Introduction to Data Engineering

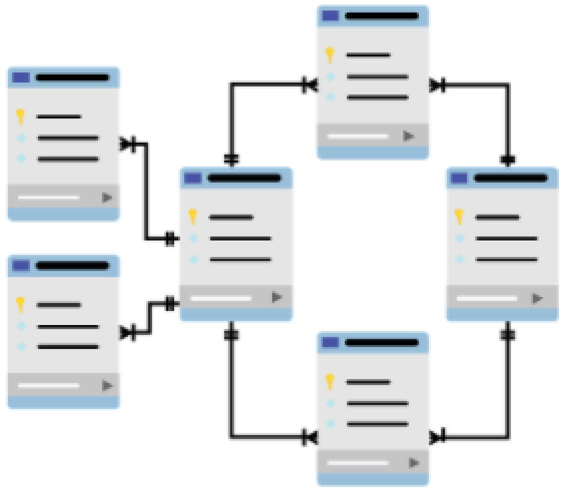
Introduction to Data Engineering

Black Raven (James Ng)
26 Mar 2021 · 31 min read

4 hours 15 Videos 57 Exercises 52,104 Participants 4,100 XP

This is a memo to share what I have learnt in Introduction to Data Engineering, capturing the learning objectives as well as my personal notes. The course is taught by Vincent Vankrunkelsven from DataCamp, and it includes 4 chapters:

Chapter 1. Introduction to Data Engineering
Chapter 2. Data engineering toolbox
Chapter 3. Extract, Transform and Load (ETL)
Chapter 4. Case Study: Data Engineering at DataCamp



Have you heard people talk about data engineers and wonder what it is they do? Do you know what data engineers do but you're not sure how to become one yourself? This course is the perfect introduction. It touches upon all things you need to know to streamline your data processing.

This introductory course will give you enough context to start exploring the world of data engineering. It's perfect for people who work at a company with several data sources and don't have a clear idea of how to use all those data sources in a scalable way. Be the first one to introduce these techniques to your company and become the company star employee.

Chapter 1. Introduction to Data Engineering

In this first chapter, you will be exposed to the world of data engineering! Explore the differences between a data engineer and a data scientist, get an overview of the various tools data engineers use and expand your understanding of how cloud technology plays a role in data engineering.

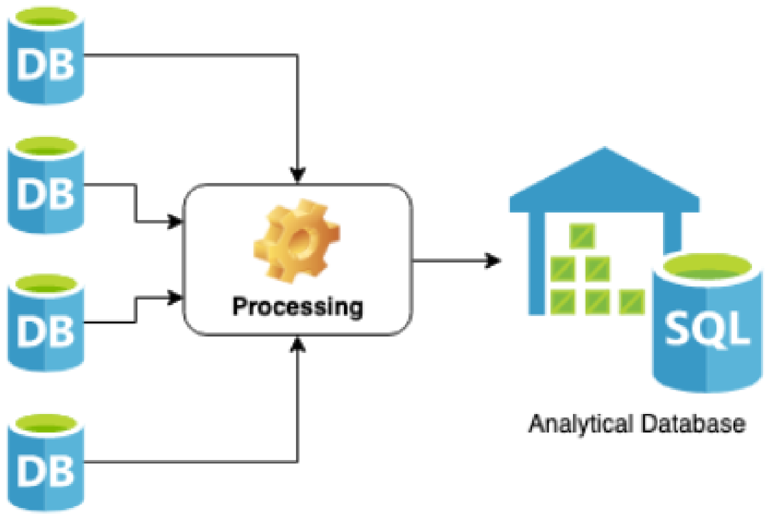
What is data engineering?

Common real world situations

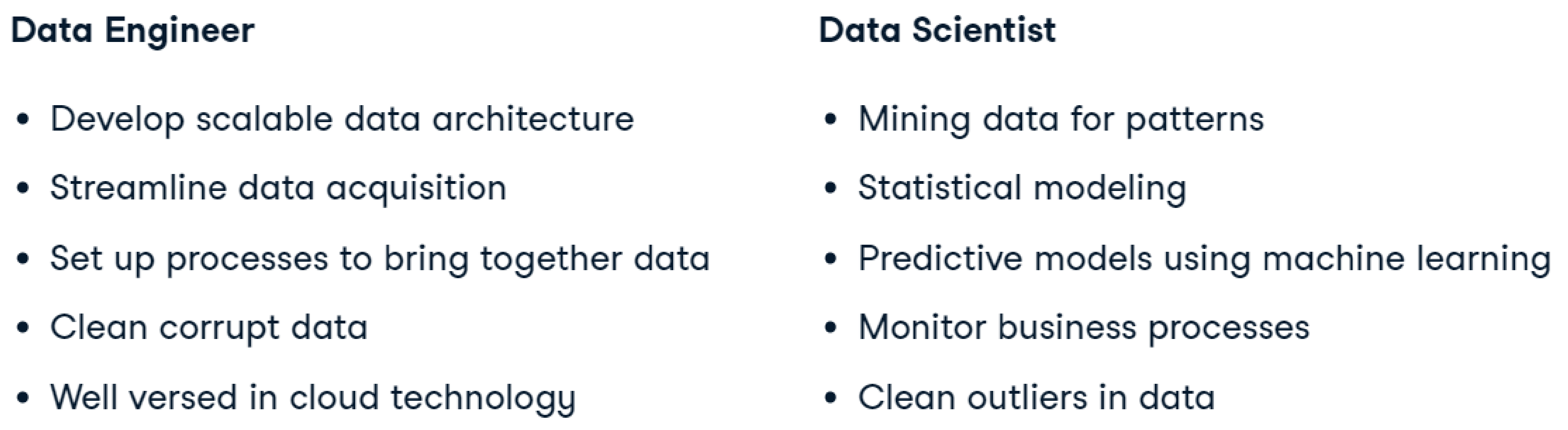
- Data is scattered
- Not available in an orderly fashion
- Not optimized for analysis
- Legacy code is causing corrupt data

Aim of data engineer:

- Gather data from different sources
- Optimize database scheme so it becomes faster to query
- Remove corrupted data
- Processing large amounts of data
- Use of clusters of machines



