Introduction to Business Analysis

Lecture 3: Data & Big Data Management. SQL

Iegor Vysnevskyi

Woosong University

March 14/20, 2023

Agenda

1. Importance of Data & Big Data Management in Today's World
2. Overview of the Role of SQL in Data & Big Data Management
3. SQL Databases vs Spreadsheets
4. Databases and Their Structure
5. Tables
6. SQL Data Types
7. SQL Dialects
8. Practical Part
9. In-class assignment

1. Importance of Data & Big Data Management in Today's World

- Data is now considered ***one of the most valuable assets*** for businesses, governments, and organizations of all types and sizes.
- With the rise of digital technologies and the Internet, data is being generated at an unprecedented rate.

Big Data refers to large and complex datasets that are difficult to process using traditional methods.

- Big Data has the potential to provide insights into consumer behavior, market trends, and other important information that can help businesses make better decisions.

- Proper Data Management allows organizations to streamline their operations, reduce costs, and improve their decision-making capabilities.
- The ability to process and analyze Big Data has become a critical skill for businesses, especially in the fields of marketing, finance, and operations.

2. Overview of the Role of SQL in Data & Big Data Management

Structured Query Language (SQL) is a programming language designed for managing and manipulating data in relational databases.

- SQL is a powerful tool for managing and analyzing data, making it an essential skill for anyone interested in a career in business analysis or data management.
- This language is especially useful for managing large and complex datasets, making it a valuable tool for managing Big Data.
- SQL is widely used in business intelligence and data analytics, providing insights into customer behavior, market trends, and other important information.
- Today knowledge of SQL is definitely ***must have skill*** for business analyst.

3. SQL Databases vs Spreadsheets

Filtering, sorting, aggregation, joining of data. All these and many other operations can be done using spreadsheet.

So, when we need SQL?

Features of Spreadsheets	Features of SQL Databases
Smaller data sets	Larger datasets
Enter data manually	Access tables across a database
Create graphs and visualizations in the same program	Prepare data for further analysis in another software
Built-in spell check and other useful functions	Fast and powerful functionality
Best when working solo on a project	Great for collaborative work and tracking queries run by all users

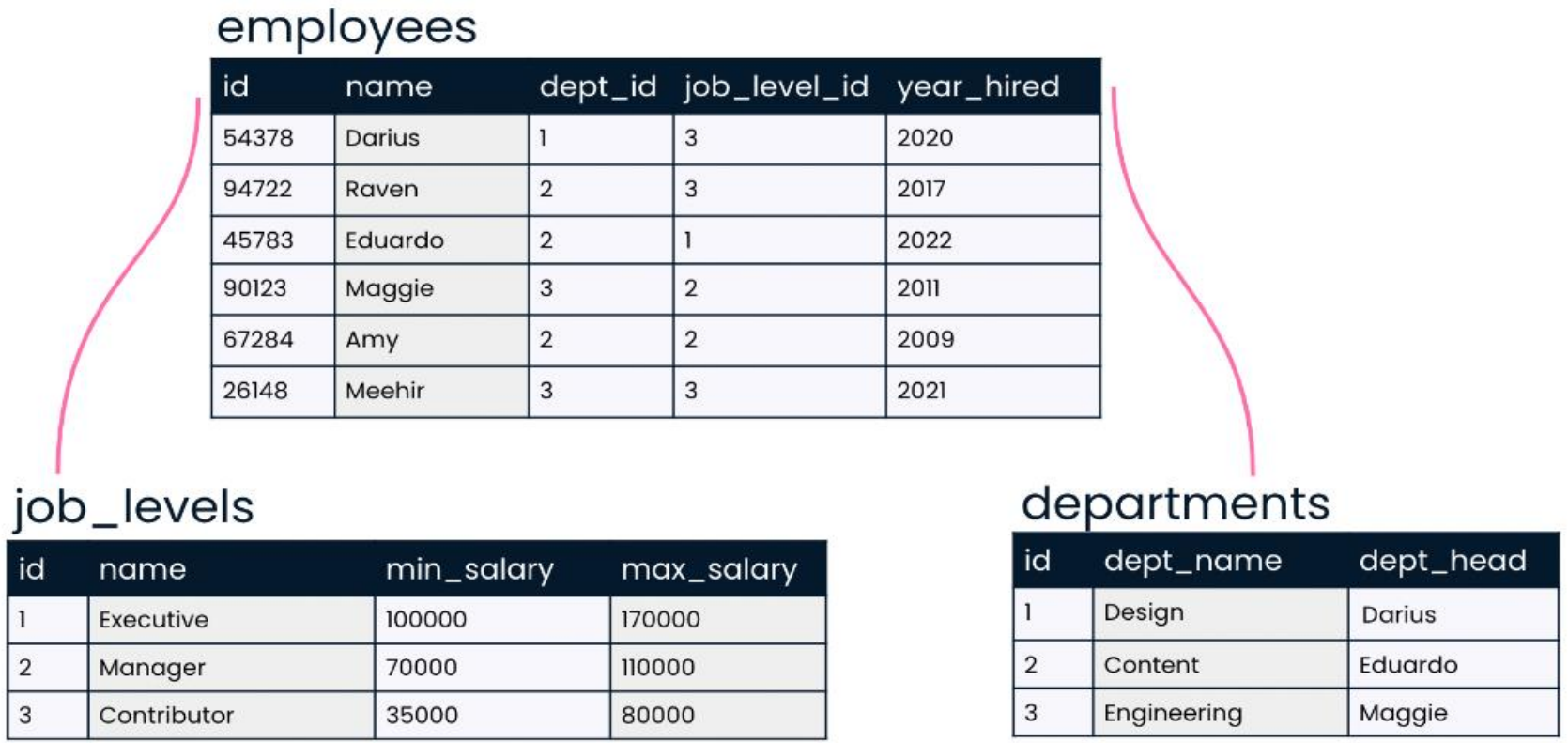
Source: Coursera course “Processing Data from Dirty to Clean”

