Data Science Foundations

Master in Big Data Solutions 2017-2018

Liana Napalkova
liana.napalkova@bts.tech

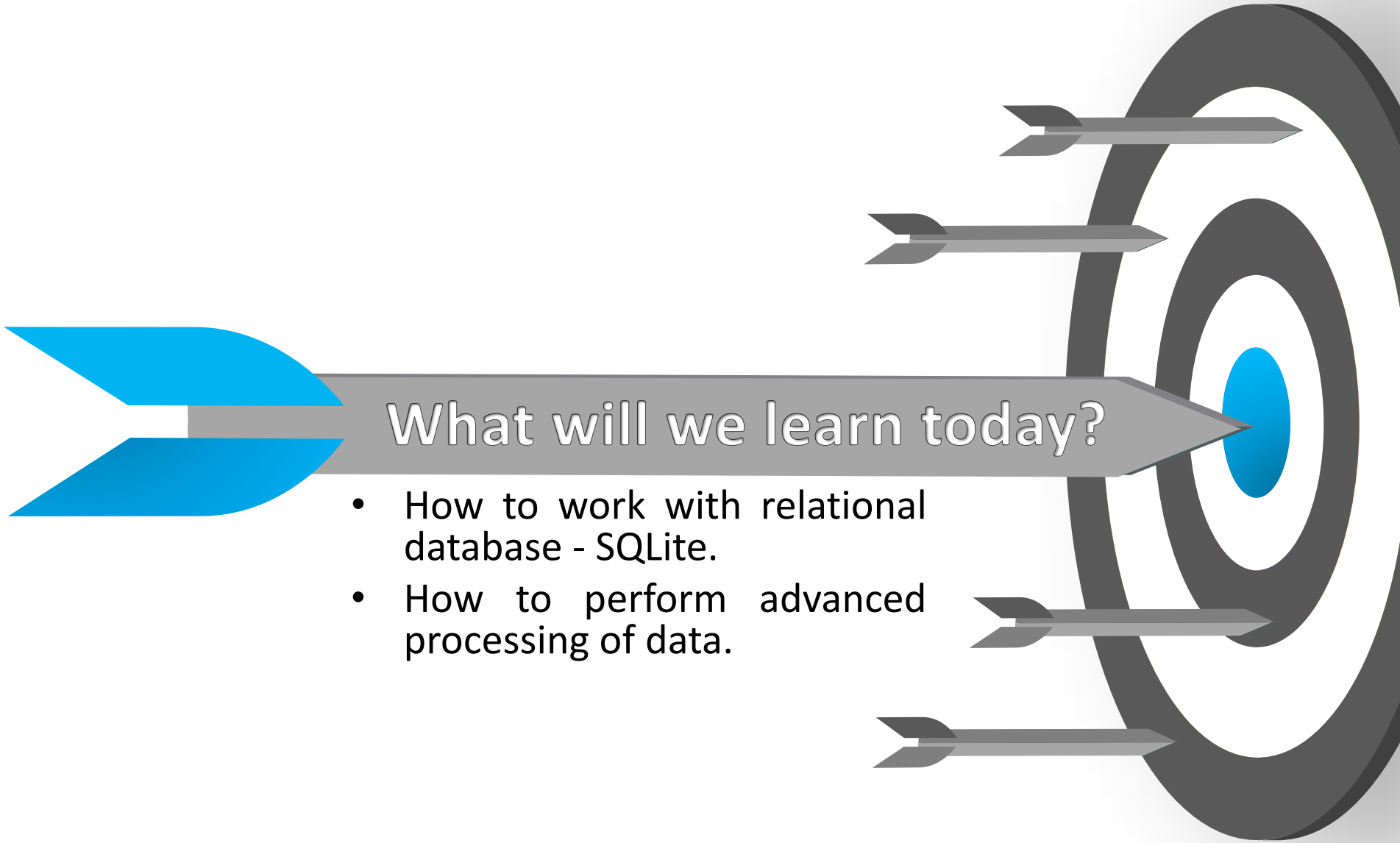
Ludovico Boratto
ludovico.boratto@bts.tech