Data engineer
= an engineer that develops, constructs, tests, and maintains architectures such as databases and large-scale processing systems



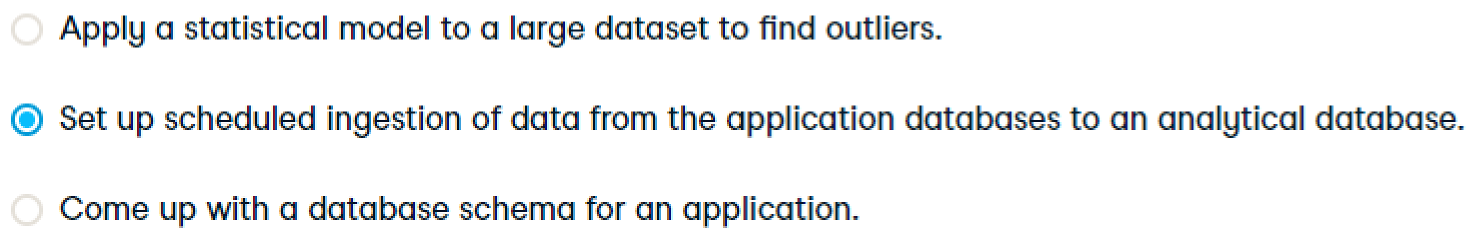
Data Engineer	Data Scientist
Develop scalable data architecture	Mining data for patterns
Streamline data acquisition	Statistical modeling
Set up processes to bring together data	Predictive models using machine learning
Clean corrupt data	Monitor business processes
Well versed in cloud technology	Clean outliers in data

Data Engineer has a deep understanding of cloud technology, whereas Data Scientist has a deep understanding of the business itself.

Tasks of the data engineer

The video presented several tasks of the data engineer. You saw that there are some differences between the tasks of data scientists and the tasks of data engineers.

Below are three essential tasks that need to happen in a data-driven company. Can you find the one that best fits the job of a data engineer?



- Apply a statistical model to a large dataset to find outliers.
- [✓ Set up scheduled ingestion of data from the application databases to an analytical database.]
- Come up with a database schema for an application.

This is very clearly the task of a data engineer.

Data engineer or data scientist?

You now know that there's a difference between the job of the data engineer and the data scientist. Remember the definition of a data engineer:

An engineer that develops, constructs, tests, and maintains architectures such as databases and large-scale processing systems.

Now let's see if you can separate tasks for the data engineer from the data scientist's tasks.

Data Engineer	Data Scientist
Set up processes to bring together data	Mining data for patterns
Develop scalable data architecture	Predictive models using machine learning
Streamline data acquisition	Statistical modeling
Clean corrupt data	Monitor business processes
Cloud technology	Clean statistical outliers in data

You classified all cards correctly!

Data engineering problems

For this exercise, imagine you work in a medium-scale company that hosts an online market for pet toys. As the company is growing, there are unmistakably some technical growing pains.

As the first data engineer, you observe some problems and have to decide where you're best suited to be of help.

[✓ Data scientists are querying the online store databases directly and slowing down the functioning of the application since it's using the same database.]

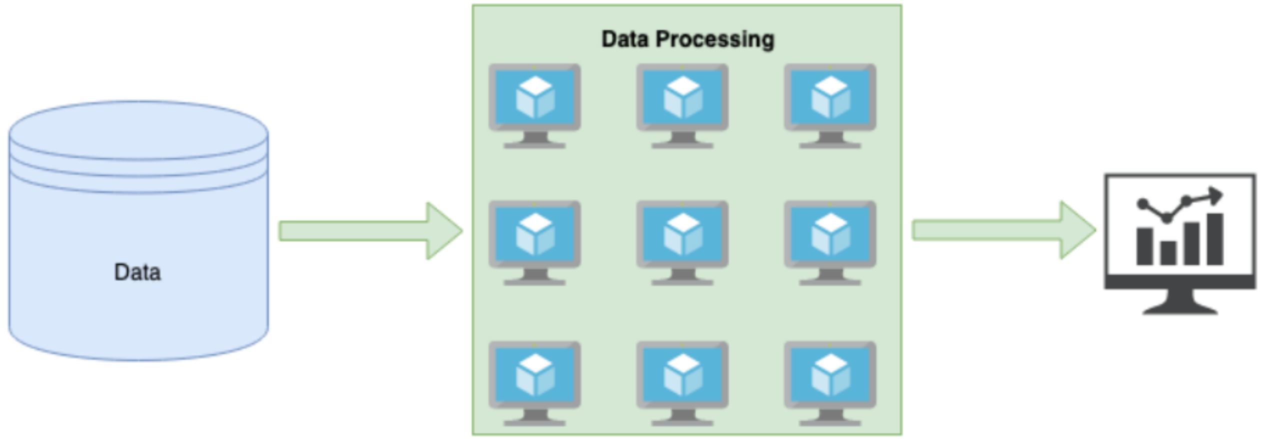
○ Harmful product recommendations are affecting the sales numbers of the online store.

○ The online store is slow because the application's database server doesn't have enough memory.

Tools of the data engineer

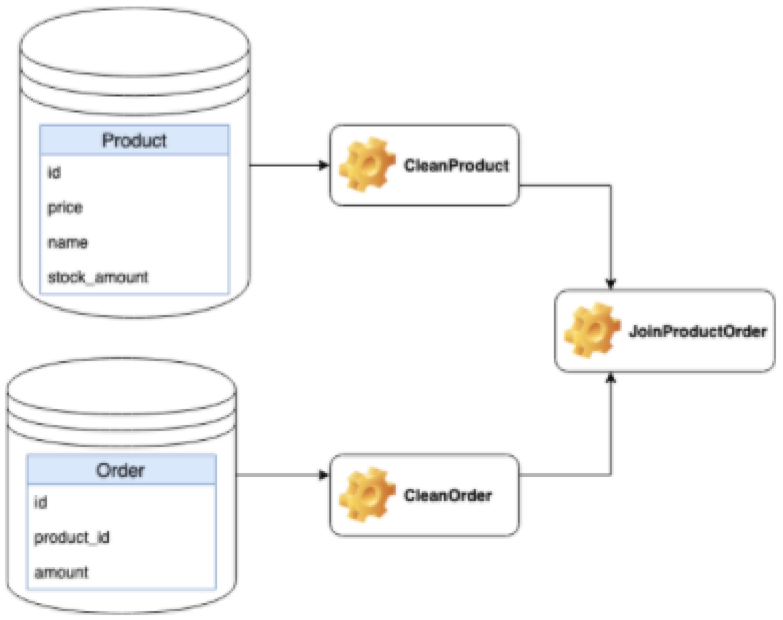
Database = computer system that holds large amounts of data
Applications rely on databases to provide certain functionality. For example, in an online store, a database holds product data like prices or amount in stock.
Other databases hold data specifically for analyses.

