SQL

Big Data Systems

Dr. Rubi Boim

Motivation (for this course)

SQL is an important "standard"

- Used in RDBMS and most Data Warehouses

- **But NOT in most NoSQL**

  - Each product has its own API

  - BUT some are built on the same building blocks
    CQL (Cassandra)

- Joins and normalization are crucial for RDBMS
  You should know them well as they are <u>anti-patterns</u> for Wide columns

Reminder - relational model

- Data is stored in tables of columns and rows

- A <u>unique key</u> identify each row

- The table is <u>unordered</u> (no first / last)

Table / relation

Columns / attributes

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	\<null\>
102	Tova Milo	Tel Aviv	\<null\>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

Rows / tuples

Structured Query Language

- An "API" for querying and maintaining the database

- Different standards (ANSI SQL, SQL3…)

Can be classified to

- Data Definition Language (DDL)
  create / alter / delete <u>tables</u>

- Data Manipulation Language (DML)
  select / insert / update /delete <u>data</u>

Data Definition Language (DDL)

create / alter / delete tables

(We present here only the basics - there are a lot more options for each operation)

CREATE TABLE

Creates a new table

```
CREATE TABLE  table(
     column1 DATATYPE,
     column2 DATATYPE,
     column3 DATATYPE,
     ...)
```

DATATYPES
VARCHAR(n),
INT, SMALLINT, MEDIUMINT, BIGINT,
FLOAT, DOUBLE,
DATE, DATETIME, TIMESTAMP,
BIT

…

CREATE TABLE

```
CREATE TABLE users(
      user_id INT,
      name VARCHAR(255),
      city VARCHAR(255),
      birthdate DATE
)
```

users

user_id	name	city	brithdate

CREATE TABLE

```
CREATE TABLE users(
     user_id INT NOT NULL,
     name VARCHAR(255),
     city VARCHAR(255),
     birthdate DATE,
     PRIMARY KEY(user_id)
)
```

MySQL syntax

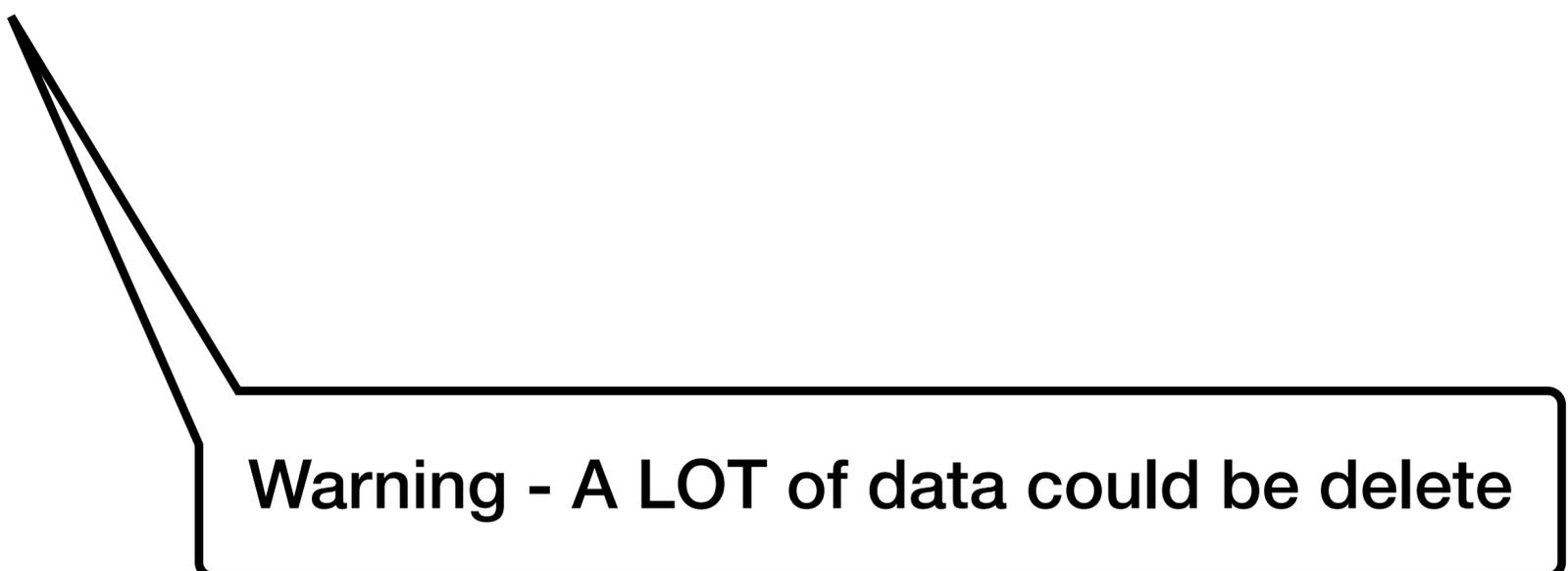Oracle / SQL Server use inline
(user_id INT NOT NULL PRIMARY KEY)