Francisco Guitierres
francisco.gutierres@bts.tech

BTS | Barcelona Technology School

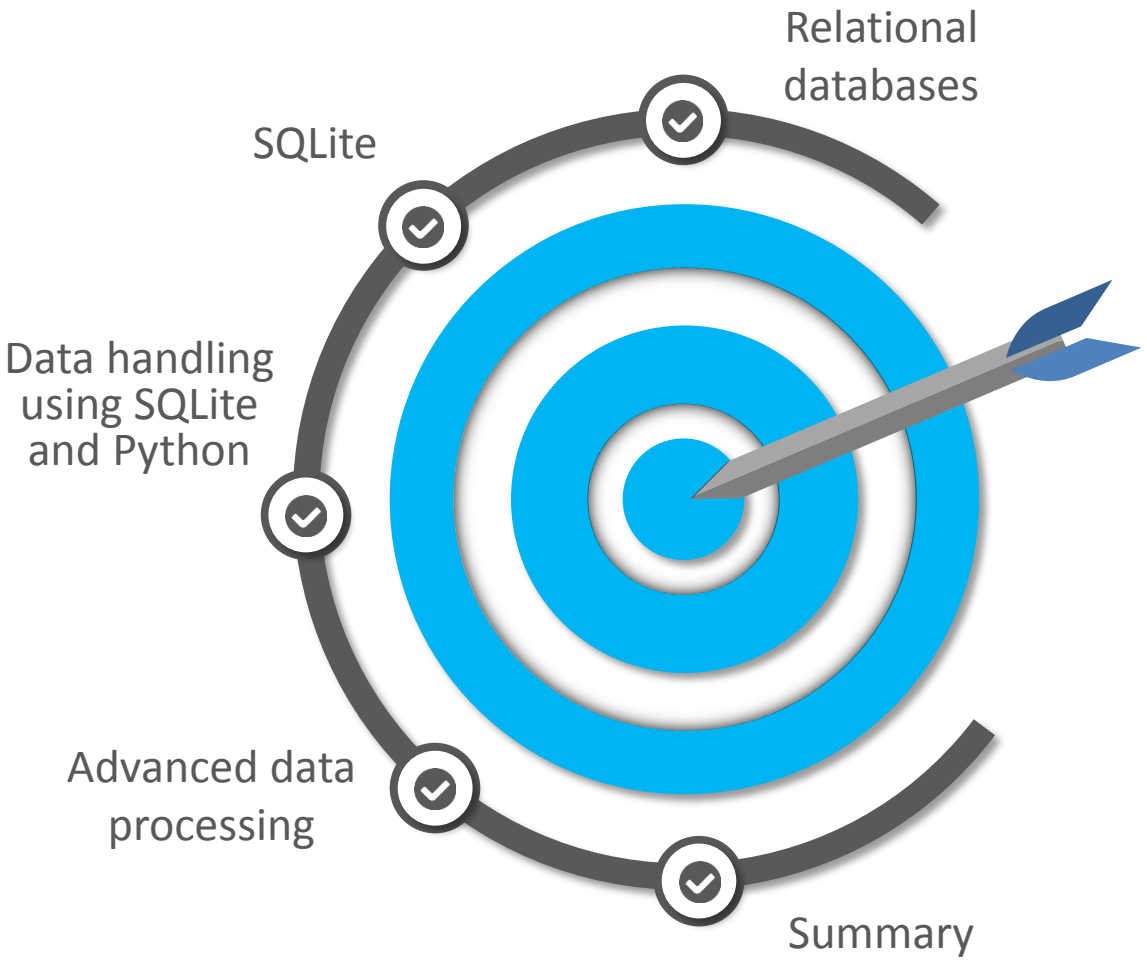
eurecat Centre Tecnològic de Catalunya

Today's Objective



- How to work with relational database - SQLite.
- How to perform advanced processing of data.