Data engineers use tools that can help them quickly process data. Processing data might be necessary to **clean** or **aggregate** data or to **join** it together from different sources. Typically, huge amounts of data have to be processed. That is where parallel processing comes into play. Instead of processing the data on one computer, data engineers use clusters of machines to process the data.



Often, these tools make an abstraction of the underlying architecture and have a simple API. A good data engineer understands these abstractions and knows their limitations.

```
df = spark.read.parquet("users.parquet")

outliers = df.filter(df["age"] > 100)

print(outliers.count())
```



Scheduling tools help to make sure data moves from one place to another at the correct time, with a specific interval. Data engineers make sure these processing jobs run in a timely fashion and that they run in the right order .

Databases

MySQL

PostgreSQL

Scheduling

Apache Airflow

OOZIE

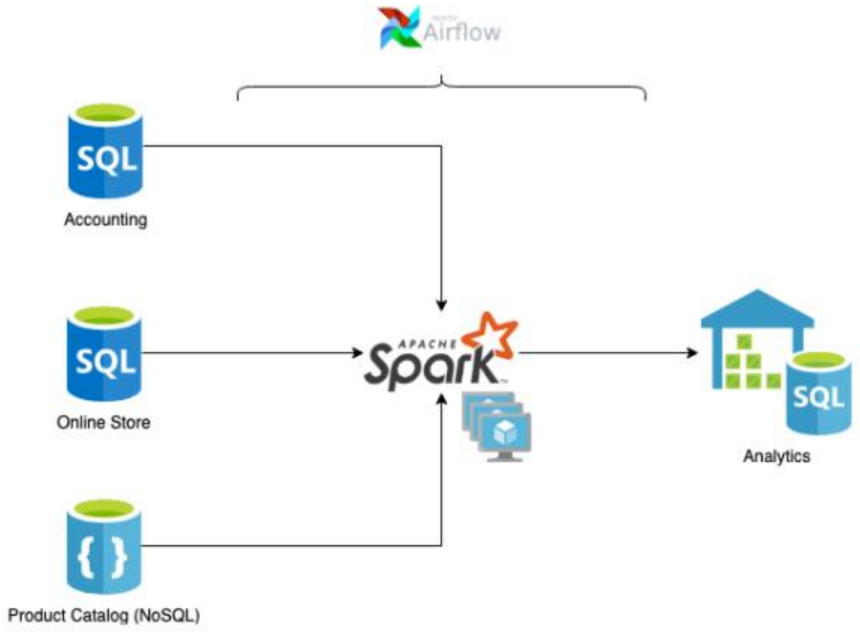
Bash tool: cron

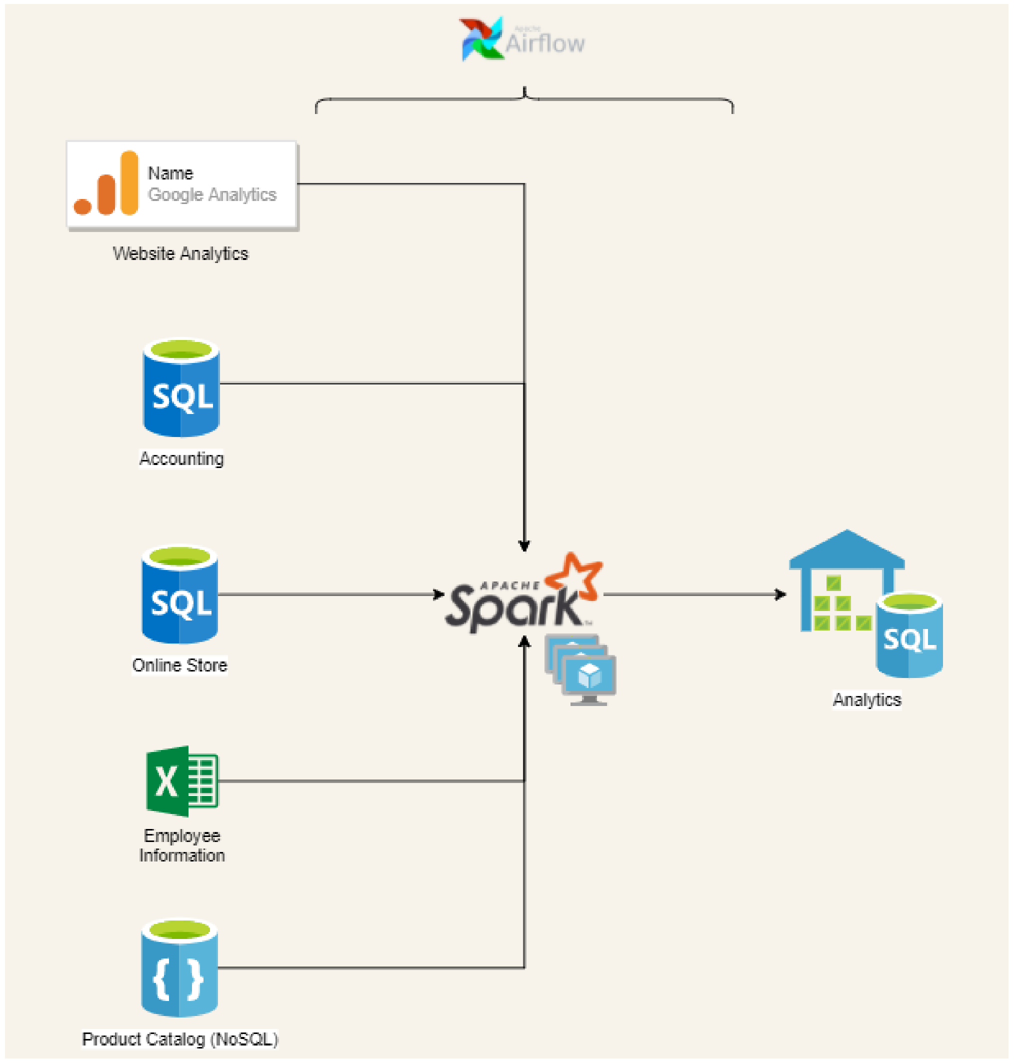
Processing

Apache Spark

HIVE

To sum everything up, you can think of the data engineering pipeline through this diagram. It extracts all data through connections with several databases, transforms it using a cluster computing framework like Spark, and loads it into an analytical database. Also, everything is scheduled to run in a specific order through a scheduling framework like Airflow. A small side note here is that the sources can be external APIs or other file formats too.





Kinds of databases

In the video, you saw that databases are an essential tool for data engineers. The data engineer's workflow often begins and ends with databases.

However, databases are not always the first step. Sometimes, data has to be pulled in from external APIs or raw files. Can you identify the database in the schematics?

- All database nodes are on the left.
- All nodes on the left and the analytics node on the right are databases.
- [✓ Accounting, Online Store, Product Catalog, and Analytics are databases.]

The analytics database is also a SQL database. All but the product catalog are SQL databases.

Processing tasks

Data engineers often have to join, clean, or organize data before loading it into a destination analytics database. This is done in the data processing, or data transformation step.

In the diagram, this is the intermediate step. Can you select the **most correct** statement about data processing?

- Data processing is often done on a single, very powerful machine.
- Data processing is distributed over clusters of virtual machines.
- Data processing is often very complicated because you have to manually distribute workload over several computers.

Answer: Data processing is distributed over clusters of virtual machines.
