DataScientest • com

Introduction to SQL

SQL queries

In the previous notebook we saw the structure of relational databases. In this notebook, you will learn the syntax of **SQL queries** with the help of a **widget** that we have developed. The widget allows you to execute queries directly on the database, in the usual Python environment of the platform, and **with the SQL syntax**.

How to use the widget

To launch the widget, you just have to execute the cells containing the code:

```
from Widget_Template import build_SQL_widget
build_SQL_widget("name_of_the_database.db")
```

This command will bring up an interface allowing you to run SQL queries on the given database. In order to do so, you must specify your query as an argument to the `build_SQL_widget` function:

This widget is based on the **SQLAlchemy** toolkit which will be presented in notebook 5. More precisely the **DBMS** used here is **SQLite**. Some queries may therefore be different (or unavailable) from other DBMSs. If this is the case, these queries will be indicated and you will be able to find a reference on the alternative way to proceed with SQLite.

The projection

The **projection is the fundamental query in SQL**. It consists in displaying some columns of a table thanks to the keyword `SELECT`.

To display the whole **table**, you can select all the columns with the asterisk `*`.

We then specify the name of the table with the keyword `FROM`.

It will be useful to get into the habit of returning to the line after each command, and to mark the end of the query with the symbol `;`.

The query will therefore take this form:

```
SELECT *
FROM table;
```

- **a)** Run the following cell to import and instantiate the widget, and to specify the database that we want to query: `chinook.db`.
- **b)** Use the Widget to display through an SQL query all columns of the **`albums`** table.

The table **`albums`** looks like this:

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2

In [1]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

Select *
From albums;

✔ Run Command

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4

Hide solution

In [ ]:
```
'''
SELECT *
FROM albums;
'''
```

From here on, you won't need to call the widget anymore: the cells should appear automatically. If a problem occurs or you want to add a SQL cell, just add a code cell and execute the `build_SQL_widget("chinook.db")` function again.

> To **show only a subset of a table**, we enter the names of the selected columns after the `SELECT` and we separate them with a comma.

- **c)** Display the `AlbumId` and `ArtistId` columns from the **`albums`** table.

In [3]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
VBox(children=(Textarea(value='', layout=Layout(flex_flow='row', height='130px', max_height='200px', max_width…
```

Hide solution

In [ ]:
```
'''
SELECT AlbumId, ArtistId
FROM albums;
'''
```

> To **avoid repetition** in a projection, you can use the keyword `DISTINCT` before the column names.

- **d)** Display all distinct `ArtistId` of the table **`albums`** .

In [2]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select Distinct ArtistId
From albums ;
```

Run Command

ArtistId
1
2
3
4
5
6

Hide solution

In [ ]:
```
'''
SELECT DISTINCT ArtistId
FROM albums;
'''
```

- **e)** Display the name, first name and position of the employees, contained in the **`employees`** table. You will first need to identify the corresponding columns.

> The **`employees`** table looks like this:

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
1	Adams	Andrew	General Manager	NaN	1962-02-18 00:00:00	2002-08-14 00:00:00	11120 Jasper Ave NW	Edmonton	AB	Canada	T5K 2N1	+1 (780) 428-9482	+1 (780) 428-3457	andrew@chinookcorp.com (mailto:andrew@chinookcorp.com)
2	Edwards	Nancy	Sales Manager	1.0	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (403) 262-3443	+1 (403) 262-3322	nancy@chinookcorp.com (mailto:nancy@chinookcorp.com)
3	Peacock	Jane	Sales Support Agent	2.0	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW	Calgary	AB	Canada	T2P 5M5	+1 (403) 262-3443	+1 (403) 262-6712	jane@chinookcorp.com (mailto:jane@chinookcorp.com)

In [3]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select FirstName, LastName, Title
From employees ;
```

✔ Run Command

FirstName	LastName	Title
Andrew	Adams	General Manager
Nancy	Edwards	Sales Manager
Jane	Peacock	Sales Support Agent
Margaret	Park	Sales Support Agent
Steve	Johnson	Sales Support Agent
Michael	Mitchell	IT Manager

Hide solution

In [ ]:
```
'''
SELECT LastName, Firstname, Title
FROM employees;
'''
```

The simple selection

The **simple selection** consists in choosing elements of a table checking a particular condition. We use the `SELECT` command in association with the `WHERE` keyword to define the condition(s). The general syntax is as follows:

```
SELECT column1, column2
FROM table1
WHERE condition;
```

1. Conditional operators

The following are the main **operators** to build conditions is SQL:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
IN	TRUE if the operand is equal to one of a list of expressions possibles
BETWEEN	TRUE if the operand is within the range of comparisons
LIKE	The LIKE operator is used in a WHERE clause to search for a specified pattern in a column
IS NULL	The IS NULL operator is used to test for empty values (NULL values)
IS NOT NULL	The IS NOT NULL condition is used in SQL to test for a non-NULL value.

It is also **possible to combine several conditions** with the logical operators `AND` , `OR` and `NOT` .

Here you will find some queries to put these notions into practice. We will use the **`employees`** table.

- **a)** Write an SQL query to fetch the employees living in `'Calgary'` .

In [4]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From employees
Where City = 'Calgary' ;
```

✔ Run Command

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
2	Edwards	Nancy	Sales Manager	1	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (403) 262-3443	+1 (403) 262-3322	nancy@chinookcorp.com
3	Peacock	Jane	Sales Support Agent	2	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW	Calgary	AB	Canada	T2P 5M5	+1 (403) 262-3443	+1 (403) 262-6712	jane@chinookcorp.com
4	Park	Margaret	Sales Support Agent	2	1947-09-19 00:00:00	2003-05-03 00:00:00	683 10 Street SW	Calgary	AB	Canada	T2P 5G3	+1 (403) 263-4423	+1 (403) 263-4289	margaret@chinookcorp.com
5	Johnson	Steve	Sales Support Agent	2	1965-03-03 00:00:00	2003-10-17 00:00:00	7727B 41 Ave	Calgary	AB	Canada	T3B 1Y7	1 (780) 836-9987	1 (780) 836-9543	steve@chinookcorp.com

Hide solution

In [ ]:
```
'''
SELECT *
FROM employees
WHERE City = 'Calgary';
'''
```

- **b)** Write an SQL query to fetch the employees living in `'Edmonton'` or in `'Lethbridge'`.

In [5]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From employees
Where City in ('Edmonton', 'Lethbridge') ;
```

Run Command

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
1	Adams	Andrew	General Manager	NaN	1962-02-18 00:00:00	2002-08-14 00:00:00	11120 Jasper Ave NW	Edmonton	AB	Canada	T5K 2N1	+1 (780) 428-9482	+1 (780) 428-3457	andrew@chinookcorp.com
7	King	Robert	IT Staff	6.0	1970-05-29 00:00:00	2004-01-02 00:00:00	590 Columbia Boulevard West	Lethbridge	AB	Canada	T1K 5N8	+1 (403) 456-9986	+1 (403) 456-8485	robert@chinookcorp.com
8	Callahan	Laura	IT Staff	6.0	1968-01-09 00:00:00	2004-03-04 00:00:00	923 7 ST NW	Lethbridge	AB	Canada	T1H 1Y8	+1 (403) 467-3351	+1 (403) 467-8772	laura@chinookcorp.com

Hide solution

In [ ]:
```
'''
SELECT *
FROM employees
WHERE City = 'Edmonton' OR City = 'Lethbridge';
'''
```

- **c)** Write an SQL query to display the name, the first name and the position of the employees who have a hierarchical superior (the column `ReportsTo` contains the Id of the employee's hierarchical superior).

In [6]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select FirstName, LastName, Title
From employees
Where ReportsTo IS NOT NULL;
```

Run Command

FirstName	LastName	Title
Nancy	Edwards	Sales Manager
Jane	Peacock	Sales Support Agent
Margaret	Park	Sales Support Agent
Steve	Johnson	Sales Support Agent
Michael	Mitchell	IT Manager
Robert	King	IT Staff

Hide solution

In [ ]:
```
'''
SELECT LastName, FirstName, Title
FROM employees
WHERE ReportsTo IS NOT NULL;
'''
```

- **d)** Write an SQL query to display employees whose ID is smaller than 5.

