SIC Batch 5

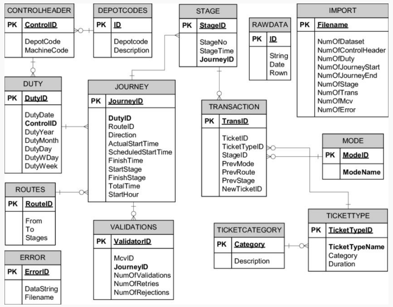
Week 4 - Data Gathering & Databases

What is Database(s)?

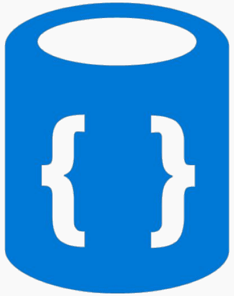
“A database is an organized collection of structured information, or data, typically stored electronically in a computer system” - Oracle

Source: dataversity.net

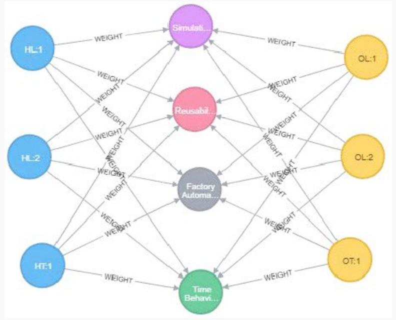
Types of Databases



Relational



Non-Relational (NoSQL)



Graph

- Defines relationship in form of tables
- Data accessed using SQL

- Stores semi-structured and unstructured data
- Horizontally scalable (distributed)

- Defines relationship on form of nodes
- Used for highly connected data relationships

Types of Databases

Database software is called a Database Management System (DBMS)

PostgreSQL

MySQL

Google BigQuery

Relational

mongoDB

cassandra

elastic

APACHE HBASE

Non-Relational (NoSQL)

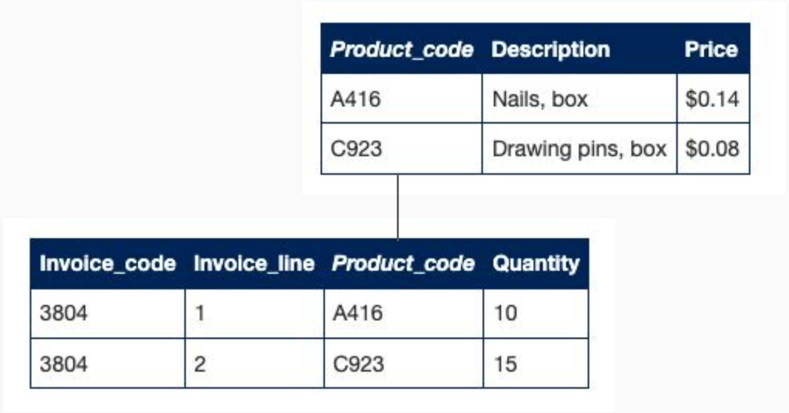
neo4j

OrientDB

Graph

Relational Database

- Table consists of rows and columns
- Tables might have columns in common that have relationship
- Each column in a table have a schema with data type

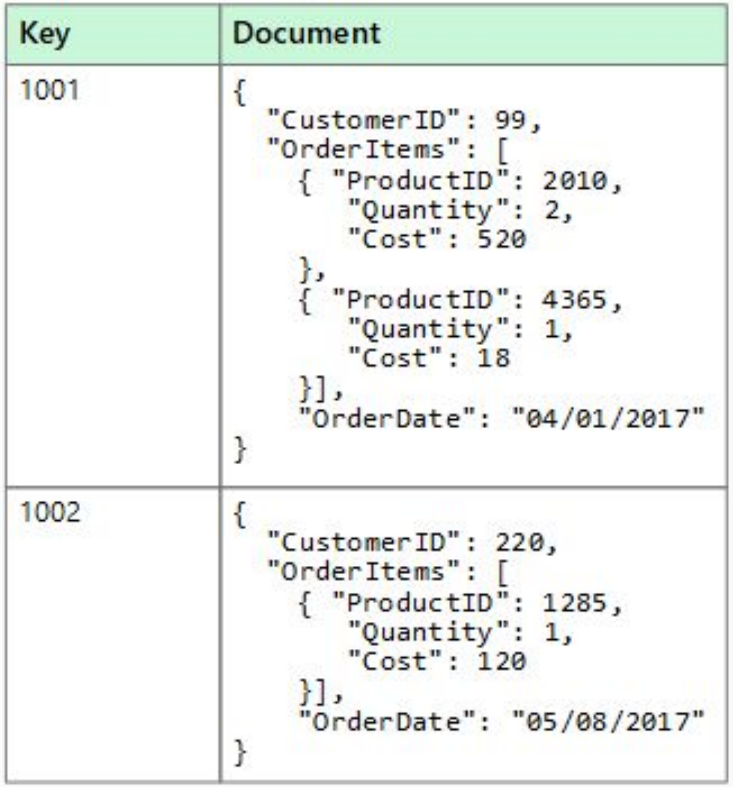


Product_code	Description	Price
A416	Nails, box	$0.14
C923	Drawing pins, box	$0.08