users

user_id	name	city	brithdate

DROP TABLE

Deletes an existing table

```
DROP TABLE table
```

Warning - A LOT of data could be delete

ALTER TABLE

Alters an existing table

```
ALTER TABLE table(
    ADD column1 DATATYPE,
    DROP column2,
    ALTER column3 newName DATATYPE

)
```

ALTER TABLE

```
CREATE TABLE users(
     user_id INT,
     name VARCHAR(255),
     city VARCHAR(255),
     birthdate DATE
```

users

user_id	name	city	brithdate

```
ALTER TABLE users(
     DROP city)
```

users (after alter)

user_id	name	brithdate

Data Manipulation Language (DML)

select / insert / update /delete data

(We present here only the basics - there are a lot more options for each operation)

SELECT

Retrieves data from the database

```
SELECT     attributes
FROM       tables
WHERE      conditions
ORDER BY   attributes
```

SELECT

**Return all users in
a descending order**

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

SELECT

Return all users in a descending order

```
SELECT *
FROM users
ORDER BY name DESC
```

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	\<null\>
102	Tova Milo	Tel Aviv	\<null\>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963
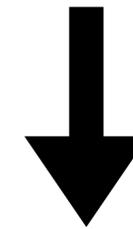
user_id	name	city	brithdate
102	Tova Milo	Tel Aviv	\<null\>
101	Rubi Boim	Tel Aviv	\<null\>
104	Michael Jordan	Chicago	17/02/1963
103	Lebron James	Los Angeles	30/12/1984

SELECT

Return the ids and names of all Tel Aviv residences

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

16

SELECT

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

Return the ids and names of all Tel Aviv residences

```
SELECT user_id, name
FROM users
WHERE city = "Tel Aviv"
ORDER BY name
```

user_id	name
101	Rubi Boim
102	Tova Milo

SELECT

Return the ids, names and birthdates of all who were born post 1980

users

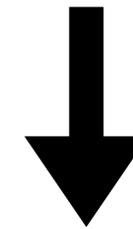| user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

SELECT

Return the ids, names and birthdates of all who were born post 1980

```
SELECT user_id, name, birthdate
FROM users
WHERE birthdate >= '01/01/1980'
ORDER BY name
```

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	\<null\>
102	Tova Milo	Tel Aviv	\<null\>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

user_id	name	brithdate
103	Lebron James	30/12/1984

SELECT

Select all cities

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

SELECT

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

Select all cities

```
SELECT city
FROM users
ORDER BY city
```

city
Chicago
Los Angeles
Tel Aviv
Tel Aviv

21

SELECT

Select all cities

```
SELECT DISTINCT city
FROM users
ORDER BY city
```

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963
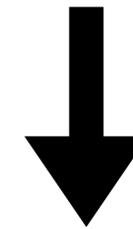
city
Chicago
Los Angeles
Tel Aviv

SELECT with Joins

- What is the connection between the tables?