Conclusion:

- If you are working with data that is already in a spreadsheet, that is most likely where you will perform your analysis.
- If you are working with data stored in a database (several relational tables), SQL will be the best tool for you to use for your analysis.

4. Databases and Their Structure

SQL database is a type of database that stores data in a structured manner using the Structured Query Language.



employees

id	name	dept_id	job_level_id	year_hired
54378	Darius	1	3	2020
94722	Raven	2	3	2017
45783	Eduardo	2	1	2022
90123	Maggie	3	2	2011
67284	Amy	2	2	2009
26148	Meehir	3	3	2021

job_levels

id	name	min_salary	max_salary
1	Executive	100000	170000
2	Manager	70000	110000
3	Contributor	35000	80000

departments

id	dept_name	dept_head
1	Design	Darius
2	Content	Eduardo
3	Engineering	Maggie

This is a relational database containing three tables.

job_level_id in the table ‘employees’ = id in the table ‘job_levels’.

Debt_id in the table ‘employees’ = id in the table ‘departments’.

5. Tables

Basic SQL table terms to be aware of:

A ***record*** is a row that holds data on an individual observation.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

A ***field*** is a column that holds one piece of information about all records.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

Primary keys (unique identifiers) identify records in a table. They are unique and often numbers.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

Foreign keys identifies how tables relate to each other. A foreign key refers to a Primary Key in another table.

You can see example on the next slide.

employees

id	name	dept_id	job_level_id	year_hired
54378	Darius	1	3	2020
94722	Raven	2	3	2017
45783	Eduardo	2	1	2022
90123	Maggie	3	2	2011
67284	Amy	2	2	2009
26148	Meehir	3	3	2021

Primary key

Foreign key

departments

id	dept_name	dept_head
1	Design	Darius
2	Content	Eduardo
3	Engineering	Maggie

Primary key

Best practices that are commonly followed when defining and working with database tables

Table names should:

- be lowercase,
- have no spaces (use underscores instead),
- refer to a collective group or be plural.

Field names should:

- be lowercase,
- have no spaces,
- be singular,
- be different from other field names,
- be different from the table name.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

6. SQL Data Types

Why to be familiar with the SQL data types?

- It will assist you to use database functions properly and write your queries accurately.
- Different types of data are stored differently and take up different space.
- Some operations only apply to certain data types.

Strings sequences of characters such as letters or punctuation.

VARCHAR is a flexible and popular string data type in SQL.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

Integers store whole numbers.

INT is a flexible and popular integer data type in SQL.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

Floats store numbers that include a fractional part.