Invoice_code	Invoice_line	*Product_code*	Quantity
3804	1	A416	10
3804	2	C923	15

Non-Relational Database

- Document-oriented
- JSON like



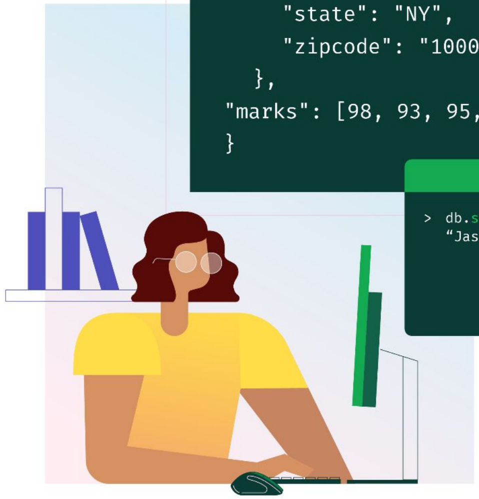
Key	Document
1001	`{ "CustomerID": 99, "OrderItems": [ { "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }`
1002	`{ "CustomerID": 220, "OrderItems": [ { "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }`

MongoDB

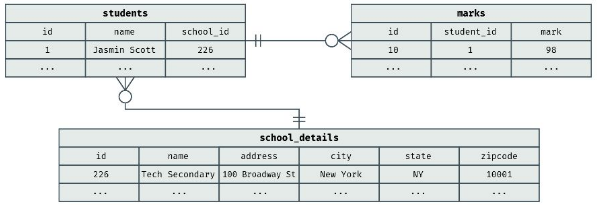
```
{
"_id": 1,
"student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
"marks": [98, 93, 95, 88, 100],
}
```

mongo

```
> db.students.find({"student_name":
  "Jasmin Scott"})
```



SQL



students

id	name	school_id
1	Jasmin Scott	226
...	...	...

marks

id	student_id	mark
10	1	98
...	...	...

school_details

id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...	...	...	...	...	...

Results

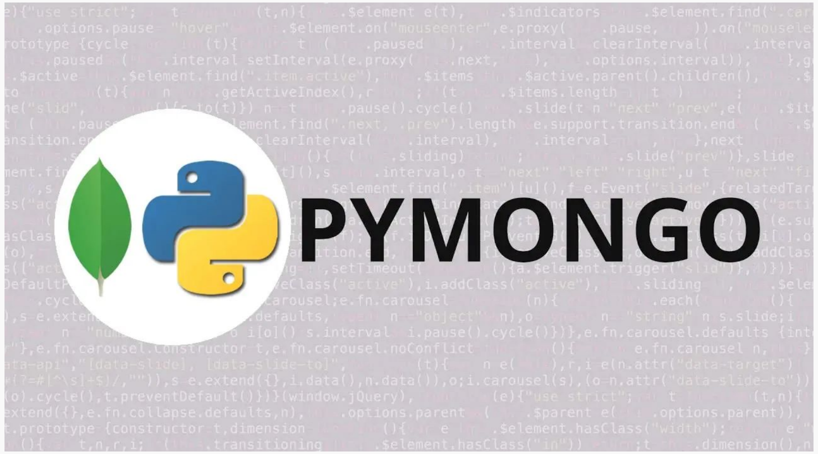
name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...	...	...	...

sql

```
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

Intro PyMongo

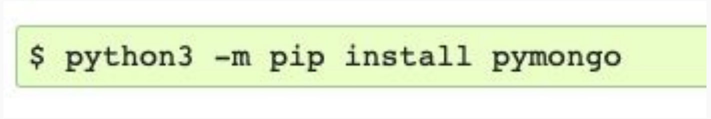
PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python



Intro PyMongo

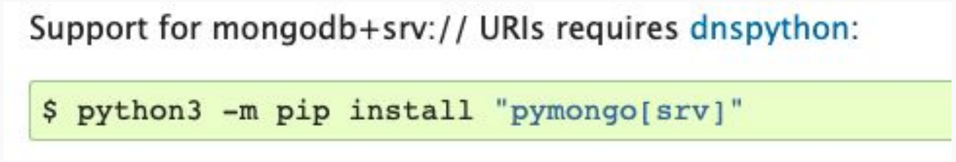
- **Installing PyMongo**

Standard Installation:



```
$ python3 -m pip install pymongo
```

With MongoDB Atlas Installation:



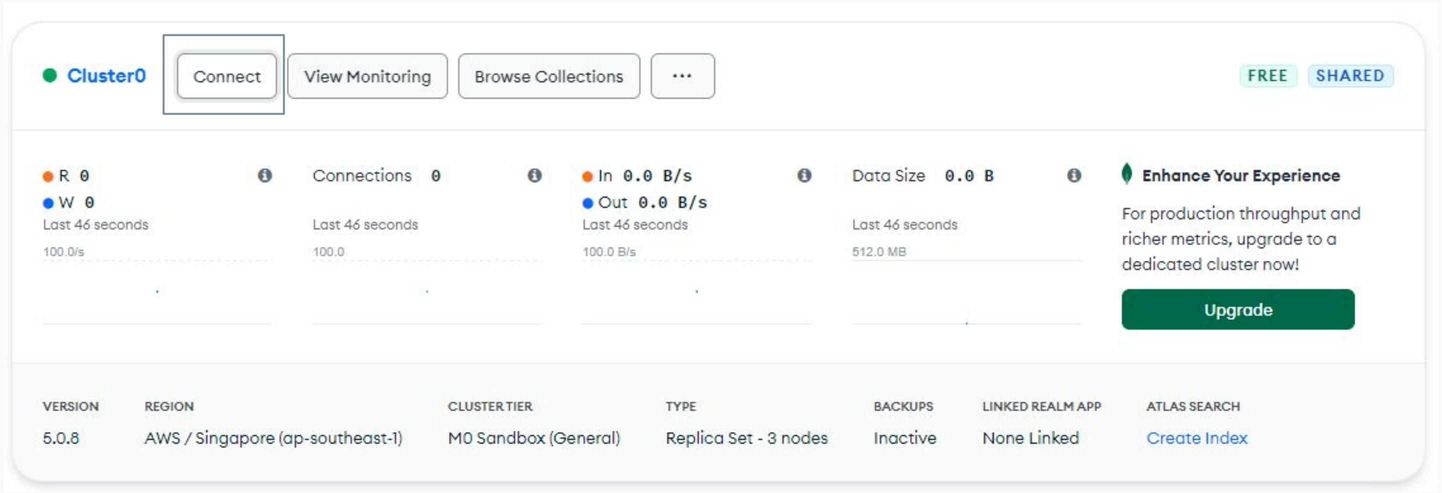
Support for mongodb+srv:// URIs requires dnspython:

```
$ python3 -m pip install "pymongo[srv]"
```

https://pymongo.readthedocs.io/en/stable/installation.html

Connect with PyMongo

- Connect with Atlas Cluster



R 0
W 0
Last 46 seconds
100.0/s

Connections 0
Last 46 seconds
100.0

In 0.0 B/s
Out 0.0 B/s
Last 46 seconds
100.0 B/s

Data Size 0.0 B
Last 46 seconds
512.0 MB

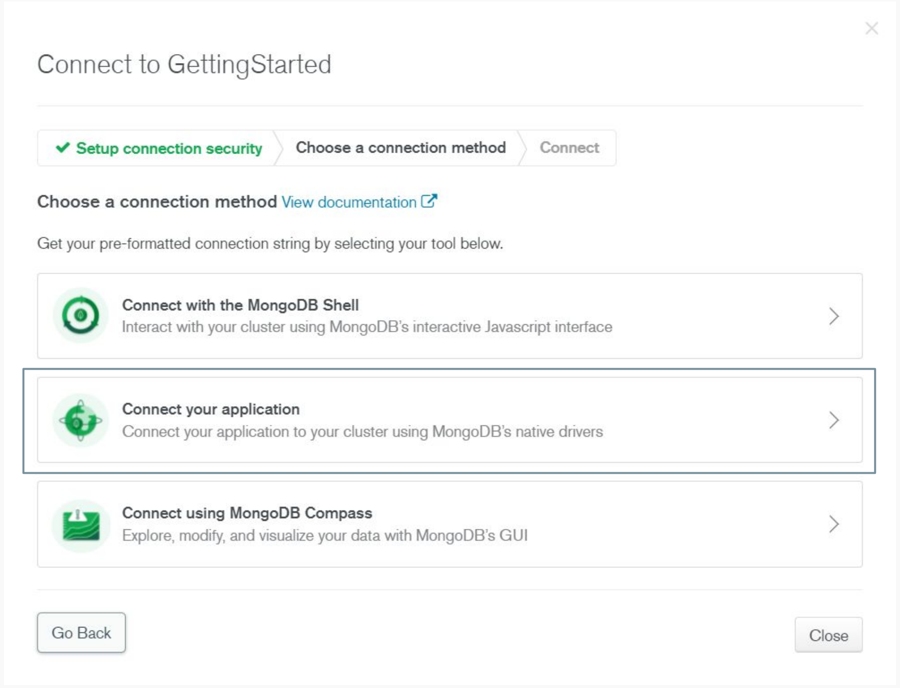
Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.8	AWS / Singapore (ap-southeast-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

Connect with PyMongo

- **Connect with Atlas Cluster**



Connect to GettingStarted

✓ Setup connection security | Choose a connection method | Connect

Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

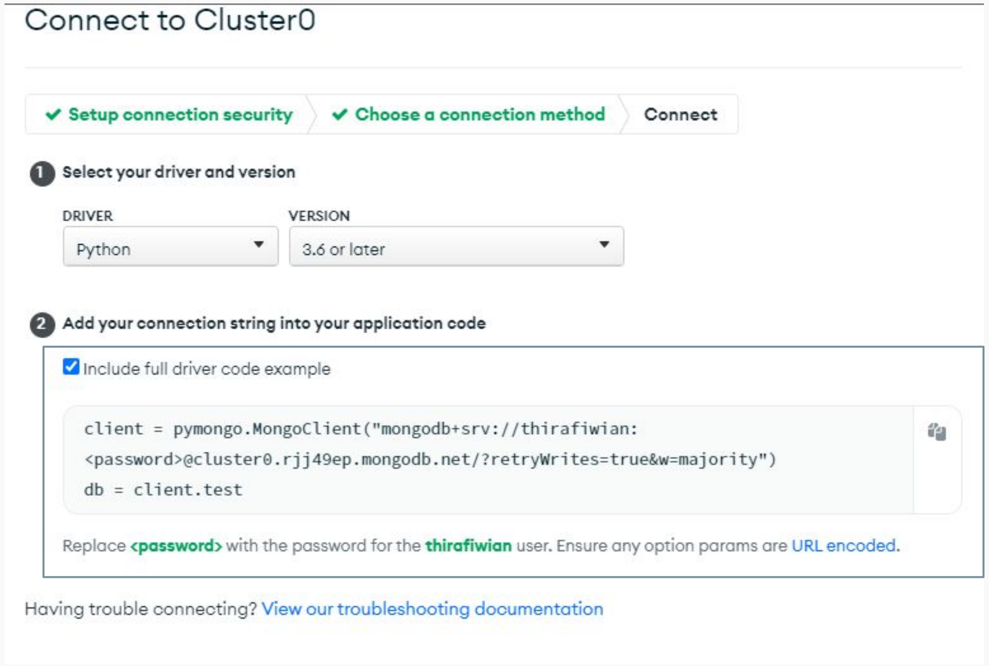
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Connect with PyMongo

- Connect with Atlas Cluster


Connect to Cluster0

1 Select your driver and version

DRIVER: Python
VERSION: 3.6 or later

2 Add your connection string into your application code

[✓] Include full driver code example