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

cities

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

SELECT with Joins

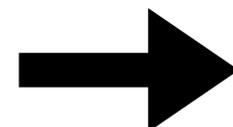- Select all users who lives in "small" cities (<1m)

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael	Chicago	17/02/1963

cities

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

SELECT with Joins

- Select all users who lives in "small" cities (<1m)

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael	Chicago	17/02/1963

cities

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

SELECT with Joins

- Select all users who lives in "small" cities (<1m)

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael	Chicago	17/02/1963

cities

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT users.*
FROM   users, cities
WHERE  users.city = cities.name AND
       cities.population < 1000000
```

user_id	name	city	brithdat
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>

SELECT with Joins

- Find all users who lives in "small" cities (<1m) in Europe

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
```

SELECT with Joins

- Find all users who lives in "small" cities (<1m) in Europe

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
```

⬇
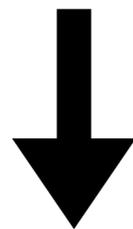
```
SELECT  users.*
FROM    users, cities, countries
WHERE   users.city = cities.name AND
        cities.population < 1000000 AND
        cities.country = countries.name AND
        countries.region = "Europe"
```

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

```
SELECT  f1.user_id
FROM    friends AS f1, friends AS f2
WHERE   f1.friend_user_id = f2.user_id AND
        f2.friend_user_id = 103
```
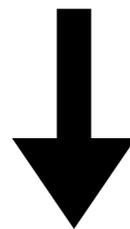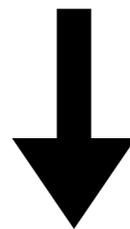
SELECT with Joins

• find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

⬇

```
SELECT f1.user_id
FROM   friends AS f1, friends AS f2
WHERE  f1.friend_user_id = f2.user_id AND
       f2.friend_user_id = 103
```

Is this ok?

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

⬇

```
SELECT DISTINCT f1.user_id
FROM    friends AS f1, friends AS f2
WHERE   f1.friend_user_id = f2.user_id AND
        f2.friend_user_id = 103
```

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

⬇

```
SELECT DISTINCT f1.user_id
FROM    friends AS f1, friends AS f2
WHERE   f1.friend_user_id = f2.user_id AND
        f2.friend_user_id = 103
```

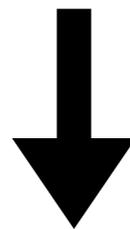What about first degree friends?

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

↓

```
SELECT DISTINCT f1.user_id
FROM   friends AS f1, friends AS f2
WHERE  f1.friend_user_id = f2.user_id AND
       f2.friend_user_id = 103 AND
       f1.friend_user_id <> 103
```
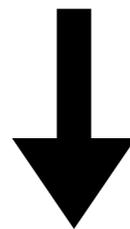
SELECT with Joins

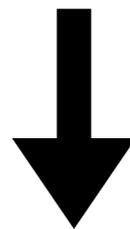- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

Can you do it with subqueries instead of joins?

```
SELECT DISTINCT f1.user_id
FROM    friends AS f1, friends AS f2
WHERE   f1.friend_user_id = f2.user_id AND
        f2.friend_user_id = 103 AND
        f1.friend_user_id <> 103
```

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```
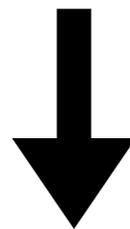
⬇

```
SELECT DISTINCT user_id
FROM    friends
WHERE   friend_user_id IN
        (SELECT user_id
         FROM friends WHERE friend_user_id = 103)
```

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

⬇

```
SELECT DISTINCT user_id
FROM    friends
WHERE   friend_user_id IN
        (SELECT user_id
         FROM friends WHERE friend_user_id = 103)
```

What about first degree friends?

SELECT with Joins

- find all 2nd degree friends of Lebron (103)

```
users(user_id, name, city, birthdate)
cities(name, country, population)
countries(name, region, population)
friends(user_id, friend_user_id, since_date)
```

```
SELECT DISTINCT user_id
FROM    friends
WHERE   friend_user_id IN
        (SELECT user_id FROM friends
         WHERE friend_user_id = 103)
AND     user_id NOT IN
        (SELECT user_id FROM friends
         WHERE friend_user_id = 103)
```
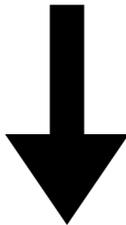
SELECT logic - Cartesian product

items

item_id	title	company_id
2003	iPad	1
2004	iPhone	1
2005	55' LED TV	2

companies

id	title
1	Apple
2	Samsung

