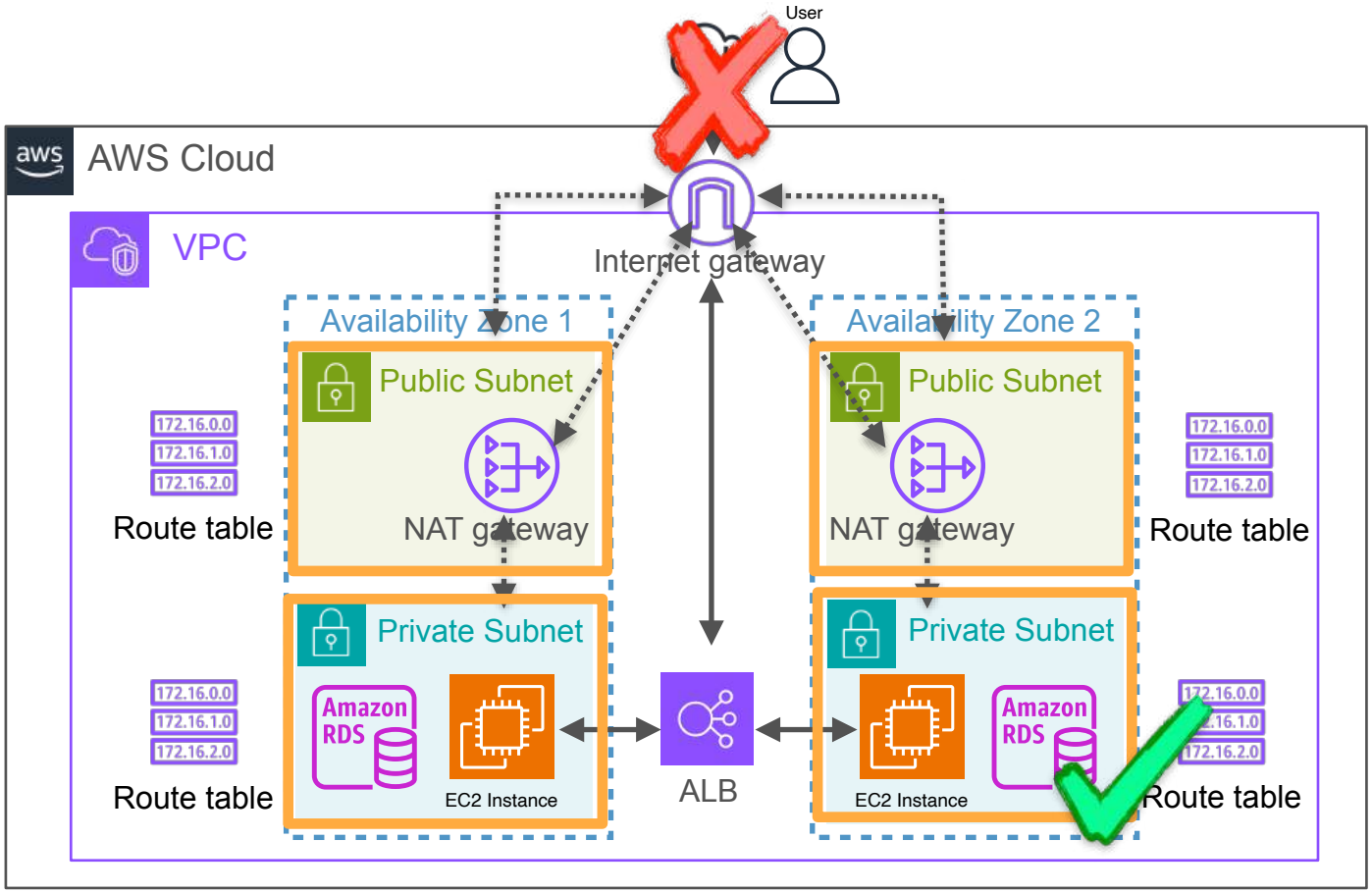
DeepLearning.AI

Interacting with Source Systems

AWS Networking - Route Tables

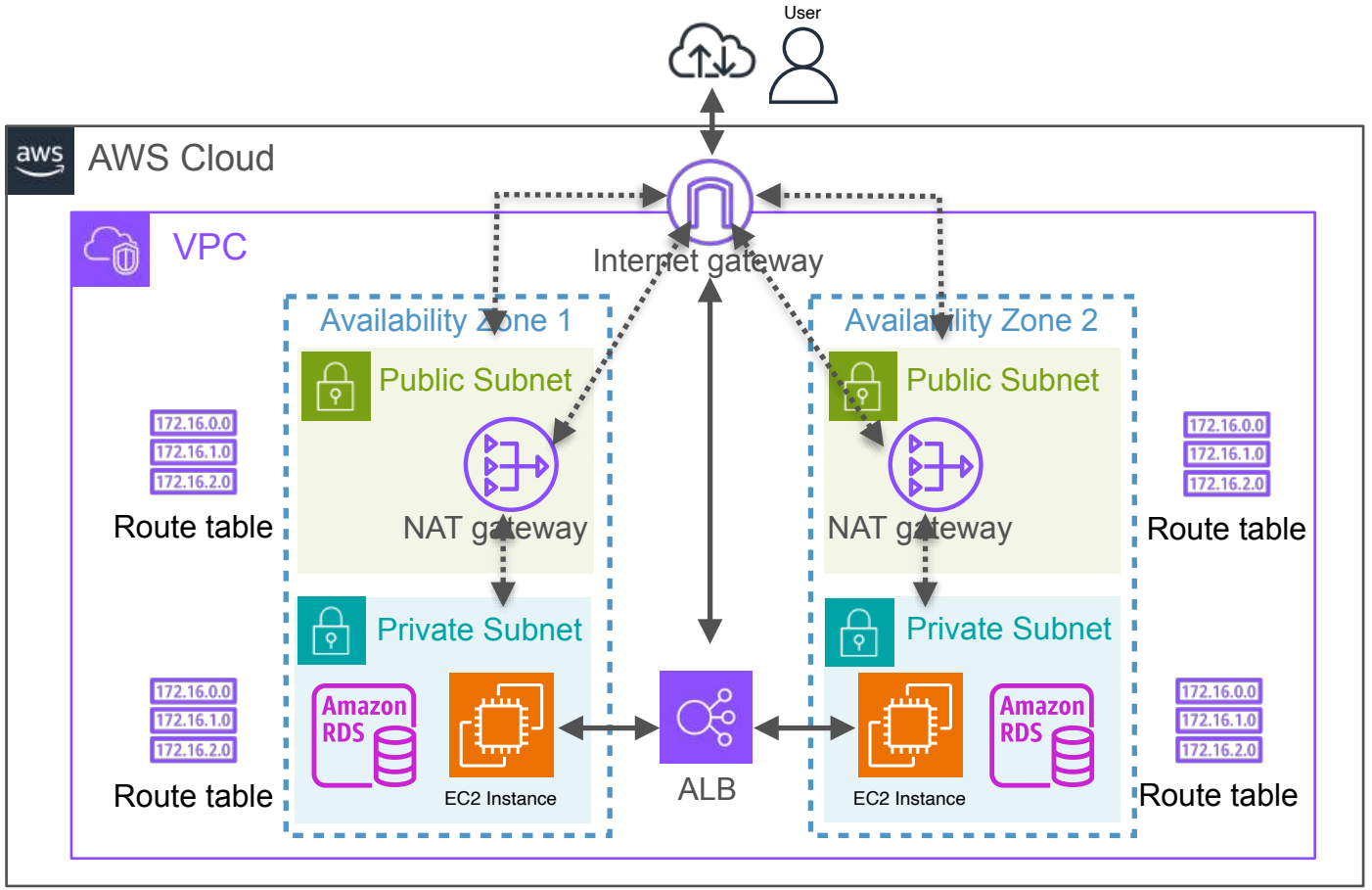


Route Tables



Route Tables

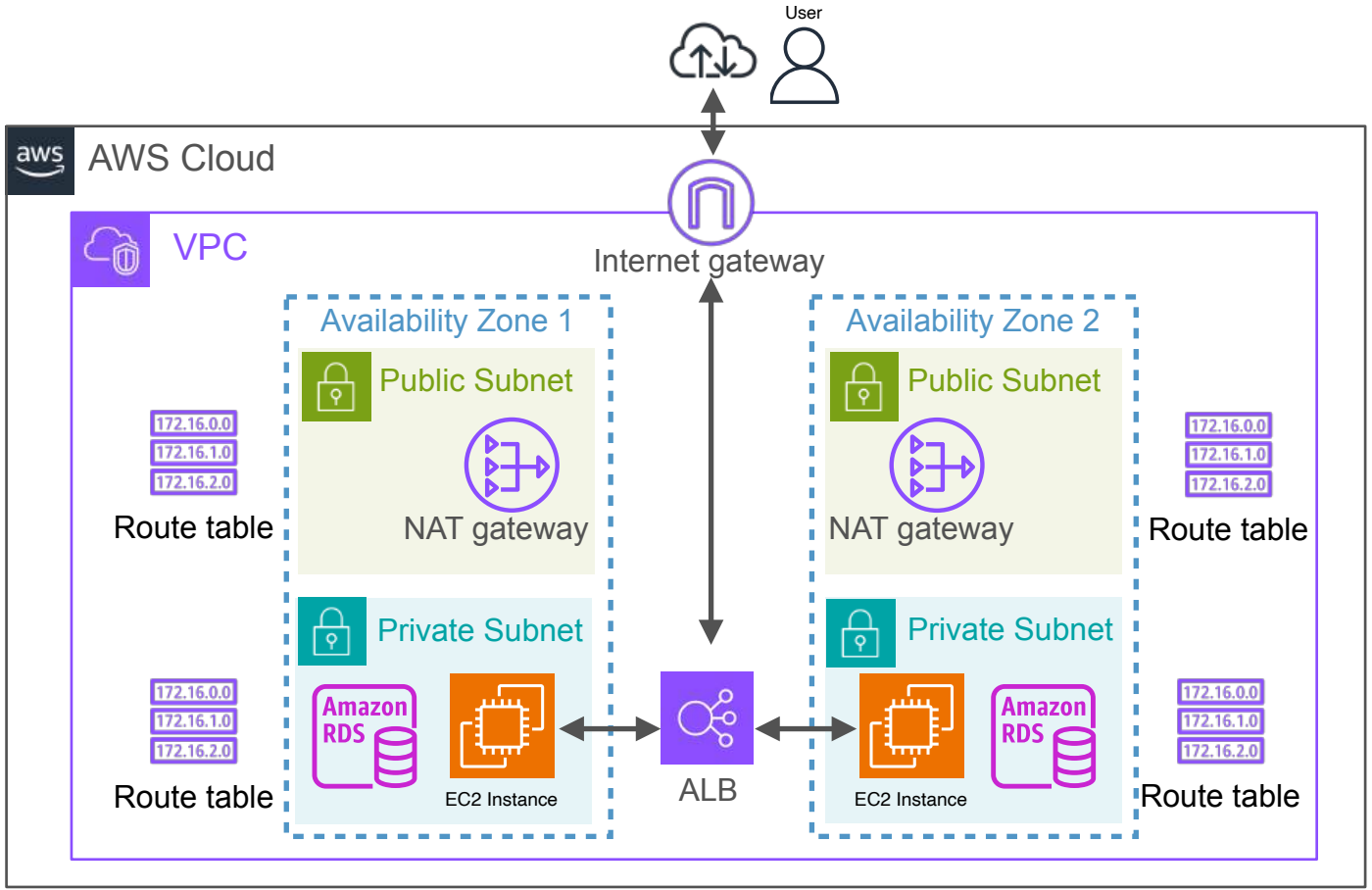
- Essential for directing network traffic within your VPC
- Default route table allows internal communication within the VPC


User
AWS Cloud
VPC
Internet gateway
Availability Zone 1
Availability Zone 2
Public Subnet
Public Subnet
NAT gateway
NAT gateway
172.16.0.0
172.16.1.0
172.16.2.0
Route table
Route table
Private Subnet
Private Subnet
Amazon RDS
Amazon RDS
EC2 Instance
EC2 Instance
ALB
Route table
Route table