This makes it very easy to scale. If things are slowing down, assign some more virtual machines to the job.

Scheduling tools

The last piece of the puzzle is the scheduler. In this example, Apache Airflow is used. You could see the scheduler as the glue of a data engineering system, holding each small piece together and organizing how they work together.

Do you know which one is NOT a responsibility of the scheduler?

○ Make sure jobs run in a specific order and all dependencies are resolved correctly.

○ Make sure the jobs run at midnight UTC each day.

○ Scale up the number of nodes when there's lots of data to be processed.

Answer: Scale up the number of nodes when there's lots of data to be processed.
The scheduler does not scale up processing power. That's the job of the processing tool.

Cloud providers

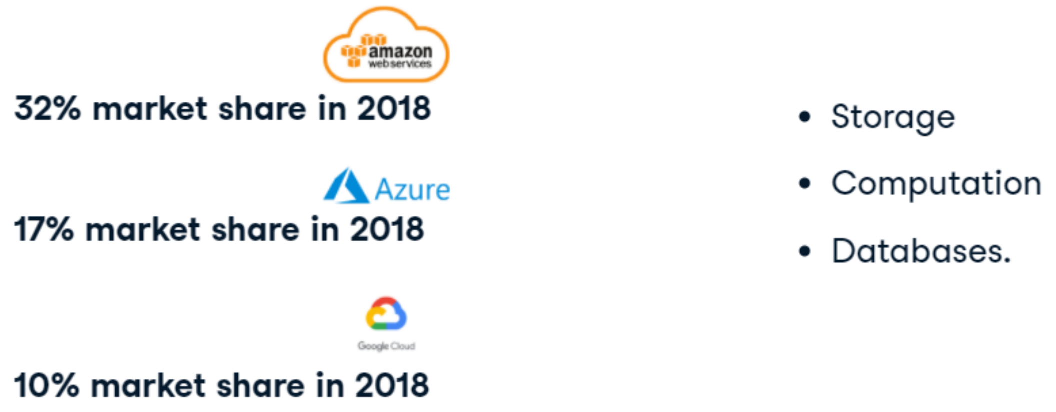
Data engineers are heavy users of the cloud, as data processing often runs on clusters of machines.

In the past, companies that relied on data processing owned their own data center. You can imagine racks of servers, ready to be used. The electrical bill and maintenance were also at the company's cost. Moreover, companies needed to be able to provide enough processing power for peak moments. That also meant that at quieter times, much of the processing power remained unused. It's this **waste of resources** that made cloud computing so appealing. In the cloud, you use the resources you need, at the time you need them. You can see that once these cloud services came to be, many companies moved to the cloud as a way of cost optimization. Apart from the costs of maintaining data centers, another reason for using cloud computing is **database reliability**.

Three big players in the cloud provider market:

1. **Amazon Web Services** or AWS. Think about the last few websites you visited. Chances are AWS hosts at least a few of them. Back in 2017, AWS had an outage, it reportedly 'broke' the internet. That's how big AWS is. AWS took up 32% of the market share in 2018.
2. **Microsoft Azure** is the second big player and took 17% of the market.
3. **Google Cloud** held 10% of the market in 2018.

These big players provide three types of services: Storage, Computation, and Databases.



32% market share in 2018

17% market share in 2018

10% market share in 2018

- Storage
- Computation
- Databases.

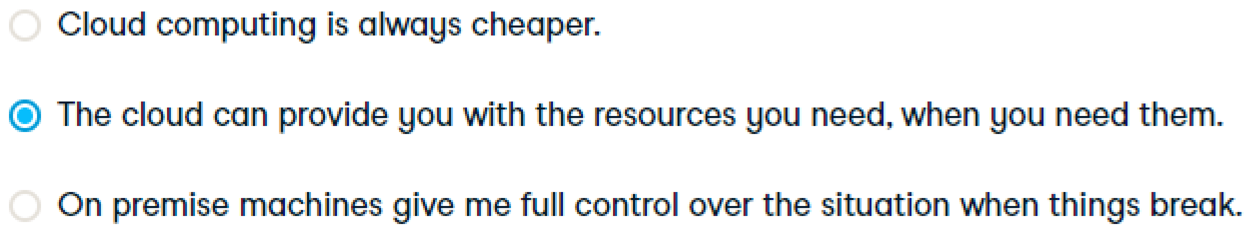
Storage
Storage services allow you to upload files of all types to the cloud. In an online store for example, you could upload your product images to a storage service. Storage services are typically very cheap since they don't provide much functionality other than storing the files reliably. AWS hosts **S3** as a storage service. Azure has **Blob Storage**, and Google has **Cloud Storage**.