NUMERIC is a flexible and popular float data type in SQL.

card_num	name	member_year	total_fine
54378	Izzy	2012	9.86
94722	Maham	2020	0
45783	Jasmin	2022	2.05
90123	James	1989	0

7. SQL Dialects

A SQL database system is a type of database management system that is designed to store, organize, and manage data using the SQL (Structured Query Language) programming language.

Most popular SQL database systems:

MySQL, Oracle, SQL Server, PostgreSQL, SQLite etc.

All SQL database systems must follow universal standards set by the International Organization for Standards and the American National Standards Institute.

However, additional features on top of these standards result in different SQL dialects.

Example (both queries will give the same result):

PostgreSQL:

```
SELECT id, name
FROM employees
LIMIT 2;
```

SQL Server:

```
SELECT id, name
FROM employees
TOP 2;
```

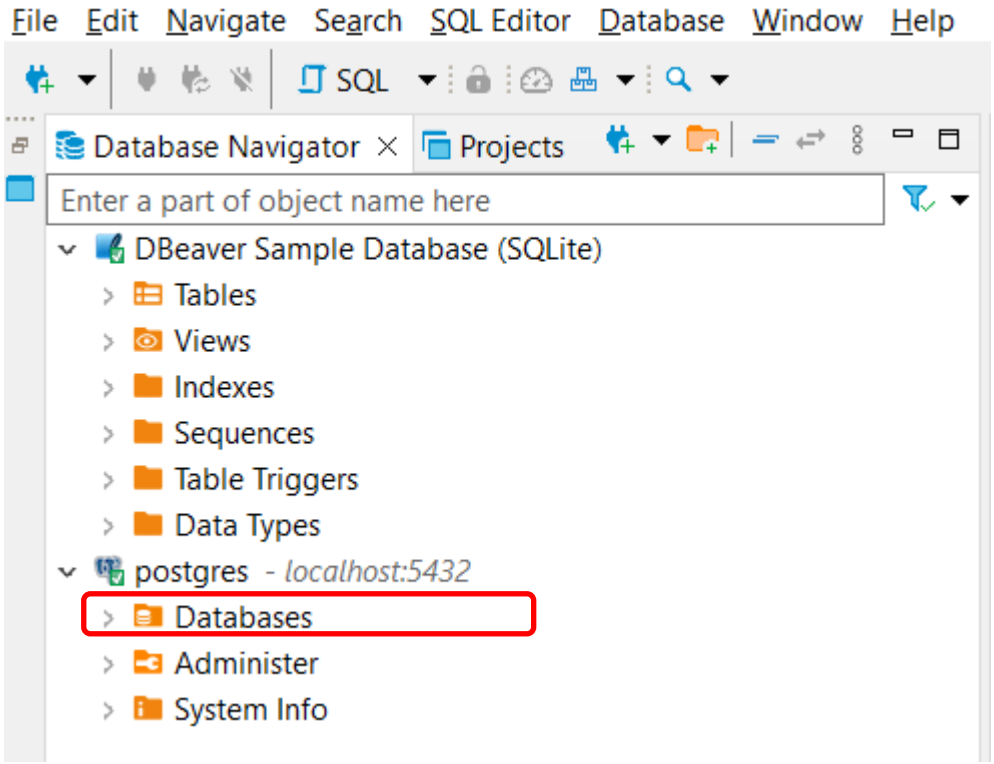
As a beginner, it is important to know that there are slight differences between dialects. But by mastering Standard SQL, which is the dialect you will be working with in this program, you will be prepared to use SQL in any database.