DeepLearning.AI

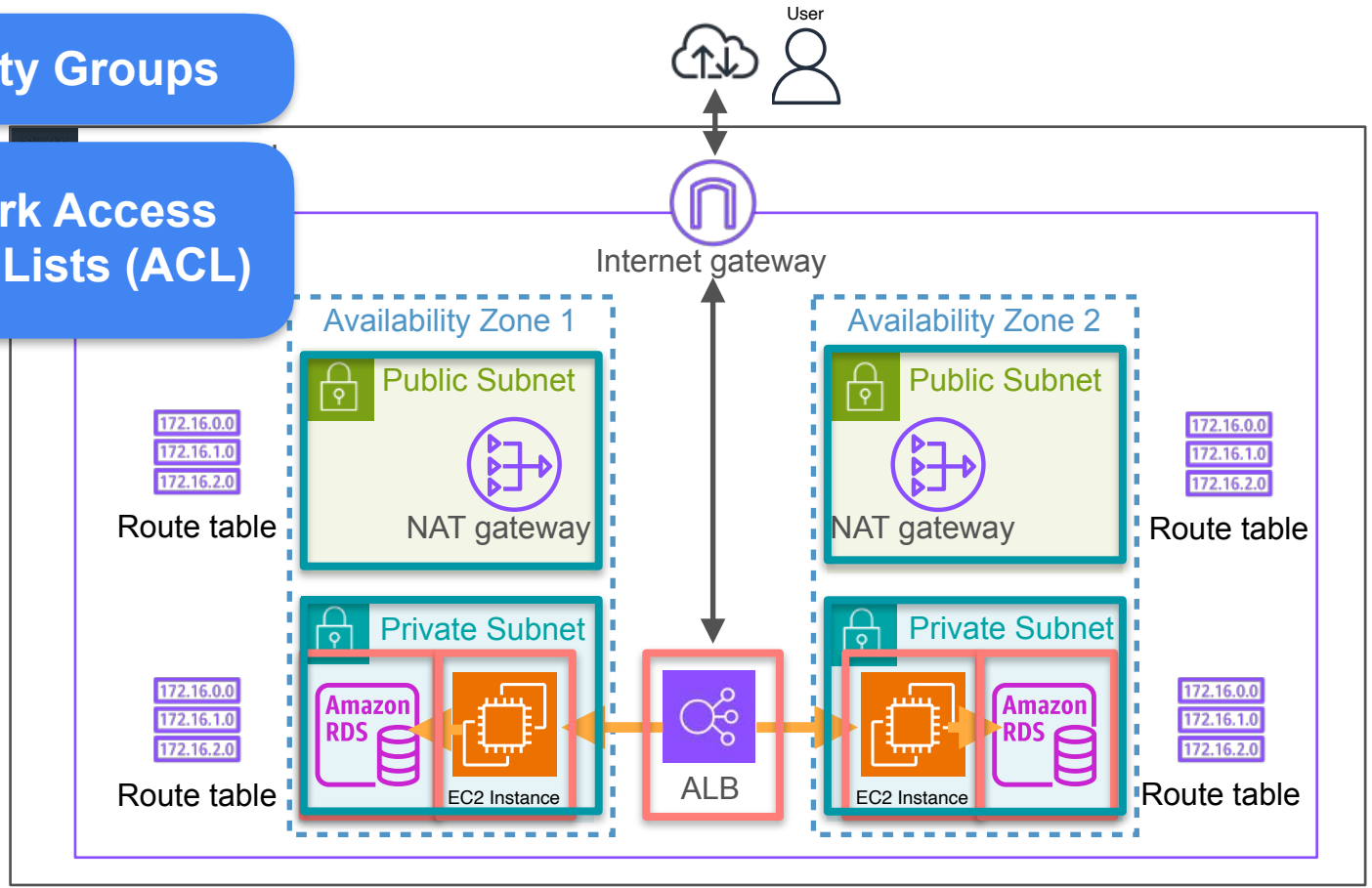
Interacting with Source Systems

AWS Networking - Network ACLs & Security Groups


User
AWS Cloud
VPC
Internet gateway
Availability Zone 1
Public Subnet
NAT gateway
Private Subnet
Amazon RDS
EC2 Instance
Availability Zone 2
Public Subnet
NAT gateway
Private Subnet
Amazon RDS
EC2 Instance
ALB
172.16.0.0
172.16.1.0
172.16.2.0
Route table
Route table
Route table
Route table