```
client = pymongo.MongoClient("mongodb+srv://thirafiwian:
<password>@cluster0.rjj49ep.mongodb.net/?retryWrites=true&w=majority")
db = client.test
```

Replace **<password>** with the password for the **thirafiwian** user. Ensure any option params are URL encoded.

Having trouble connecting? View our troubleshooting documentation

Connect with PyMongo

- **Connect with Atlas Cluster**

```
import pymongo # meng-import library pymongo yang sudah kita install
client =
pymongo.MongoClient("mongodb+srv://jphartogi:BhdvAX9DPH9kjsTx@gettingstarted
.zo2se.mongodb.net/GettingStarted?retryWrites=true&w=majority")
db = client.test
print(db)
```

Query with PyMongo

- **Insert Data**
  - Jika kita ingin membuat sebuah *document* dalam MongoDB, maka kita harus membuat sebuah *dictionary* dimana *keys* adalah *column headers* dan *values* adalah *attribute* dari data yang kita ingin simpan dalam *database*.
  - Kita dapat menggunakan *function* `collection.insert_many()` untuk menyimpan beberapa dokumen sekaligus, atau `collection.insert_one()` untuk menyimpan satu data saja. Disini kita akan menggunakan *database* contoh yang telah kita buat sebelumnya.

Query with PyMongo

- **Insert Data**

```
import pymongo # meng-import library pymongo yang sudah kita install
client = pymongo.MongoClient("MASUKAN ID KALIAN")
db = client['MyDatabase'] # ganti sesuai dengan nama database kalian
my_collections = db['MyCollection'] # ganti sesuai dengan nama collections kalian

# Data yang ingin dimasukkan
murid_1 = {'nama':'John Doe','Jurusan':'IPS','Nilai':90}
murid_2 = {'nama':'Jane Doe', 'Jurusan':'IPA','Nilai':85}

results = my_collections.insert_many([murid_1,murid_2])
print(results.inserted_ids) # akan menghasilkan ID dari data yang kita masukkan
```

Query with PyMongo

- **Read Data**
  - Setelah data sudah masuk ke dalam *database*, maka kita juga bisa membaca data tersebut menggunakan PyMongo. Untuk membaca seluruh data kita dapat menggunakan *function* `collections.find()` untuk membaca seluruh data dalam *collections*.

Query with PyMongo

- **Read Data**