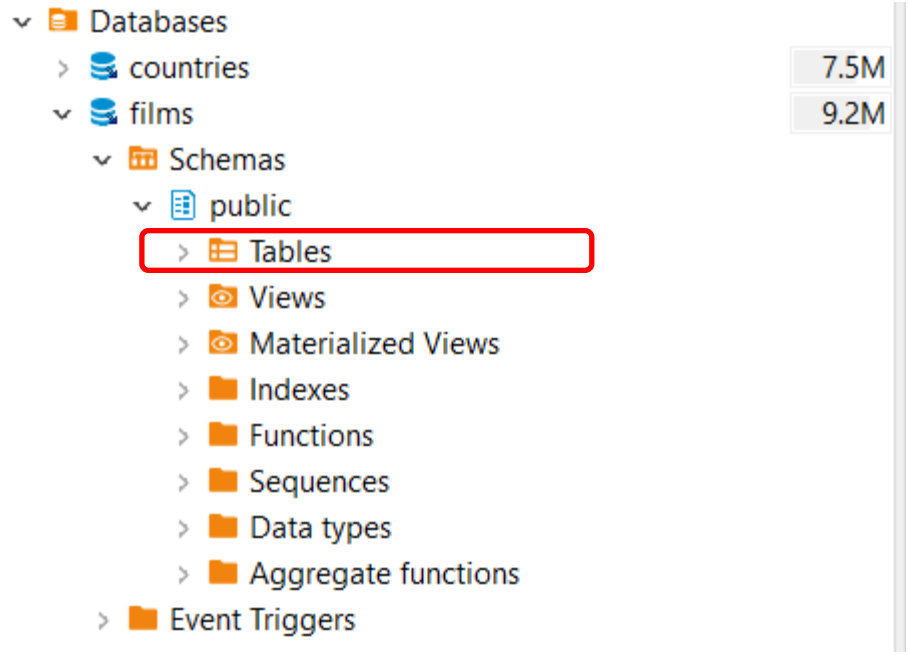
8. Practical Part

Loading the tables

1. Right-click on Databases folder and select “Create new database”.
2. Assign the name for database.

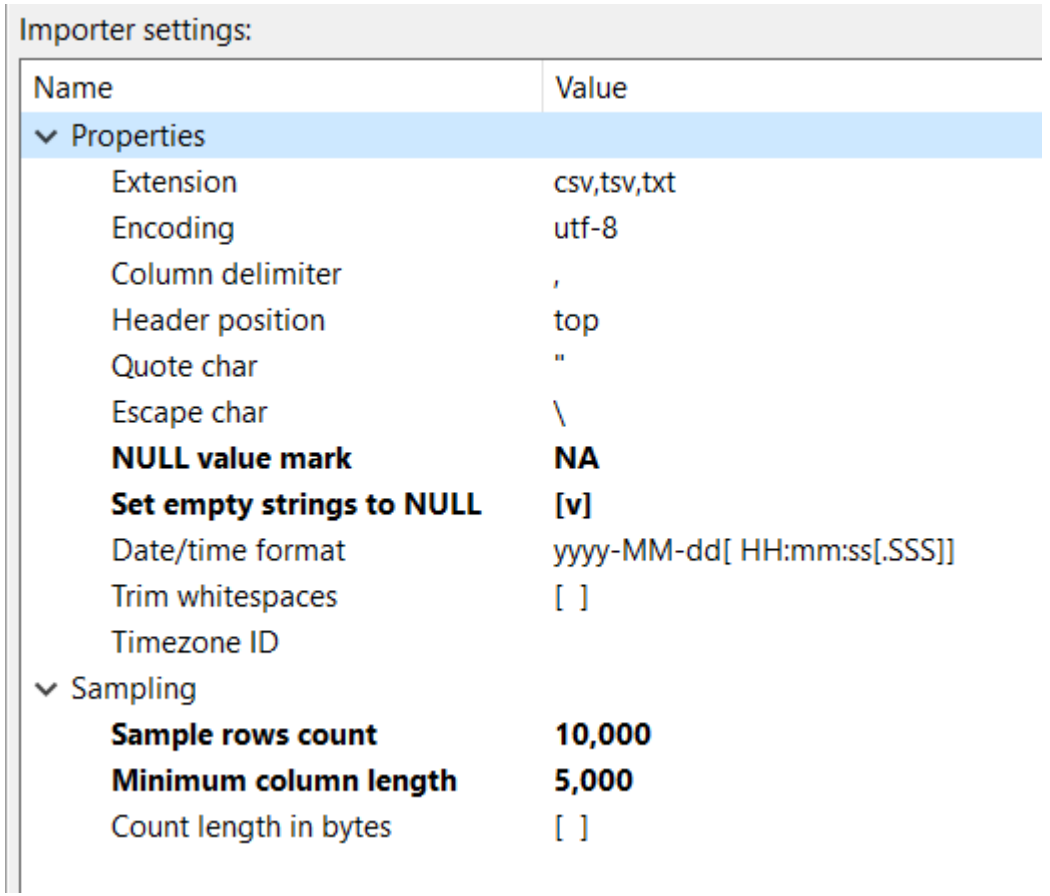
3. Right-click on Tables folder of newly-created database and select “Import data”.
4. Click Input file(s) and select files you want to import.