Computation
Computation services allow you to perform computations on the cloud, example host web servers. These services are usually flexible, and you can start or stop virtual machines as needed. AWS has **EC2** as a computation service, Azure has **Virtual Machines**, and Google has **Compute Engine**.

Databases
Cloud providers host databases. For SQL databases, AWS has **RDS**. Azure has **SQL Database**, and Google has **Cloud SQL**.

Why cloud computing?

In the video you saw the benefits of using cloud computing as opposed to self-hosting data centers. Can you select the most **correct** statement about cloud computing?

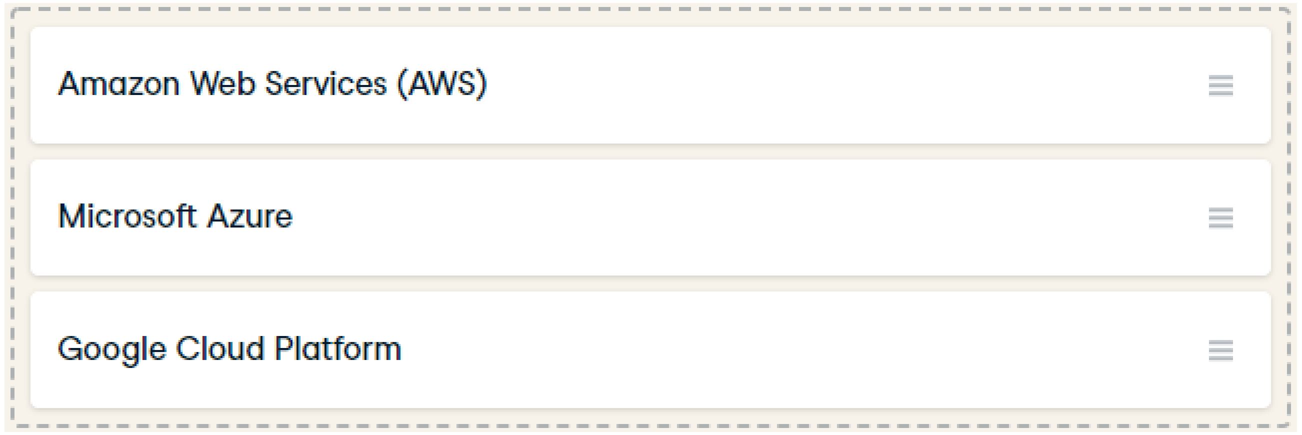

- Cloud computing is always cheaper.
- [✓ The cloud can provide you with the resources you need, when you need them.]
- On premise machines give me full control over the situation when things break.

This property of cloud computing is also called cloud elasticity.

Big players in cloud computing

You've seen a few examples of cloud service providers. You learned about the big three, who together took up more than 50% of the market share in 2018. They took 32%, 17%, and 10% of the market share.

Can you order the big three correctly?


1. Amazon Web Services (AWS)
2. Microsoft Azure
3. Google Cloud Platform

You'll see these names pop up in the data engineering world from time to time, so keep an eye out for them!

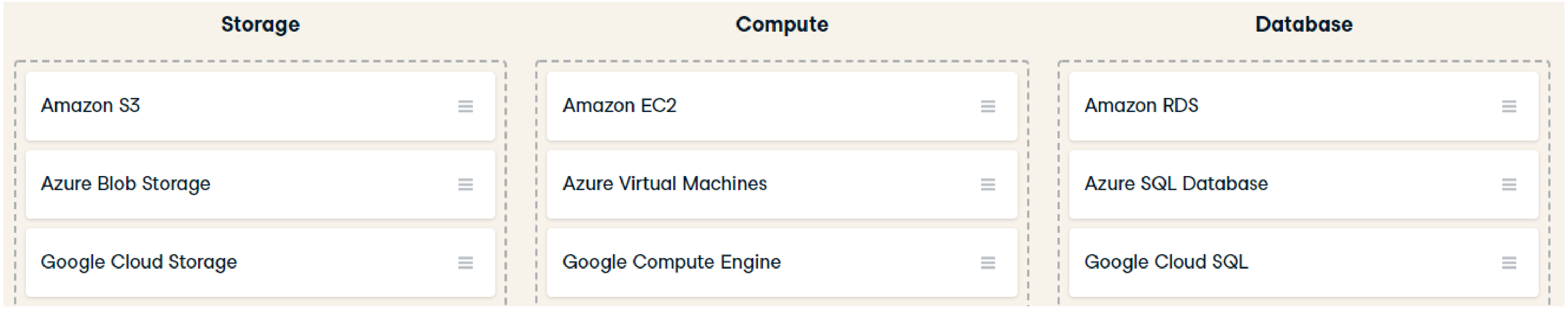
Cloud services

There are a bunch of cloud services that all have different use cases. It can be hard to keep track of all of them.

Here's an overview of the ones you've seen:

- **Storage**: use the cloud to store unstructured data, like files.
- **Compute**: use virtual machines in the cloud to do complex calculations.
- **Database**: use databases, typically SQL, in the cloud to hold structured data.

In addition, different cloud providers have different names for each of these services…



Storage	Compute	Database
Amazon S3	Amazon EC2	Amazon RDS
Azure Blob Storage	Azure Virtual Machines	Azure SQL Database
Google Cloud Storage	Google Compute Engine	Google Cloud SQL

All of these different services can be a bit overwhelming at first, but you'll learn the difference quickly and learn to appreciate the variations.