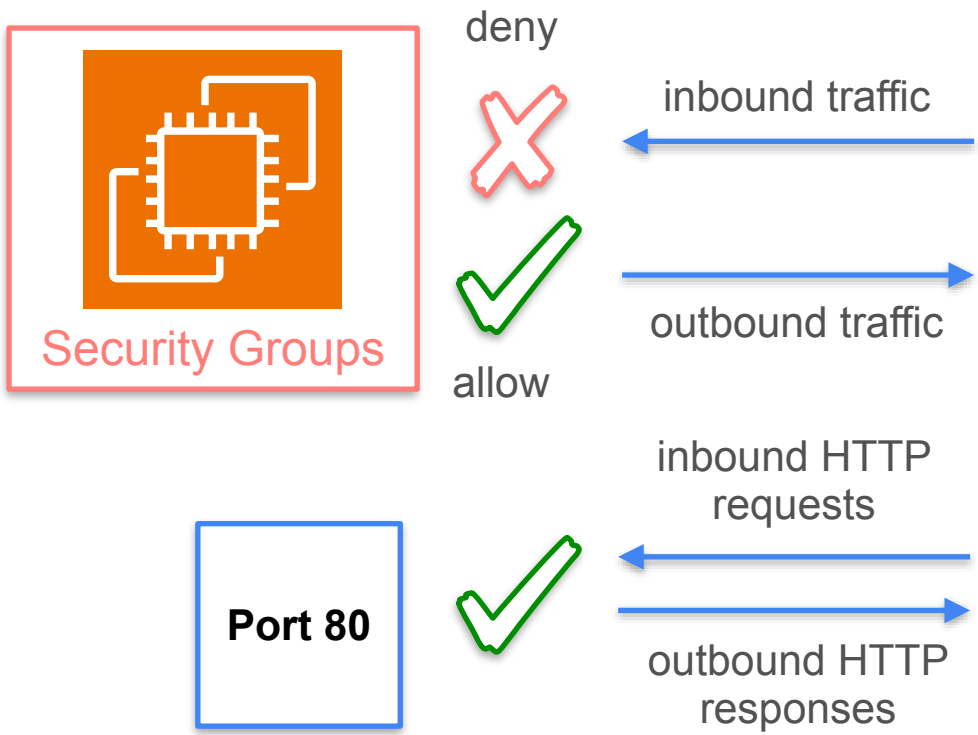
Security Groups

Network Access Control Lists (ACL)



Security Groups

Instance level virtual firewalls, controlling both inbound and outbound traffic



Inbound Rules

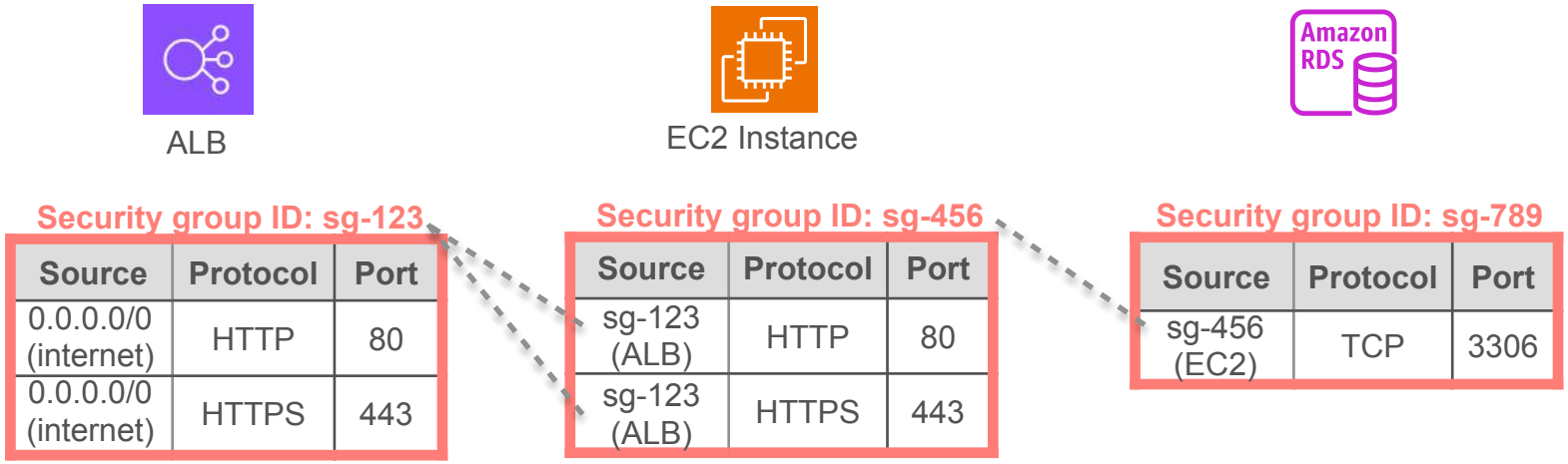
- Determine what types of traffic you want to allow
- Where you want to allow that traffic to come from