Loading the tables

5. Check Importer settings to avoid unwanted limitations for the data.

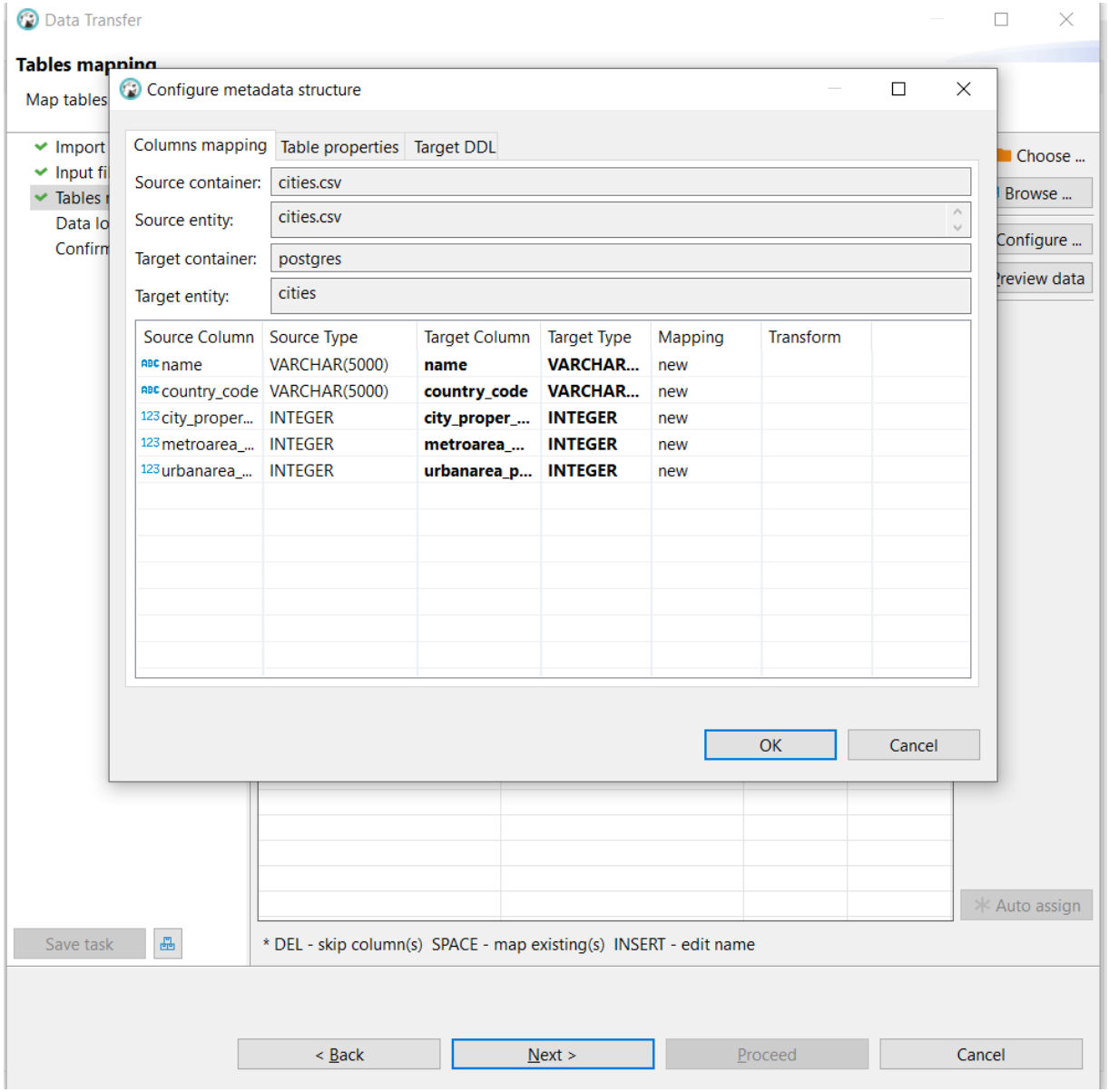


Loading the tables

6. Press “Next” button and “Preview data” to check whether file is imported correctly.

7. Then, press “Configure” button to make sure the data type is assigned the way you need.

8. Press “Proceed” button.



Your First Query

In SQL, a ***query*** is a request for data or information from a database.