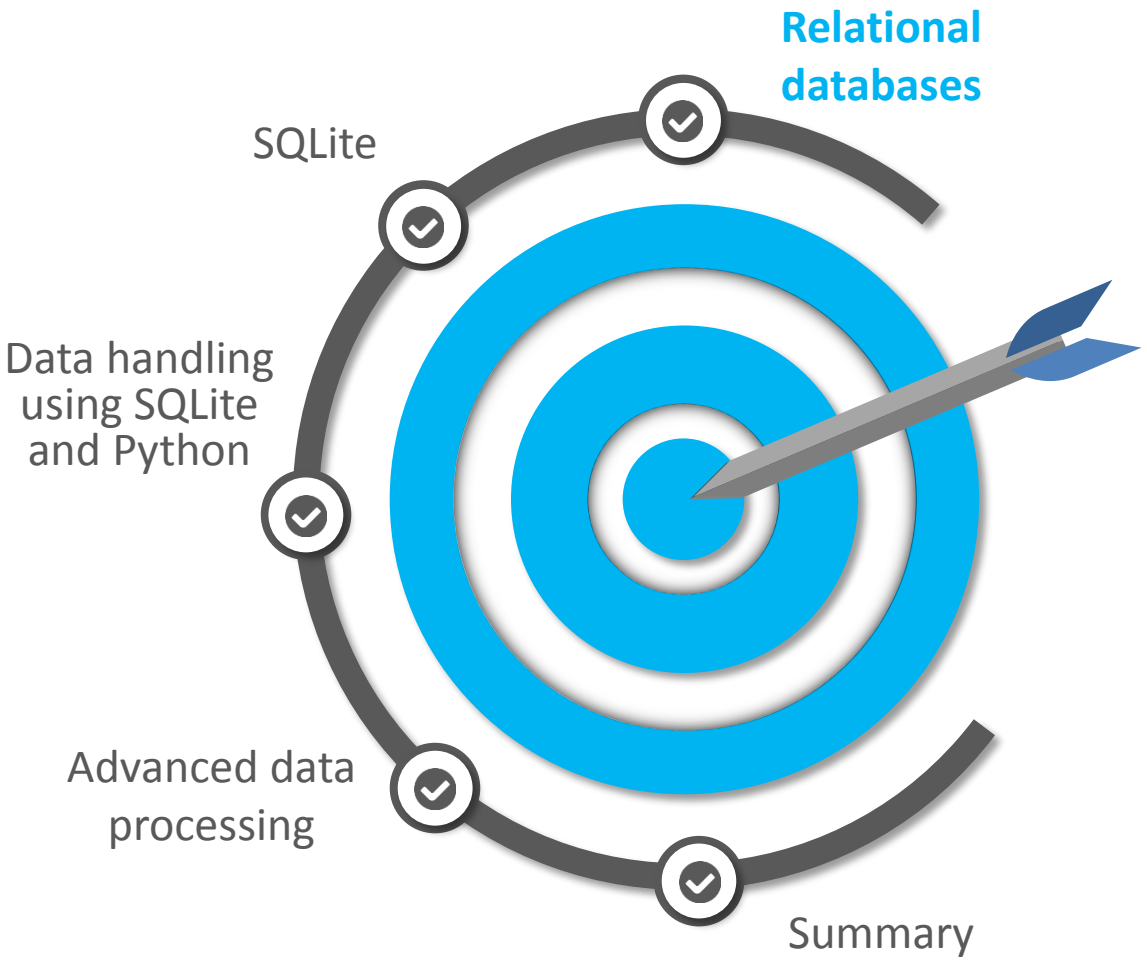
Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

Relational databases

- In a real world the data is often stored in relational databases.
- Relational databases store the data as tables (rows and columns).
- The power of the relational database lies in its ability to efficiently retrieve data from those tables using the query language.

Relational databases

- **Database** - contains a set of tables.
- **Table** - contains tuples and attributes.
- **Tuple (or row)** - a set of fields that generally represents an "object", for example, a book, a student, etc.
- **Attribute (also column or field)** – a feature that describes the object represented by the row, for example:
  - ✓ The object "book" might have attributes like "title", "authors", "number of pages", etc.
  - ✓ Which are possible attributes of the object "student"?