```
import pymongo # meng-import library pymongo yang sudah kita install
client = pymongo.MongoClient("MASUKAN ID KALIAN")
db = client['MyDatabase'] # ganti sesuai dengan nama database kalian
my_collections = db['MyCollection'] # ganti sesuai dengan nama
collections kalian

for x in my_collections.find():
  print(x)
```

Challenges

Challenge!

Buatlah sebuah aplikasi Flask yang terkoneksi kedalam MongoDB dan terdiri dari beberapa kondisi berikut

1. POST API dengan
   a. route /sensor1
   b. 2 buah data (buat dummy i.e temperature, kelembapan) dan timestamp
   c. Simpan data tersebut pada database

Challenge!

Buatlah sebuah aplikasi Flask yang terkoneksi kedalam MongoDB dan terdiri dari beberapa kondisi berikut

1. GET API dengan
    a. route
        i. /sensor1/#NAMA_SENSOR_1/all
        ii. /sensor1/#NAMA_SENSOR_1/avg
        iii. /sensor1/#NAMA_SENSOR_2/all
        iv. /sensor1/#NAMA_SENSOR_2/avg

API dengan /all dan /avg akan berfungsi untuk

1. Mengambil seluruh data temperature/kelembapan dari database (/all)
2. Return nilai rata-rata dari seluruh data tersebut (/avg)

Challenge Opsional!

Buatlah sebuah aplikasi Flask yang terkoneksi kedalam MongoDB dan terdiri dari beberapa kondisi berikut

1. GET API dengan
   a. route