Security Groups are stateful

- Allow inbound traffic to an instance automatically allows the return traffic

Security Groups

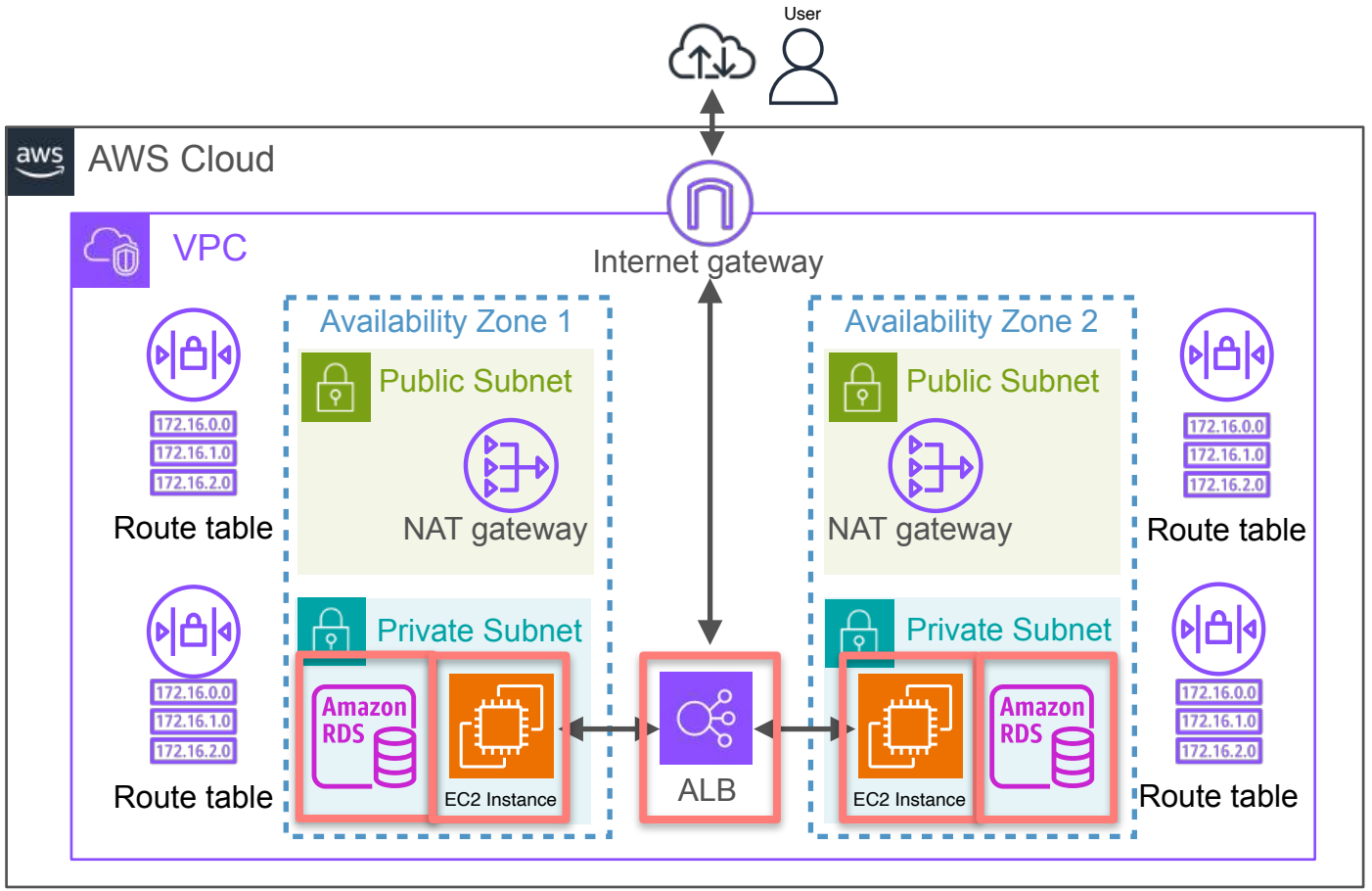
Instance level virtual firewalls, controlling both inbound and outbound traffic



Security Group Chaining

Network Access Control Lists (ACL)

- They provide an additional layer of security at the subnet level
- Network ACLs are stateless
- You need to define both inbound and outbound rules explicitly
- Useful for implementing security policies at the subnet level


User
AWS Cloud
VPC
Internet gateway
Availability Zone 1
Public Subnet
NAT gateway
Private Subnet
Amazon RDS
EC2 Instance
Availability Zone 2
Public Subnet
NAT gateway
Private Subnet
Amazon RDS
EC2 Instance
ALB
172.16.0.0
172.16.1.0
172.16.2.0
Route table
Route table
Route table
Route table

VPCs and subnets

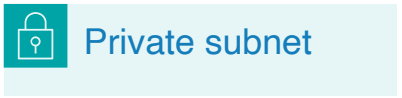
- Give you a way to define a private network on AWS.

Route Tables

- Direct traffic within the VPC and to the internet.

Public Subnets

Private Subnets



Internet Gateway

- Allow resources within public subnets to access the internet.

NAT Gateway

- Enable instances to initiate outbound connections securely.

Security Groups

- They act as virtual firewalls at the instance level
- They control both inbound and outbound traffic
- They are stateful

Network ACLs

- They provide an additional layer of security at the subnet level
- They are stateless, ie. require explicit rules for both inbound and outbound traffic

If you encounter connectivity issues:

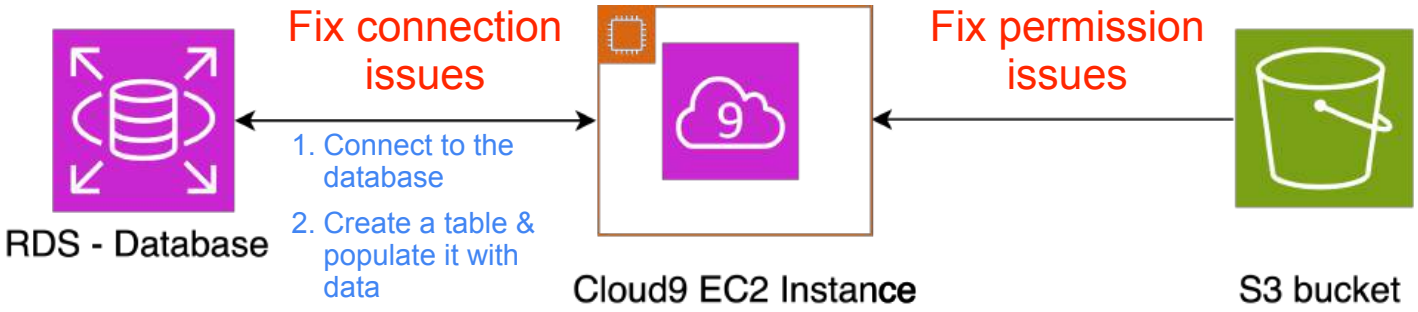
1. Verify that your VPC has an internet gateway properly attached
2. Verify that the route tables have appropriate rules to direct traffic correctly
3. Verify that the route table associations with the subnets are configured correctly
4. Check security groups to make sure they have the needed rules in place
5. Review network ACLs to confirm they allow the necessary traffic
6. Double-check instance configurations to ensure they are associated with the correct security groups and subnets

DeepLearning.AI

Connecting to Source Systems

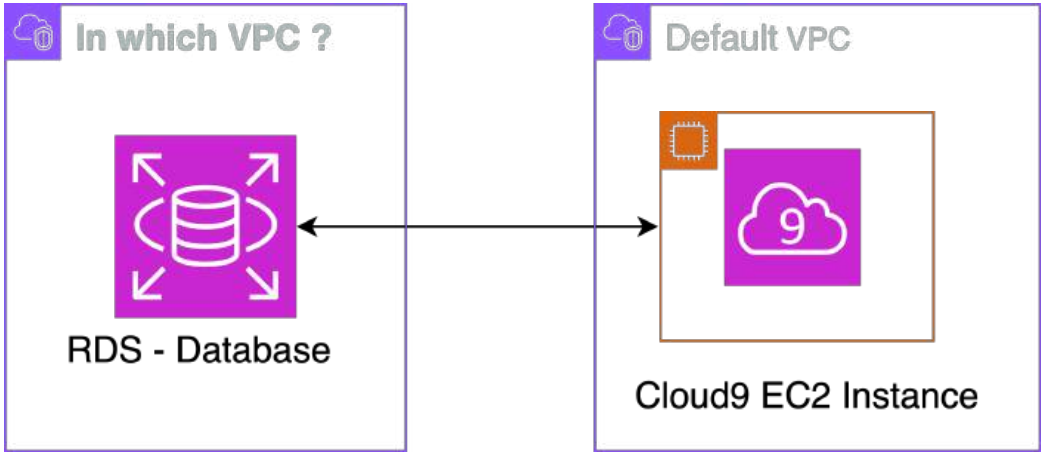
Lab Walkthrough - Database Connectivity and Troubleshooting on AWS

Database Connectivity and Troubleshooting on AWS



- Skip this video, jump straight into the lab and go for it
  - The lab instructions contain hints
- Or, start the lab and follow along with me as you go through this video.
  - When an issue occurs, I'll be inviting you to pause the video
  - After that, I'll show you how to fix it.

Database Connectivity and Troubleshooting on AWS



DeepLearning.AI

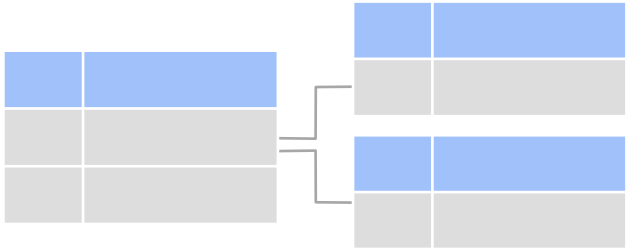
Working with Source Systems

Week 1 Summary

Week 1 Summary

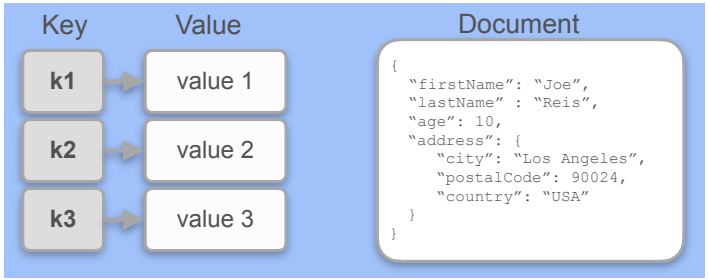
Understand how source systems work

Relational databases



Object Storage

NoSQL databases



Logs

user id	action	status	timestamp
67945	user added a product x to their cart	success	01-01-2025:10.30
38910	invalid values typed for product quantity	fail	01-01-2025:10.32