Relational databases

Database "university"

students

name	surnames	email	country	course
John	Berry	jb@uni.edu	UK	1
Bianca	Mendez	bm@uni.edu	Venezuela	1
Bruno	Fiadino	bf@uni.edu	Italy	2

courses

Course ID	Course title
1	Course #1
2	Course #2

Relational databases

Structured Query Language is the language we use to issue commands to the database:

- Create a table
- Retrieve some data
- Insert data
- Delete data

Relational databases

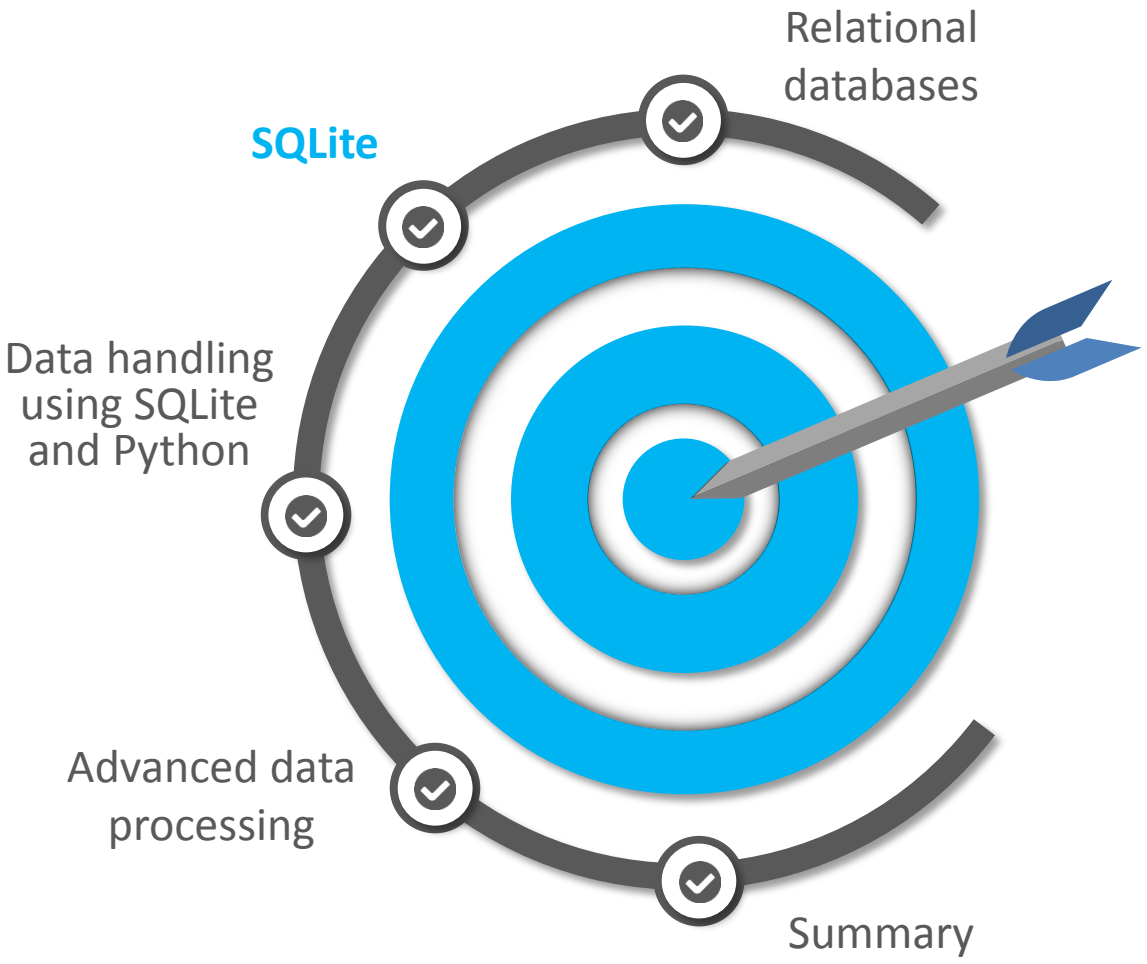
Three Database Management Systems in wide use:

- **Oracle** - Large, commercial, enterprise-scale.
- **MySQL** - Simple but very fast and scalable - commercial open source.
- **SQL Server** – Microsoft's enterprise-scale solution.