      i. /sensor1/#NAMA_SENSOR/all?sort=lowest/highest
      ii. /sensor1/#NAMA_SENSOR/avg?start=01-02-2024&end=02-02-2024

API dengan ?sort dan ?start-end akan berfungsi untuk

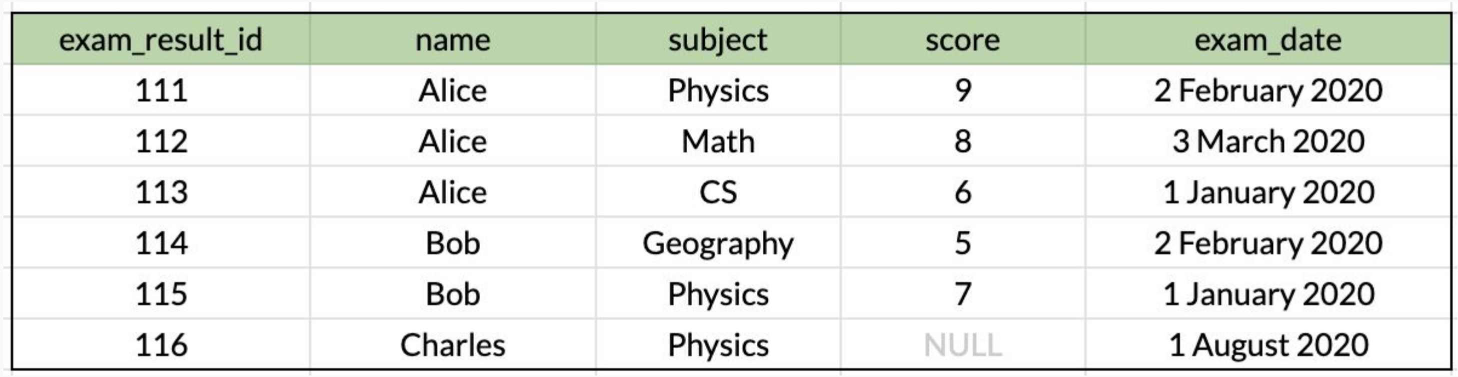
1. Sortir data dari yang terendah ke yang tertinggi atau sebaliknya
2. Sortir data dengan timestamp tertentu

Q&A

Query with SQL

Let’s take a closer look at a table

student_grade

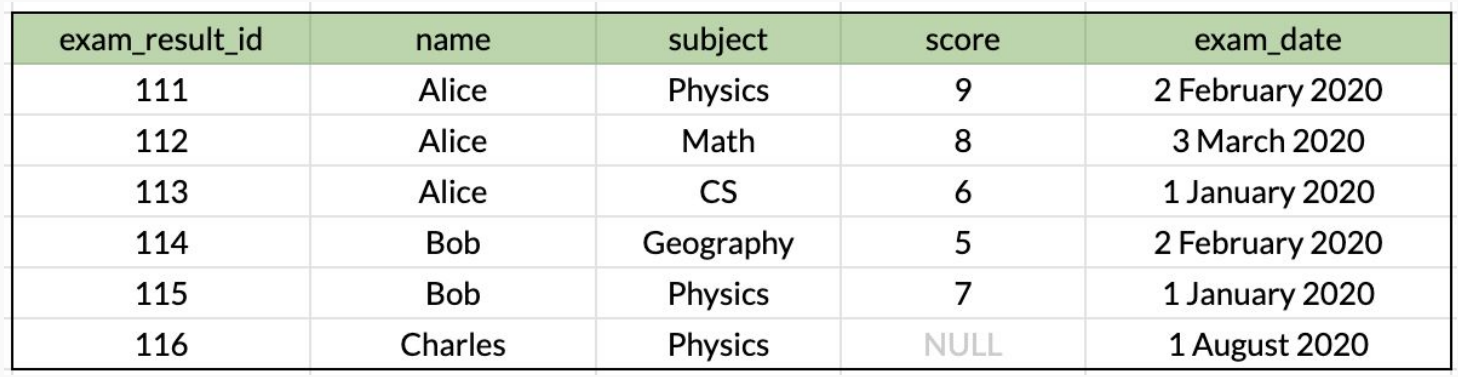


exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

A table/relation is a collection of entities of the same type.

Let’s take a closer look at a table

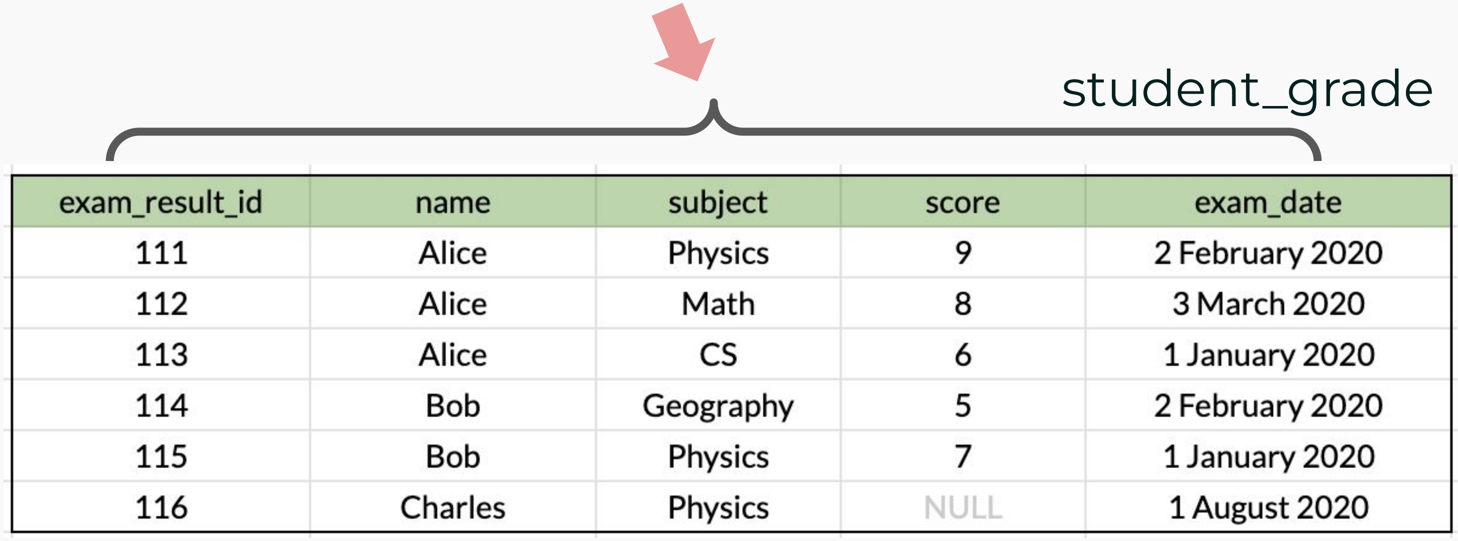
student_grade



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

Each row/record represents a distinct entity.

Let’s take a closer look at a table

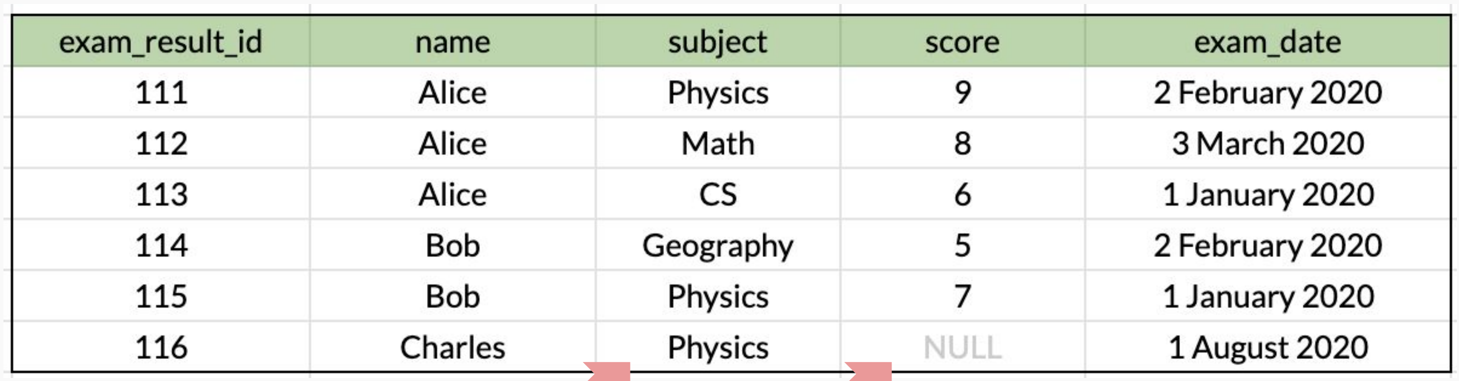

student_grade

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

Each column/field/attribute represents an attribute of the entities.

Let’s take a closer look at a table

student_grade



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

Each cell holds one specific piece of information.
SQL lets you query this table!

SQL lets you query this table

Source table

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

SELECT statement/ SQL query