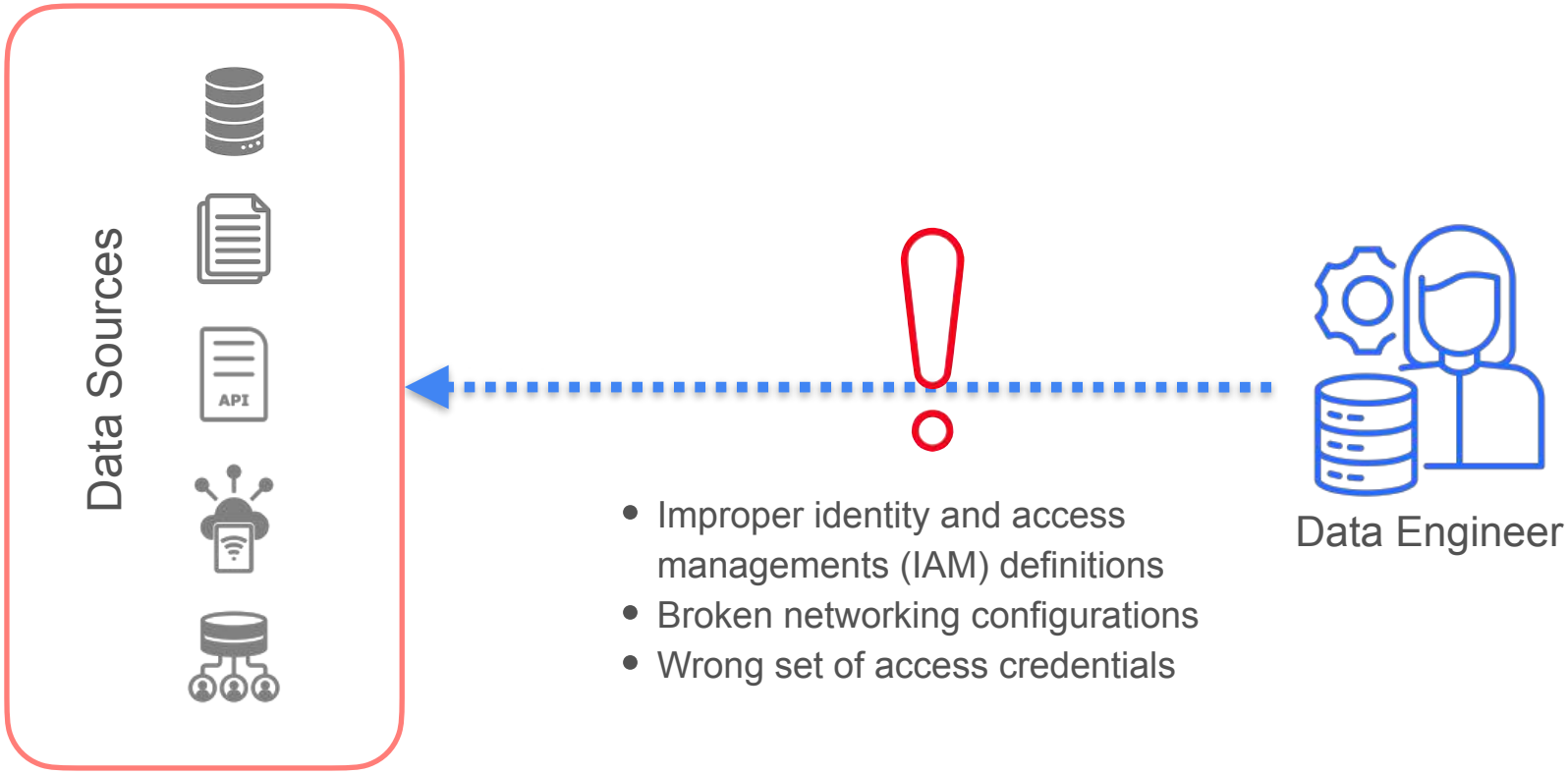
Streaming Systems

Event Router / Streaming Broker

Week 1 Summary

- How to connect to data sources
- Basics of networking
- Importance of IAM in ensuring security in source systems

Week 1 Summary



- Improper identity and access managements (IAM) definitions
- Broken networking configurations
- Wrong set of access credentials