```
SELECT * FROM items, companies
```

SELECT logic - Cartesian product

items

item_id	title	company_id
2003	iPad	1
2004	iPhone	1
2005	55' LED TV	2

companies

id	title
1	Apple
2	Samsung

`SELECT * FROM` `items, companies`

item_id	title	company_id	id	title
2003	iPad	1	1	Apple
2003	iPad	1	2	Samsung
2004	iPhone	1	1	Apple
2004	iPhone	1	2	Samsung
2005	55' LED TV	2	1	Apple
2005	55' LED TV	2	2	Samsung

SELECT logic - Cartesian product

items

item_id	title	company_id
2003	iPad	1
2004	iPhone	1
2005	55' LED TV	2

companies

id	title
1	Apple
2	Samsung

```
SELECT * FROM items, companies
WHERE company_id = id
```

item_id	title	company_id	id	title
2003	iPad	1	1	Apple
2003	iPad	1	2	Samsung
2004	iPhone	1	1	Apple
2004	iPhone	1	2	Samsung
2005	55' LED TV	2	1	Apple
2005	55' LED TV	2	2	Samsung

SELECT logic - Cartesian product

items

item_id	title	company_id
2003	iPad	1
2004	iPhone	1
2005	55' LED TV	2

companies

id	title
1	Apple
2	Samsung

```
SELECT * FROM items, companies
WHERE company_id = id
```

item_id	title	company_id	id	title
2003	iPad	1	1	Apple
2004	iPhone	1	1	Apple
2005	55' LED TV	2	2	Samsung

INSERT / UPDATE / DELETE

(We present here only the basics - there are a lot more options for each operation)

INSERT

Insert data to the database

```
INSERT INTO table(A1,…,An)
VALUES        (V1,…,Vn)
```

- Without attributes all values are required in order

- Missing attributes will be added as NULL

INSERT

```
INSERT INTO users(user_id, name, city)
VALUES (101, 'Rubi Boim', 'Tel Aviv')


INSERT INTO users
VALUES (103, 'Lebron James', 'Los Angeles', '30/12/1984')
```

users

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984

DELETE

Deletes data from the database

```
DELETE FROM table
WHERE       conditions
```

Warnings

- double check the **conditions**

- If no conditions are set, **ALL DATA will be deleted**

DELETE

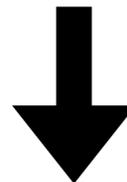| user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

```
DELETE FROM users WHERE user_id = 104
DELETE FROM users WHERE city = 'Tel Aviv'
```

DELETE

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

```
DELETE FROM users WHERE user_id = 104
DELETE FROM users WHERE city = 'Tel Aviv'
```
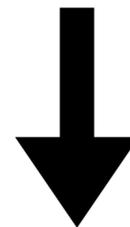
user_id	name	city	brithdate
103	Lebron James	Los Angeles	30/12/1984

UPDATE

Update data in the database

```
UPDATE  table
SET     attr1 = <value>,
        attr1 = <value>
WHERE   conditions
```

Warnings

- double check the **conditions**

- If no conditions are set, **ALL DATA will be updated**

UPDATE

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

```
UPDATE users SET city = 'Tel-Aviv'
WHERE city = 'Tel Aviv'
```

UPDATE

user_id	name	city	brithdate
101	Rubi Boim	Tel Aviv	<null>
102	Tova Milo	Tel Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

```
UPDATE users SET city = 'Tel-Aviv'
WHERE city = 'Tel Aviv'
```

user_id	name	city	brithdate
101	Rubi Boim	Tel-Aviv	<null>
102	Tova Milo	Tel-Aviv	<null>
103	Lebron James	Los Angeles	30/12/1984
104	Michael Jordan	Chicago	17/02/1963

Aggregation / Grouping / Union / Subqueries

Aggregation

Aggregates the rows and calculate a function

```
SELECT AVG(attr)
FROM    table
WHERE   conditions
```

Popular operations

- COUNT, AVG, SUM, MIN, MAX, AVG

Aggregation

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

What is the average population of all cities?

Aggregation

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

What is the average population of all cities?