```
SELECT score
FROM student_grades
```

```
SELECT ______
FROM ________
```

Query result

score
9
8
6
5
7
NULL

SELECT clause to choose desired columns

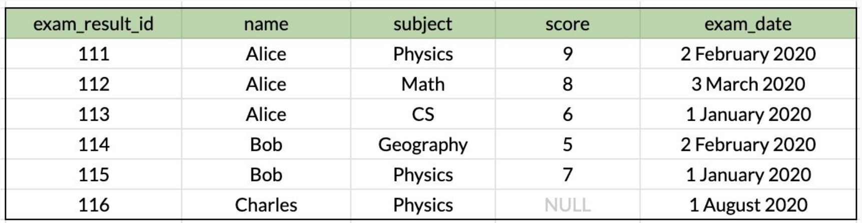
SELECT can rename and reorder columns

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT score AS grade,
       name
FROM student_grades
```

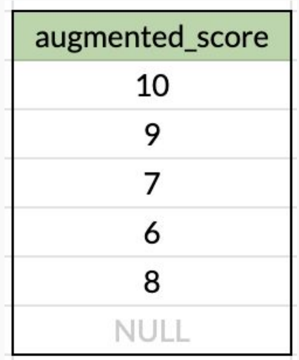
grade	name
9	Alice
8	Alice
6	Alice
5	Bob
7	Bob
NULL	Charles

SELECT can define new columns



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT score+1 AS augmented_score
FROM student_grades
```



augmented_score
10
9
7
6
8
NULL

SELECT * to choose all columns

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
```

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

WHERE clause to choose rows

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020

SELECT *
FROM student_grades
WHERE $\underbrace{\text{name = "Alice"}}_{\textbf{Boolean expression}}$

Aliases in the SELECT clause cannot be used in WHERE clause!

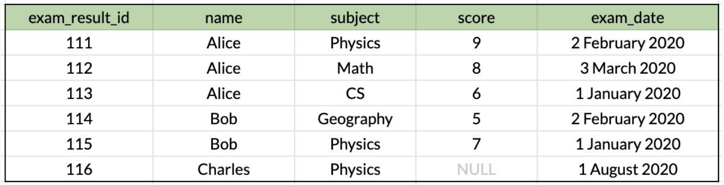
WHERE … IN clause to select from a list of value

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
WHERE name IN ("Alice", "Bob")
```

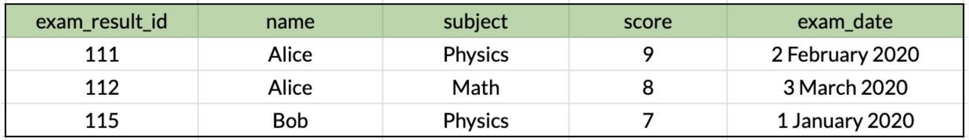
exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020

WHERE clause can accept >1 Boolean expression



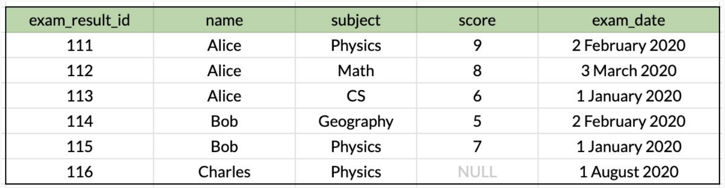
exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
WHERE
        (name IN ("Alice", "Bob"))
        AND
        (score >= 7)
```



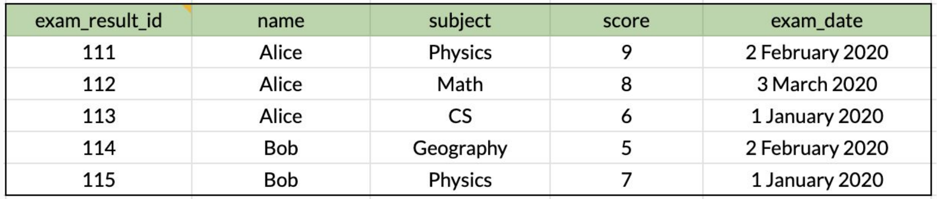
exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
115	Bob	Physics	7	1 January 2020

ISNULL and IS NOT NULL to filter null values



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
WHERE score IS NOT NULL
```



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020

TIPS: comment and format your queries!