Chapter 2. Data engineering toolbox

Now that you know the primary differences between a data engineer and a data scientist, get ready to explore the data engineer's toolbox! Learn in detail about different types of databases data engineers use, how parallel computing is a cornerstone of the data engineer's toolkit, and how to schedule data processing jobs using scheduling frameworks.

Databases

What is a database?

- Holds data
- Organizes data
- Retrieve/Search data through Database Management System (DBMS)

> *A usually large collection of data organized especially for rapid search and retrieval.*

Databases

- Very organized
- Functionality like search, replication, ...

File storage

- Less organized
- Simple, less added functionality

Structured: database schema

- Relational database

Semi-structured

- JSON

```
{ "key": "value"}
```

Unstructured: schemaless, more like files

- Videos, photos

SQL

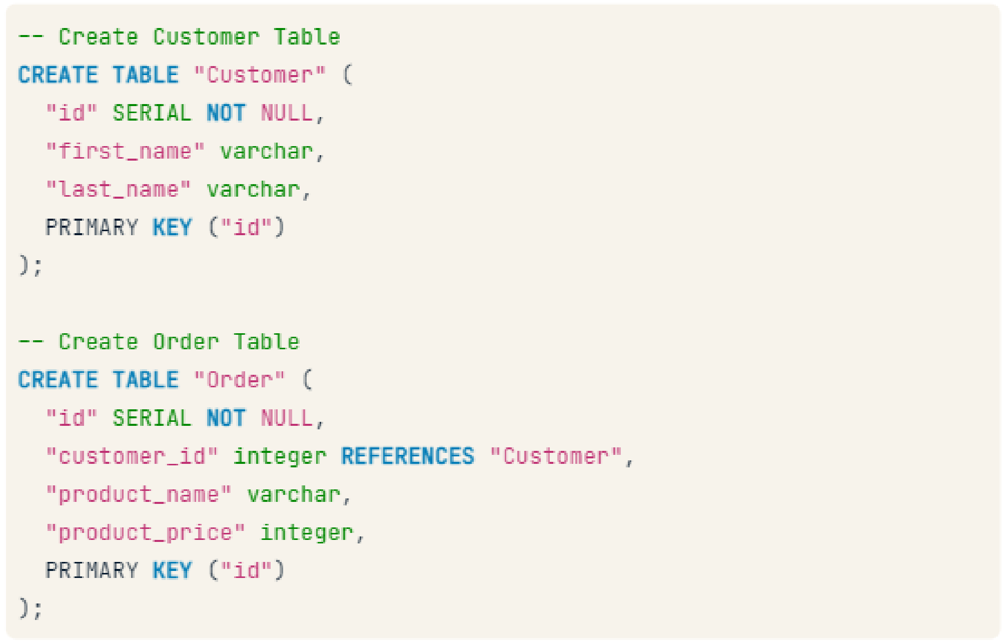
- Tables
- Database schema
- Relational databases

NoSQL

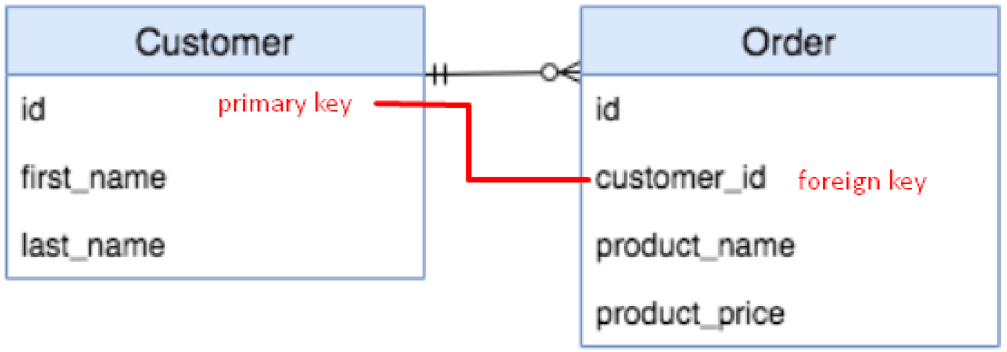
- Non-relational databases
- Structured or unstructured
- Key-value stores (e.g. caching)
- Document DB (e.g. JSON objects)

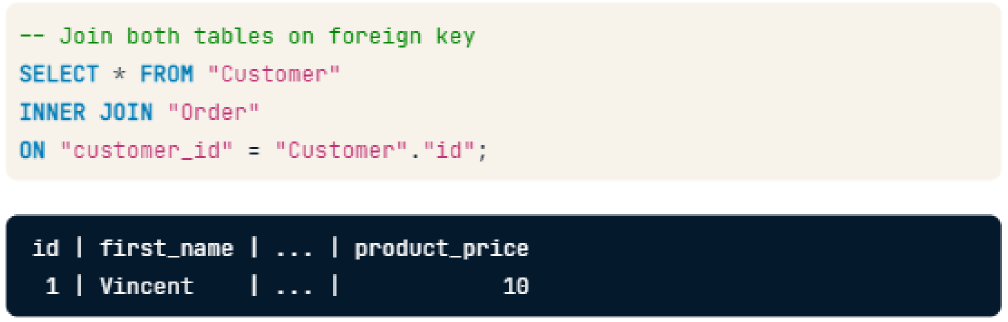
There are several types of NoSQL databases and they are not all unstructured. Two highly used NoSQL database types are key-value stores like Redis or document databases like MongoDB. In key-value stores, the values are simple. Typical use cases are caching or distributed configuration. Values in a document database are structured or semi-structured objects, for example, a JSON object.