In [7]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From employees
Where EmployeeId <=5
```

Run Command

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
1	Adams	Andrew	General Manager	NaN	1962-02-18 00:00:00	2002-08-14 00:00:00	11120 Jasper Ave NW	Edmonton	AB	Canada	T5K 2N1	+1 (780) 428-9482	+1 (780) 428-3457	andrew@chinookcorp.com
2	Edwards	Nancy	Sales Manager	1.0	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (403) 262-3443	+1 (403) 262-3322	nancy@chinookcorp.com
3	Peacock	Jane	Sales Support Agent	2.0	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW	Calgary	AB	Canada	T2P 5M5	+1 (403) 262-3443	+1 (403) 262-6712	jane@chinookcorp.com
4	Park	Margaret	Sales Support Agent	2.0	1947-09-19 00:00:00	2003-05-03 00:00:00	683 10 Street SW	Calgary	AB	Canada	T2P 5G3	+1 (403) 263-4423	+1 (403) 263-4289	margaret@chinookcorp.com

Hide solution

In [ ]:
```
'''
SELECT *
FROM employees
WHERE EmployeeId <= 5;
'''
```

- **e)** Write an SQL query to display employees whose ID is between 2 and 7 (included).

In [8]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From employees
Where EmployeeId Between 2 and 7 ;
```

✔ Run Command

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
2	Edwards	Nancy	Sales Manager	1	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (403) 262-3443	+1 (403) 262-3322	nancy@chinookcorp.com
3	Peacock	Jane	Sales Support Agent	2	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW	Calgary	AB	Canada	T2P 5M5	+1 (403) 262-3443	+1 (403) 262-6712	jane@chinookcorp.com
4	Park	Margaret	Sales Support Agent	2	1947-09-19 00:00:00	2003-05-03 00:00:00	683 10 Street SW	Calgary	AB	Canada	T2P 5G3	+1 (403) 263-4423	+1 (403) 263-4289	margaret@chinookcorp.com
5	Johnson	Steve	Sales Support	2.0	1965-03-03	2003-10-17	7727B 41 Ave	Calgary	AB	Canada	T3B 1Y7	1 (780)	1 (780)	steve@chinookcorp.com

Hide solution

In [ ]:
```
'''
SELECT *
FROM employees
WHERE EmployeeId >= 2 AND EmployeeId <= 7;
'''
#or
'''
SELECT *
FROM employees
WHERE EmployeeId BETWEEN 2 AND 7;
'''
```

- **f)** Write an SQL query to display employees born before May 10, 1973. Pay attention to the date format.