```
-- Select Alice and Bob’s good exam results
SELECT *
FROM student_grades
WHERE
    (name IN ("Alice", "Bob"))
    AND
    (score >= 7)
```

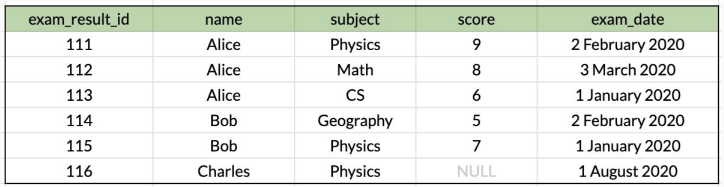
LIMIT to limit the number of rows returned

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
LIMIT 4
```

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020

DISTINCT when you don’t want duplicate result



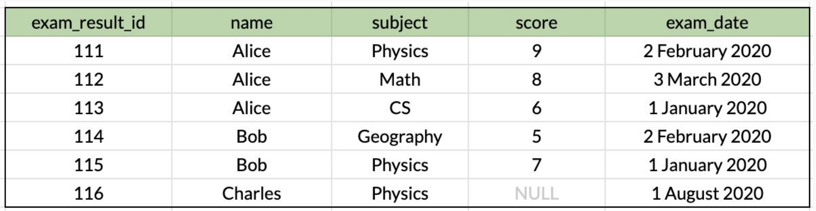
exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT DISTINCT name
FROM student_grades
```

```
Name
Alice
Bob
Charles
```

Quick Checkpoint:
How many rows returned if we select distinct name and subject?

GROUP BY to group rows based on shared values



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT name, AVG(score) as
average_score
FROM student_grades
GROUP BY name
```

name	average_score
Alice	7.666666667
Bob	6
Charles	null

Aggregation Functions

AVG()

COUNT()

SUM()

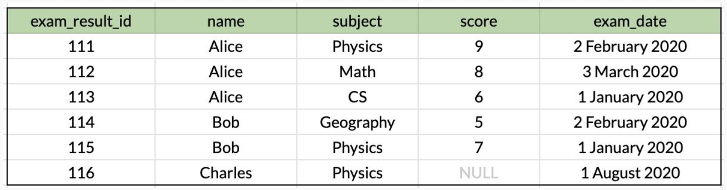
SQL Aggregate Functions

Grouping multi rows

MAX()

MIN()

Sample Aggregate (MIN, MAX, COUNT)

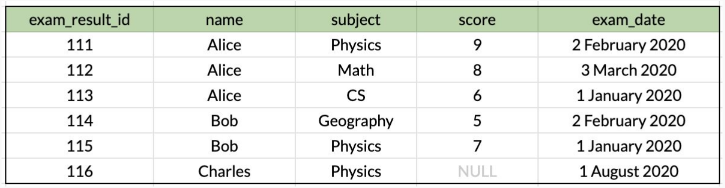


exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT MIN(exam_result_id) AS min_id,
          MAX(exam_result_id) AS max_id,
          COUNT(score) AS count_score
FROM student_grades
```

Quick Checkpoint:
What will that query return?

Sample Aggregate (MIN, MAX, AVG)



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT name,
        MIN(exam_result_id) AS min_id,
        MAX(exam_result_id) AS max_id,
        AVG(score) AS avg_score
FROM student_grades
GROUP BY name
```

Quick Checkpoint:
What will that query return?

ORDER BY to sort the query results by column

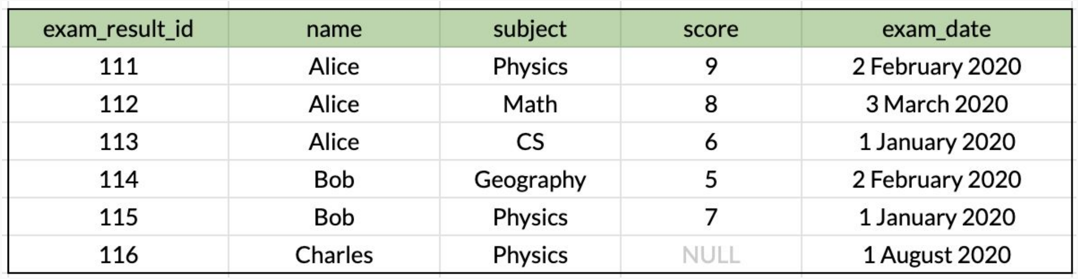
exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

```
SELECT *
FROM student_grades
ORDER BY score DESC
```

exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
115	Bob	Physics	7	1 January 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
116	Charles	Physics	NULL	1 August 2020

Challenges

Given the table below, write a query to get the names and scores of students who took either physics or math exams and whose score is available (not null). Sort the rows from highest to lowest.



exam_result_id	name	subject	score	exam_date
111	Alice	Physics	9	2 February 2020
112	Alice	Math	8	3 March 2020
113	Alice	CS	6	1 January 2020
114	Bob	Geography	5	2 February 2020
115	Bob	Physics	7	1 January 2020
116	Charles	Physics	NULL	1 August 2020

Q&A