The database schema
A **schema** describes the structure and relations of a database.



```
-- Create Customer Table
CREATE TABLE "Customer" (
  "id" SERIAL NOT NULL,
  "first_name" varchar,
  "last_name" varchar,
  PRIMARY KEY ("id")
);

-- Create Order Table
CREATE TABLE "Order" (
  "id" SERIAL NOT NULL,
  "customer_id" integer REFERENCES "Customer",
  "product_name" varchar,
  "product_price" integer,
  PRIMARY KEY ("id")
);
```



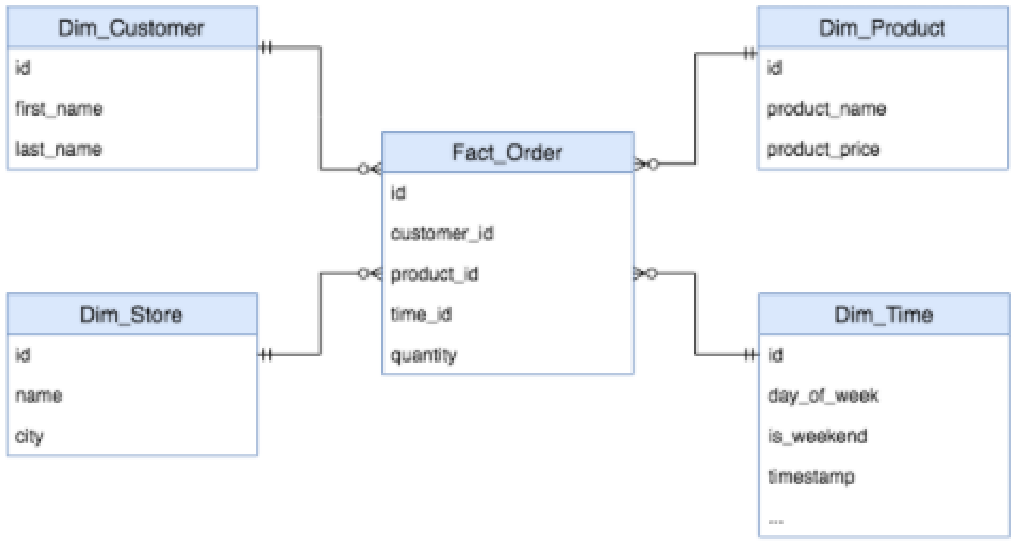


```
-- Join both tables on foreign key
SELECT * FROM "Customer"
INNER JOIN "Order"
ON "customer_id" = "Customer"."id";
```

```
 id | first_name | ... | product_price
  1 | Vincent    | ... |            10
```

The SQL statements on the left create the tables of the schema.

SQL: **Star schema**
In data warehousing, a schema you'll see often is the star schema. A lot of analytical databases like Redshift have optimizations for these kinds of schemas.

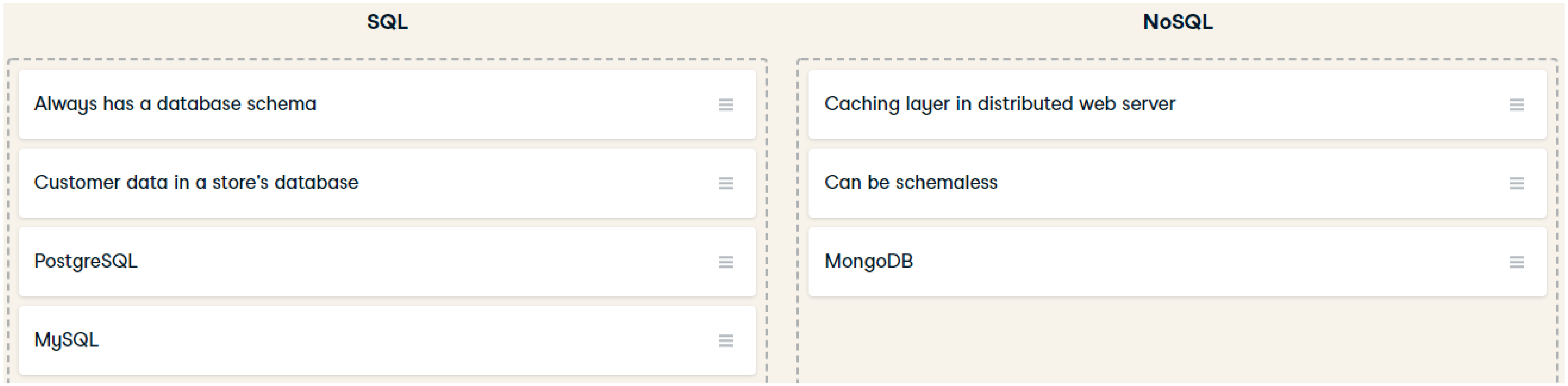


Wikipedia: https://en.wikipedia.org/wiki/Star_schema
According to Wikipedia, "the star schema consists of one or more fact tables referencing any number of dimension tables."
Fact tables contain records that represent things that happened in the world, like orders.
Dimension tables hold information on the world itself, like customer names or product prices.

SQL vs NoSQL

In the video on Databases, you saw the difference between SQL and NoSQL. You saw that SQL uses database schemas to define relations and that NoSQL allows you to work with less structured data.

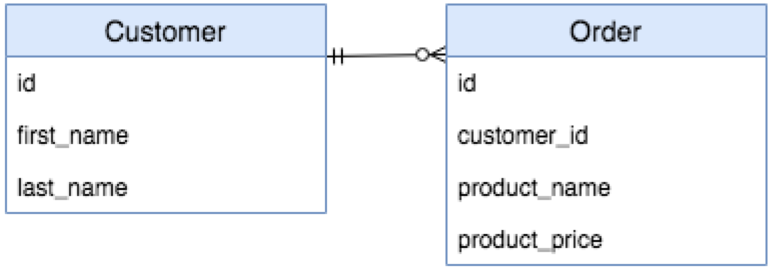
You might also remember some open-source examples of SQL and NoSQL databases and their use cases.



SQL	NoSQL
Always has a database schema	Caching layer in distributed web server
Customer data in a store's database	Can be schemaless
PostgreSQL	MongoDB
MySQL	

The database schema

By now, you know that SQL databases always have a database schema. In the video on databases, you saw the following diagram:



A PostgreSQL database is set up in your local environment, which contains this database schema. It's been filled with some example data. You can use `pandas` to query the database using the `read_sql()` function. You'll have to pass it a database engine, which has been defined for you and is called `db_engine`.

The `pandas` package imported as `pd` will store the query result into a DataFrame object, so you can use any DataFrame functionality on it after fetching the results from the database.