In [9]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From employees
Where BirthDate <= '1973-05-10' ;
```

✔ Run Command

EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
1	Adams	Andrew	General Manager	NaN	1962-02-18 00:00:00	2002-08-14 00:00:00	11120 Jasper Ave NW	Edmonton	AB	Canada	T5K 2N1	+1 (780) 428-9482	+1 (780) 428-3457	andrew@chinookcorp.com
2	Edwards	Nancy	Sales Manager	1.0	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (403) 262-3443	+1 (403) 262-3322	nancy@chinookcorp.com
4	Park	Margaret	Sales Support Agent	2.0	1947-09-19 00:00:00	2003-05-03 00:00:00	683 10 Street SW	Calgary	AB	Canada	T2P 5G3	+1 (403) 263-4423	+1 (403) 263-4289	margaret@chinookcorp.com
5	Johnson	Steve	Sales Support	2.0	1965-03-03	2003-10-17	7727B 41 Ave	Calgary	AB	Canada	T3B 1Y7	1 (780)	1 (780)	steve@chinookcorp.com

Hide solution

In [ ]:
```
'''
SELECT *
FROM employees
WHERE BirthDate <= '1973-05-10';
'''
```

2. Regular Expressions (Regex)

As seen in module **131 - Text Mining -**, **regular expressions** also called **regex**, are strings that describe, according to a precise syntax, a set of possible characters. The regex syntax in SQL is **slightly different from the regex syntax in Python**.

Here are the most frequently used regular expressions:

Regular Expressions	Description
%	matches any combination of characters of any size
_	matches any single character
+	matches at least one instance of the previous expression
^	corresponds to the beginning of the line
$	searches at the end of the line
<	matches only if the word starts at this point
>	matches only if the word ends at this point
\n	matches if there is a line break
[ ]	matches if any of the characters between brackets are present
[^...]	matches if any of the characters between brackets and before the ^ are present
[ABQ]%	the string must start with an A, B or Q and can be of any length
[AB][CD]%	the string must start with an A or B and the second character must be C or D and can be of any length

In [10]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Name Like 'P%' ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99
6	Put The Finger On You	1	1	1	Angus Young, Malcolm Young, Brian Johnson	205662	6713451	0.99
19	Problem Child	4	1	1	AC/DC	325041	10617116	0.99
40	Perfect	6	1	1	Alanis Morissette & Glenn Ballard	188133	6145404	0.99
59	Put You Down	7	1	1	Jerry Cantrell	196231	6420530	0.99
66	Por Causa De Você	8	1	2	None	169900	5536496	0.99

Hide solution

In [ ]:
```
'''
SELECT *
FROM tracks
WHERE Name LIKE 'P%';
'''
```

- **h)** Write an SQL query to display information for 3-character titles

In [11]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Name Like '__' ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
217	Mel	21	1	7	Caetano Veloso - Waly Salomão	294765	9854062	0.99
445	She	37	1	1	Gene Simmons, S. Coronel	248346	8229734	0.99
474	She	39	1	4	Billie Joe Armstrong -Words Green Day -Music	134164	4425128	0.99
992	DOA	79	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett	252186	8232342	0.99
1010	Low	81	1	4	Foo Fighters	268120	8847196	0.99
1699	Giz	140	1	7	Dado Villa-Lobos/Marcelo Bonfá	202213	6677671	0.99

Hide solution

In [ ]:
```
'''
SELECT *
FROM tracks
WHERE Name Like '___';
'''
```

- **i)** Write an SQL query to display titles of less than 5 characters.

In [12]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Name NOT LIKE '____%' ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
159	FX	17	1	3	Tony Iommi, Bill Ward, Geezer Butler, Ozzy Osbourne	103157	3331776	0.99
212	Drão	21	1	7	Gilberto Gil	156264	5065932	0.99
217	Mel	21	1	7	Caetano Veloso - Waly Salomão	294765	9854062	0.99
250	Macô	24	1	7	Chico Science	249600	8253934	0.99
445	She	37	1	1	Gene Simmons, S. Coronel	248346	8229734	0.99

Hide solution

In [ ]:
```
'''
SELECT *
FROM tracks
WHERE Name NOT LIKE '_____%';
'''
```

- **j)** Write a SQL query to display tracks composed by `'Jimi Hendrix'`. Attention, according to the albums the first name and the name can be reversed.

In [13]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Composer Like '%Jimi Hendrix%' ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1479	Foxy Lady	120	1	1	Jimi Hendrix	199340	6480896	0.99
1480	Manic Depression	120	1	1	Jimi Hendrix	222302	7289272	0.99
1481	Red House	120	1	1	Jimi Hendrix	224130	7285851	0.99