- It is a command or a set of commands that are executed against a database in order to retrieve, update, insert, or delete data.
- A query is typically composed of one or more SQL statements, which are used to specify what data should be retrieved from the database, how it should be retrieved, and how it should be formatted.

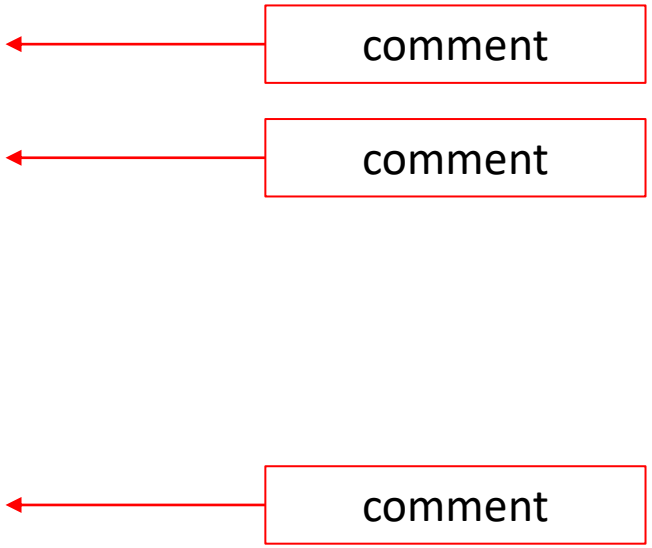
Your First Query

Based on example: table “films”

Query:

```
--look at the table films in SQL as it is
SELECT *        --select all columns
FROM films;

/*choose only
some columns*/
SELECT id, title, release_year, country
FROM films;
```



Good Practice

Lets come back for a moment to the first query and pay attention to some details

Query:

```
SELECT *
FROM films;
```

- Keywords (SELECT, FROM) are written in uppercase to make it easier to visually distinguish them from other parts of the code.
- Semicolon used to indicate the end of a SQL statement, and is typically required to be included at the end of each individual SQL statement.

Your First Query

Query:

```
--count the number of the films
SELECT COUNT(title) AS num_titles
FROM films;
```

*Rename columns in our result set using **aliasing***

- But what if for some chance, there are films with the same titles

Query:

```
--count the number of unique titles
SELECT count(DISTINCT title) AS num_titles
FROM films;
```

Query:

```
SELECT count(DISTINCT country) AS num_countries
FROM films;
```

Filtering

Query:

```
--choose the films made in Germany
SELECT *
FROM films
WHERE country = 'UK';


--choose the films with the budget more than 1,000,000
SELECT *
FROM films
WHERE budget >= 1000000;
```

Multiple Filters

Query:

```
--choose the films made in UK with the budget more than 1,000,000
SELECT *
FROM films
WHERE budget >= 1000000
AND country != 'UK';

--choose the films release in 1930-1940
SELECT *
FROM films
WHERE release_year BETWEEN 1930 AND 1940;
```

Multiple Filters

Code:

```
--choose the films made in UK or Germany
SELECT *
FROM films
WHERE (country = 'UK' OR country = 'Germany‘);
```

Notice that we write “country = “ twice

OR

Code:

```
SELECT *
FROM films
WHERE country IN ('UK', 'Germany');
```

Missing Values

Code:

```
--select all columns from the table 'films' where gross is null value
SELECT *
FROM films
WHERE gross IS NULL;

--select all columns from the table 'films' where gross is NOT null value
SELECT *
FROM films
WHERE gross IS NOT NULL;
```

Grouping

Query:

--find the total budget for the respective year

SELECT release_year, **SUM**(budget)

FROM films

GROUP BY release_year;

Note:

There are a lot of functions that can be used instead of SUM depending on our purposes, such as:

- COUNT - returns the number of rows in each group
- MAX - returns the maximum value in each group
- MIN - returns the minimum value in each group
- AVG - returns the average of a column for each group etc.

More complicated grouping

Query:

```
--find the total budget for each country in respective year
SELECT country, release_year, SUM(budget) AS total_budget
FROM films
GROUP BY country, release_year;
```

Sorting

Query:

```
SELECT release_year, SUM(budget) AS total_budget
FROM films
GROUP BY release_year
--sort the table info by release_year in ascending order
ORDER BY release_year;
```

!!! By default sorting is made in ascending order.