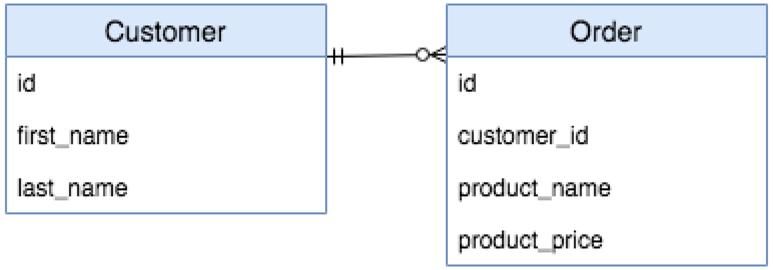
```
# Complete the SELECT statement
data = pd.read_sql("""
SELECT first_name, last_name FROM "Customer"
ORDER BY last_name, first_name
""", db_engine)

# Show the first 3 rows of the DataFrame
print(data.head(3))

# Show the info of the DataFrame
print(data.info())
```

```
<script.py> output:
      first_name last_name
    0    Connagh    Bailey
    1      Brook     Bloom
    2        Ann    Dalton
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 7 entries, 0 to 6
    Data columns (total 2 columns):
    first_name    7 non-null object
    last_name     7 non-null object
    dtypes: object(2)
    memory usage: 192.0+ bytes
    None
```

You now know how to query a SQL database from Python using pandas.

Joining on relations

You've used the following diagram in the previous exercise:



You've learned that you can use the `read_sql()` function from `pandas` to query the database. The real power of SQL is the ability to join information from multiple tables quickly. You do this by using the `JOIN` statement.

When joining two or more tables, `pandas` puts all the columns of the query result into a DataFrame.

```
# Complete the SELECT statement
data = pd.read_sql("""
SELECT * FROM "Customer"
INNER JOIN "Order"
ON "Order"."customer_id"="Customer"."id"
""", db_engine)

# Show the id column of data
print(data.id)
```

```
<script.py> output:
       id  id
    0   1   1
    1   2   2
    2   1   3
    3   5   4
    4   3   5
```

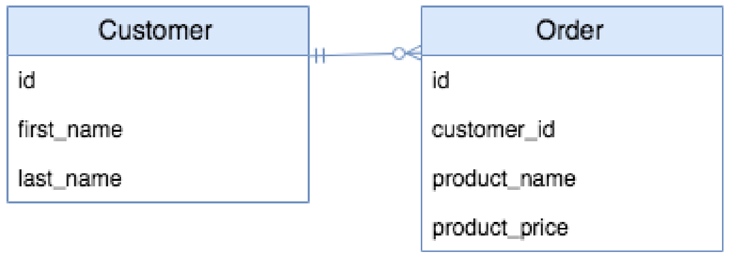
In the IPython Shell, you can see that data.id outputs 2 columns. This is often a source of error, so a better strategy here would be to select specific columns or use the AS keyword in SQL to rename columns.

Star schema diagram

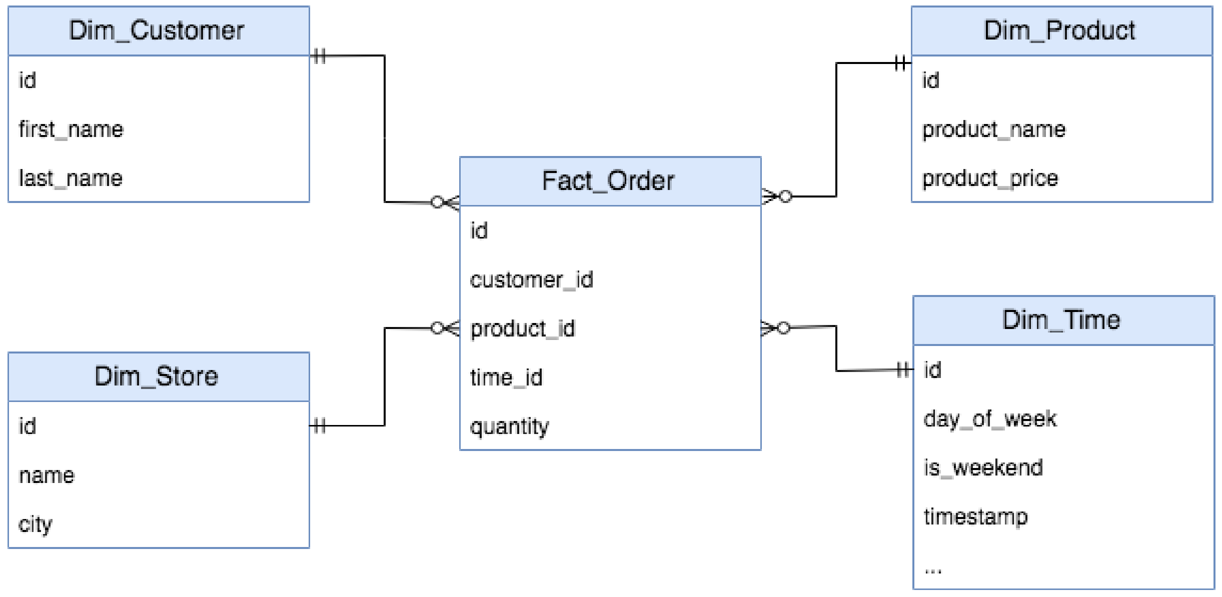
Which of the following images is a star schema?

Possible Answers

- A
- [✓ B]
- None



(A.)



(B.)

What is parallel computing

Chapter 3. Extract, Transform and Load (ETL)

Having been exposed to the toolbox of data engineers, it's now time to jump into the bread and butter of a data engineer's workflow! With ETL, you will learn how to extract raw data from various sources, transform this raw data into actionable insights, and load it into relevant databases ready for consumption!

Chapter 4. Case Study: Data Engineering at DataCamp

Cap off all that you've learned in the previous three chapters by completing a real-world data engineering use case from DataCamp! You will perform and schedule an ETL process that transforms raw course rating data, into actionable course recommendations for DataCamp students!

Course completed!

Recap topics covered:

- Basics

Next Steps

- Basics

Happy learning!