```
SELECT avg(population)
FROM cities
```
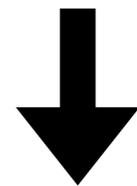
avg(population)
2,387,500

Aggregation

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

What is the average population of all cities?

How many "big cities" (>1m) are in the DB?

```
SELECT avg(population)
FROM cities
```

avg(population)
2,387,500

Aggregation

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

What is the average population of all cities?

```
SELECT avg(population)
FROM cities
```

⬇
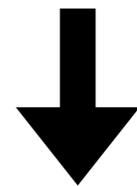
avg(population)
2,387,500

How many "big cities" (>1m) are in the DB?

```
SELECT count(*)
FROM cities
WHERE population > 1000000
```

⬇
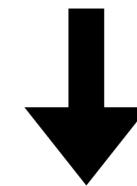
count(*)
3

GROUP BY

Aggregates on specific attributes

```
SELECT     attributes
FROM       table
WHERE      conditions
GROUP BY   attributes
HAVING     aggregates
```

- SELECT contains only aggregates / group by attributes

- GROUP BY is performed <u>after</u> the WHERE

- HAVING contains only aggregates attributes and performed finally

GROUP BY

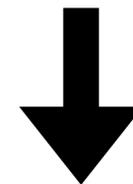**For each country, how many cities are in the DB?**

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

GROUP BY

For each country, how many cities are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
GROUP BY country
```

country	count(*)
Israel	1
France	1
USA	2

GROUP BY

What is the average population of all cities per country?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

GROUP BY

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

What is the average population of all cities per country?

```
SELECT country, avg(population)
FROM cities
GROUP BY country
```
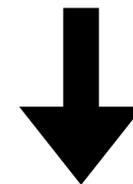
country	avg(population)
Israel	450,000
USA	3,500,000
France	2,100,000

GROUP BY

Which are the countries with exactly 1 city in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

GROUP BY

Which are the countries with exactly 1 city in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
GROUP BY country
HAVING count(*) = 1
```
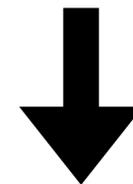
country	count(*)
Israel	1
France	1

GROUP BY

For each country, how many "big cities" (>1m) are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

GROUP BY

For each country, how many "big cities" (>1m) are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
WHERE population > 1000000
GROUP BY country
```
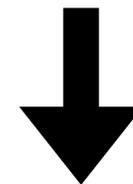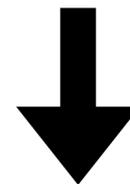
country	count(*)
Israel	0
USA	2
France	1

GROUP BY

For each country,
how many "big cities"
(>1m) are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
WHERE population > 1000000
GROUP BY country
```

country	count(*)
Israel	0
USA	2
France	1

What is the problem here?

GROUP BY

For each country, how many "big cities" (>1m) are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
WHERE population > 1000000
GROUP BY country
```

country	count(*)
USA	2
France	1

GROUP BY

For each country,
how many "big cities"
(>1m) are in the DB?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT country, count(*)
FROM cities
WHERE population > 1000000
GROUP BY country
```

country	count(*)
USA	2
France	1

HW - how to return the query
with <Israel,0>?

UNION, INTERSECTION, DIFFERENCE

```
SELECT name
FROM cities
WHERE country = 'USA'

UNION

SELECT name
FROM cities
WHERE country <> 'USA' AND
      population < 1000000
```

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

➡️ ?

- Attribute names must be the same (use "AS")

UNION, INTERSECTION, DIFFERENCE

```
SELECT name
FROM cities
WHERE country = 'USA'


UNION


SELECT name
FROM cities
WHERE country <> 'USA' AND
      population < 1000000
```

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

➡
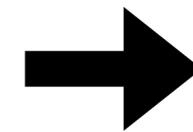
name
Tel Aviv
Chicago
Los Angeles

- Attribute names must be the same (use "AS")

Subqueries

Which cities has a lower population than all the cities in USA?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

Subqueries

Which cities has a lower population than all the cities in USA?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT name
FROM cities
WHERE population < ALL
      (SELECT population
       FROM cities
       WHERE country = 'USA')
```
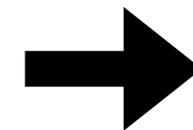
name
Tel Aviv
Paris