1482	Can You See Me	120	1	1	Jimi Hendrix	153077	4987068	0.99
1483	Love Or Confusion	120	1	1	Jimi Hendrix	193123	6329408	0.99
1484	I Don't Live Today	120	1	1	Jimi Hendrix	235311	7661214	0.99

Hide solution

In [ ]:
```
'''
SELECT *
FROM tracks
WHERE Composer LIKE '%Jimi%' AND Composer LIKE '%Hendrix%';
'''
#This syntax allows us to get all the titles whose composer names include Jimi and Hendrix,
#in any order.
```

- **k)** Write a SQL query to display music from an association of several artists.

> Hint: For an association of several artists, the name is separated by a / or - except for the group AC/DC.

In [15]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Composer LIKE '%-%' or (Composer LIKE '%/%' and Composer not Like ' __/__') ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. Dirkscneider & W. Hoffman	252051	4331779	0.99
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99
15	Go Down	4	1	1	AC/DC	331180	10847611	0.99
16	Dog Eat Dog	4	1	1	AC/DC	215196	7032162	0.99
17	Let There Be Rock	4	1	1	AC/DC	366654	12021261	0.99
18	Bad Boy Boogie	4	1	1	AC/DC	267728	8776140	0.99

Hide solution

In [ ]:
```
'''
SELECT Name, Composer
FROM tracks
WHERE (Composer LIKE '%/%' AND Composer NOT LIKE '__/__') OR Composer LIKE '%-%';
'''
```

Data Sorting

> It is possible to sort the results of a query by column values in ascending order by using the `ORDER BY` keyword with `ASC` (default), or `DESC` (to sort in descending order).
>
> Answer the following questions using the **`tracks`** table.

- **l)** Sort the music titles in alphabetical order: we will limit the query to the first 10 titles.

> The keyword `LIMIT` allows to choose the number of results to display.

In [16]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Order By Name LIMIT 10 ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
3027	"40"	239	1	1	U2	157962	5251767	0.99
2918	"?"	231	3	19	None	2782333	528227089	1.99
3412	"Eine Kleine Nachtmusik" Serenade In G, K. 525: I. Allegro	281	2	24	Wolfgang Amadeus Mozart	348971	5760129	0.99
109	#1 Zero	11	1	4	Cornell, Commerford, Morello, Wilk	299102	9731988	0.99
3254	#9 Dream	255	2	9	None	278312	4506425	0.99
602	'Round Midnight	48	1	2	Miles Davis	357459	11590284	0.99

Hide solution

In [ ]:
```
'''
SELECT Name
FROM tracks
ORDER BY Name ASC LIMIT 10;
'''
#ou puisque ASC est le paramètre par défaut du mot-clé ORDER BY
'''
SELECT Name
FROM tracks
ORDER BY Name LIMIT 10;
'''
```

- **l)** Write a SQL query to display AC/DC titles in alphabetical order.

In [17]:
```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Where Composer = 'AC/DC'
Order By Name ;
```

✔ Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
18	Bad Boy Boogie	4	1	1	AC/DC	267728	8776140	0.99
16	Dog Eat Dog	4	1	1	AC/DC	215196	7032162	0.99
15	Go Down	4	1	1	AC/DC	331180	10847611	0.99
21	Hell Ain't A Bad Place To Be	4	1	1	AC/DC	254380	8331286	0.99
17	Let There Be Rock	4	1	1	AC/DC	366654	12021261	0.99
20	Overdose	4	1	1	AC/DC	369319	12066294	0.99

Hide solution

In [ ]:
```
'''
SELECT Name, Composer
FROM tracks
WHERE Composer LIKE '%AC/DC%'
ORDER BY Name;
'''
```

- **m)** Display the 5 longest tracks in the **`tracks`** table as well as their duration.

> We can use the column `Milliseconds` .

In [18]:

```
from Widget_Template import build_SQL_widget
build_SQL_widget("chinook.db")
```

```
Select *
From tracks
Order By Milliseconds DESC LIMIT 5 ;
```

Run Command

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
2820	Occupation / Precipice	227	3	19	None	5286953	1054423946	1.99
3224	Through a Looking Glass	229	3	21	None	5088838	1059546140	1.99
3244	Greetings from Earth, Pt. 1	253	3	20	None	2960293	536824558	1.99
3242	The Man With Nine Lives	253	3	20	None	2956998	577829804	1.99
3227	Battlestar Galactica, Pt. 2	253	3	20	None	2956081	521387924	1.99

Hide solution

In [ ]:

```
'''
SELECT Name, Milliseconds
FROM tracks
ORDER BY Milliseconds DESC LIMIT 5;
'''
```

Conclusion

In this notebook we learned how to make SQL queries to examine a database

Here are the main operations that you need to remember:

- **Projection** :

```
SELECT col1, col2
FROM table;
```

- **Simple Selection** :

```
SELECT *
FROM table
WHERE condition;
```

- **Conditional operators and regular expressions with `LIKE`** .
- **Data Sorting** :

```
SELECT *
FROM table
ORDER BY col ASC;
```

Unvalidate