Query:

```
SELECT release_year, SUM(budget) AS total_budget
FROM films
GROUP BY release_year
--sort the table info by release_year in descending order
ORDER BY release_year DESC;
```

Sorting

Query:

```
SELECT release_year, SUM(budget) AS total_budget
FROM films
GROUP BY release_year
ORDER BY release_year DESC
--show only top 5 results
LIMIT 5;
```

All in one

Query:

```
SELECT
        --count number of unique film titles for each year
        COUNT(DISTINCT title) AS film_num,
        release_year,
        --count average budget for each year
        AVG(budget) AS avg_budget
FROM films

--do these calculations only for films released in 1930-1960 and only for USA or UK
WHERE (release_year BETWEEN 1930 AND 1960)
        AND (country = 'USA' OR country = 'UK‘)
GROUP BY release_year
--order by number of films
ORDER BY film_num DESC
--show only top 10 results
LIMIT 10;
```

Statement order

- SQL statements should be executed in a specific order for successful execution and proper results.
- This order helps to ensure that the necessary data is first selected and then processed in a logical sequence before it is finally returned in a specified order.
- The order is particularly important when using clauses like GROUP BY, which rely on data that is generated in earlier statements.
- Failure to follow the correct sequence can lead to syntax errors or incorrect results.

```
SELECT
        COUNT(DISTINCT title) AS film_num,
        release_year,
        AVG(budget) AS avg_budget
FROM films
WHERE (release_year BETWEEN 1930 AND 1960)
        AND (country = 'USA' OR country = 'UK‘)
GROUP BY release_year
ORDER BY film_num DESC
LIMIT 10;
```

Joining tables

Let’s say we want to add to our table “films” info about IMDB score from table “reviews”.

Query:

```
SELECT films.*, reviews.imdb_score
FROM films

LEFT JOIN reviews
ON films.id = reviews.film_id

ORDER BY id;
```

2. Select all columns of table “films” and column “imdb_score” of table “reviews”.

1. Joining tables

a join condition that specifies how the two tables, are related to each other.

- ***films.id*** is a primary key
- ***reviews.film_id*** is a foreign key

Joining tables

- It is rather common situation that two (or more) tables that we need to join can have the same column names.
- To identify which table the column is selected from, in SELECT statement we specify the name of the table.
- Alias is a shorthand way of referring to a table or a column in a query.

Query:

```
SELECT f.*, r.imdb_score
FROM films AS f

LEFT JOIN reviews AS r
ON films.id = reviews.film_id

ORDER BY id;
```

Creating and Deleting Tables

- It is rather common situation that two (or more) tables that we need to join can have the same column names.
- To identify which table the column is selected from, in SELECT statement we specify the name of the table.
- Alias is a shorthand way of referring to a table or a column in a query.

Query:

```
CREATE TABLE employees (
employee_id INT PRIMARY KEY,    --column name/ data type/ limitation
first_name VARCHAR NOT NULL,
last_name VARCHAR NOT NULL,
age INT,
email VARCHAR UNIQUE)
```

Creating and Delating Tables

Query:

```
--delete table
DROP TABLE employees;


--create another table for further practice
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
first_name VARCHAR NOT NULL,
last_name VARCHAR NOT NULL,
age INT)
```

Modifying Tables

Query:

```
ALTER TABLE employees
ADD email VARCHAR UNIQUE,      --add new column
ALTER COLUMN age
        SET DATA TYPE VARCHAR,  --change the data type of the column
DROP COLUMN first_name;         --delete column
```

Rename the table

Query:

```
ALTER TABLE employees
RENAME TO employees_last_name;
```

Adding Values to the Table

Query:

```
INSERT
        INTO employees_last_name (employee_id, last_name, age, email)
VALUES      (1, 'Smith', 30, 'smith@example.com'),
            (2, 'Johnson', 35, 'johnson@example.com'),
            (3, 'Williams', 28, 'williams@example.com');
```

9. In-class assignment

- You will be working with the database which contains information about most populous cities in the world.
- There are two tables in the database.
- Open them using SQL and quickly look at the tables to get a sense of what they contain.