Subqueries

Which cities has a lower population than all the cities in USA?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT name
FROM cities
WHERE population < ALL
      (SELECT population
       FROM cities
       WHERE country = 'USA')
```
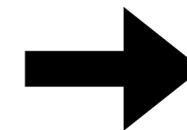
name
Tel Aviv
Paris

Can you think of another version?

Subqueries

Which cities has a lower population than all the cities in USA?

name	country	population
Tel Aviv	Israel	450,000
Chicago	USA	3,000,000
Paris	France	2,100,000
Los Angeles	USA	4,000,000

```
SELECT name
FROM cities
WHERE population <
      (SELECT min(population)
       FROM cities
       WHERE country = 'USA')
```
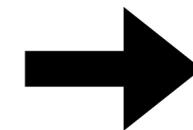
name
Tel Aviv
Paris

Quick questions

Question (1)

Find all action movies

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(user_id, movie_id, view_timestamp)
```

Question (1)

Find all action movies

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(user_id, movie_id, view_timestamp)
```

↓
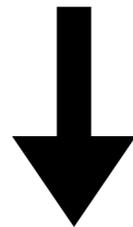
```
SELECT movies.*
FROM   movies
WHERE  genre = 'action'
```

Question (2)

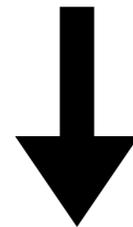Find all action movies viewed by Lebron (id = 103)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(user_id, movie_id, view_timestamp)
```
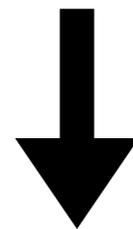
Question (2)

Find all action movies viewed by Lebron (id = 103)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(user_id, movie_id, view_timestamp)
```

⬇

```
SELECT movies.*
FROM   views, movies
WHERE  views.user_id = 103 AND
       views.movie_id = movies.movie_id AND
       movies.genre = 'action'
```

Question (2)

Find all action movies viewed by Lebron (id = 103)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(user_id, movie_id, view_timestamp)
```

↓

```
SELECT movies.*
FROM   views, movies
WHERE  views.user_id = 103 AND
       views.movie_id = movies.movie_id AND
       movies.genre = 'action'
```

Do we need DISTINCT?

Question (3)

Find all action movies viewed by Lebron (id = 103)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```
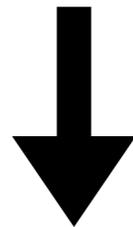
What is the difference?

Question (3)

Find all action movies viewed by Lebron (id = 103)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```

```
SELECT DISTINCT movies.*
FROM   views, movies
WHERE  views.user_id = 103 AND
       views.movie_id = movies.movie_id AND
       movies.genre = 'action'
```
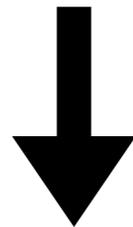
Question (4)

"people who watched American pie (id = 23) also watched"

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```

Question (4)

"people who watched American pie (id = 23) also watched"

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```

⬇

```
SELECT DISTINCT m.*
FROM    movies AS m, views AS v1, views AS v2
WHERE   m.id = v1.movie_id AND
        m.id <> 23 AND
        v1.user_id = v2.user_id AND
        v2.movie_id = 23
```

Question (5)

"people who watched American pie (id = 23) also watched"
(ordered by weekly popularity)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```
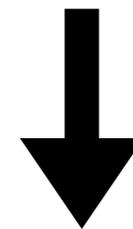
Question (5)

"people who watched American pie (id = 23) also watched"
(ordered by weekly popularity)

```
users(id, name, city, birthdate)
movies(id, name, rating, genre)
views(view_id, user_id, movie_id, view_timestamp)
```

```
SELECT m.id, m.name, count(*)
FROM   movies AS m, views AS v
WHERE  m.id = v.movie_id AND
       v.timestamp > 123456789 AND
       v.id IN (<QUESTION4>)
GROUP BY m.id, m.name
ORDER BY count(*) DESC
```