Many other smaller projects, free and open source:

- **SQLite**,
- **PostgreSQL**,
- and others.

Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

SQLite

- SQLite is a very popular database - it is free and fast and small.
- SQLite Browser allows us to directly manipulate SQLite files.
- SQLite is embedded in Python and a number of other languages.
- Download link:

http://sqlitebrowser.org/

Relational databases

Database "university"

students

s_pk	name	surnames	email	country	c_pk
1	John	Berry	jb@uni.edu	UK	1
2	Bianca	Mendez	bm@uni.edu	Venezuela	1
3	Bruno	Fiadino	bf@uni.edu	Italy	2

courses

c_pk	title
1	Course #1
2	Course #2

SQLite

Start simple

We will do the following exercise:

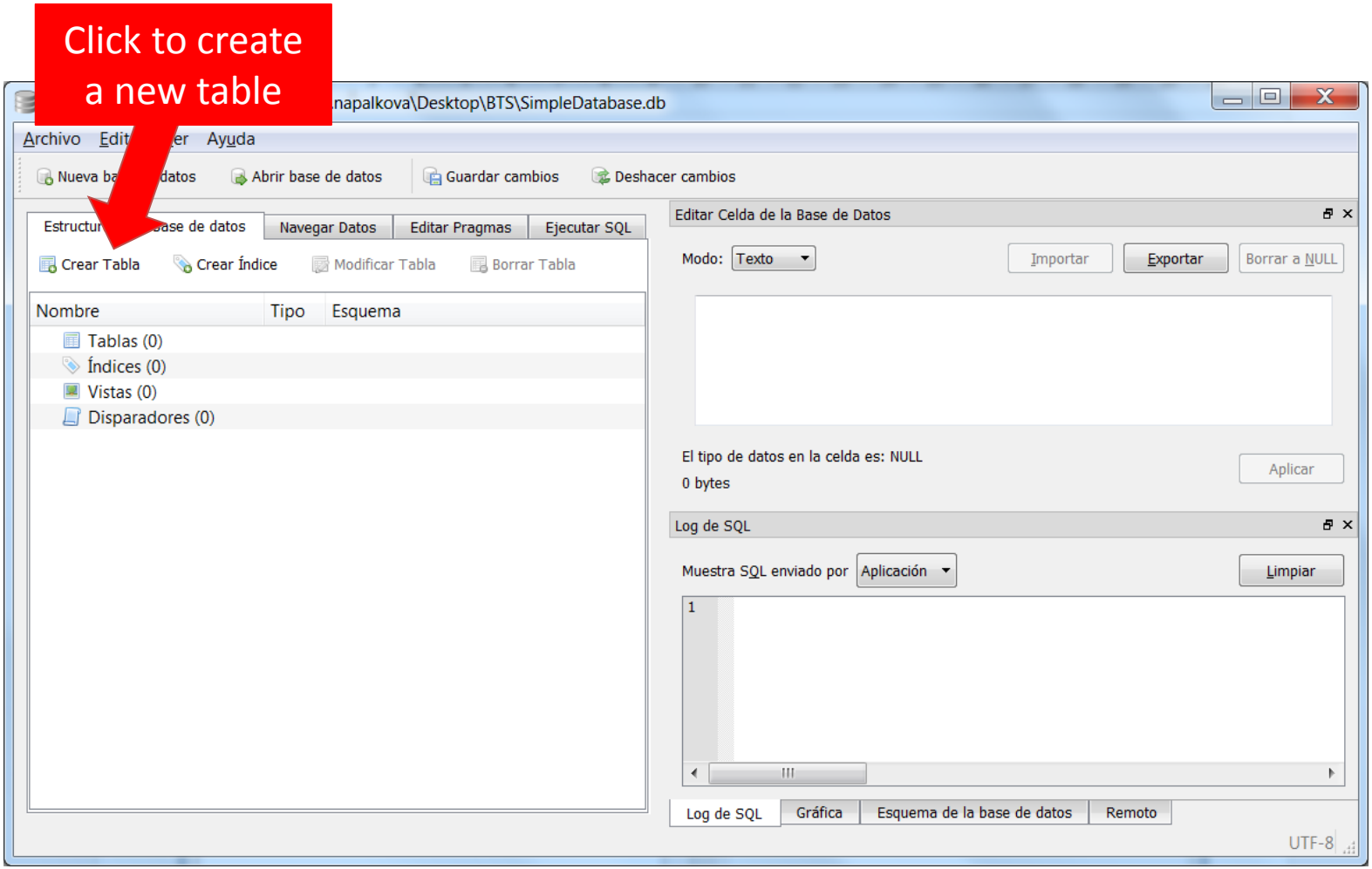
1. Open SQLite

2. Create a new database and save it in “Session_3” in your local folder “DataScienceFoundations”

3. Create a new table called “students”:

SQLite

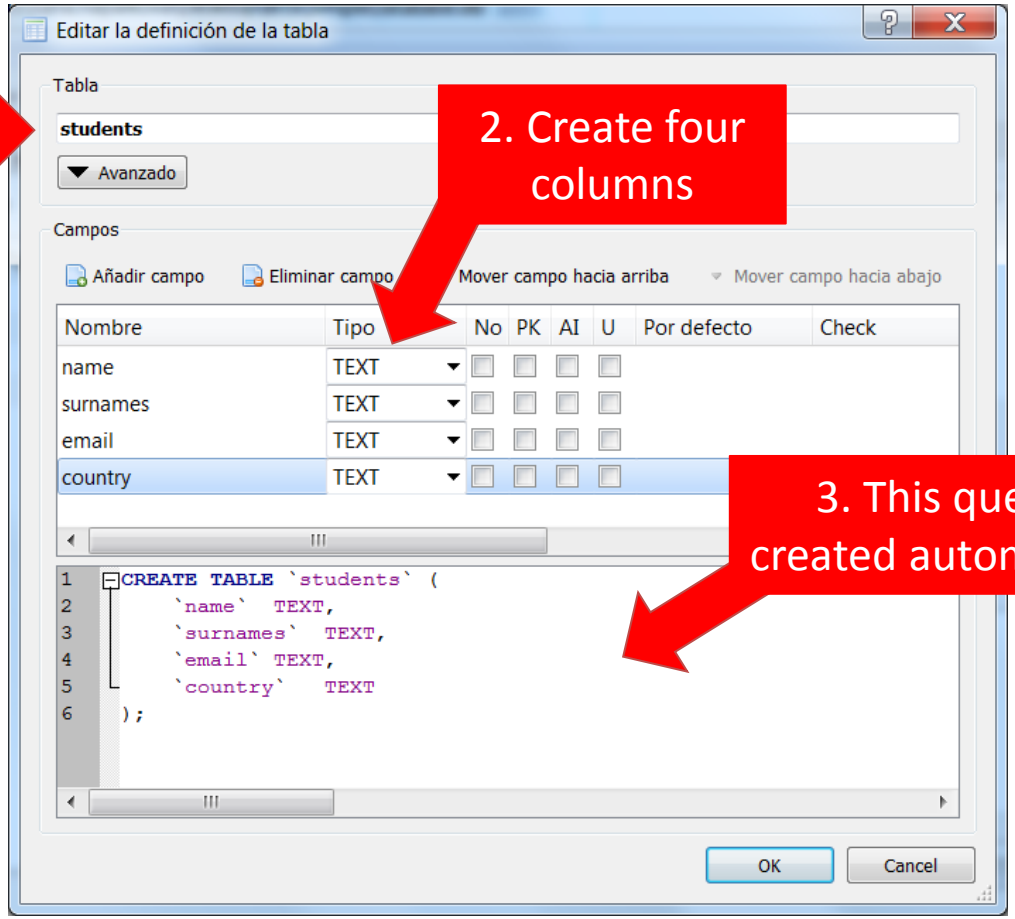
Start simple



SQLite

Start simple

1. Set the name of a table as “students”

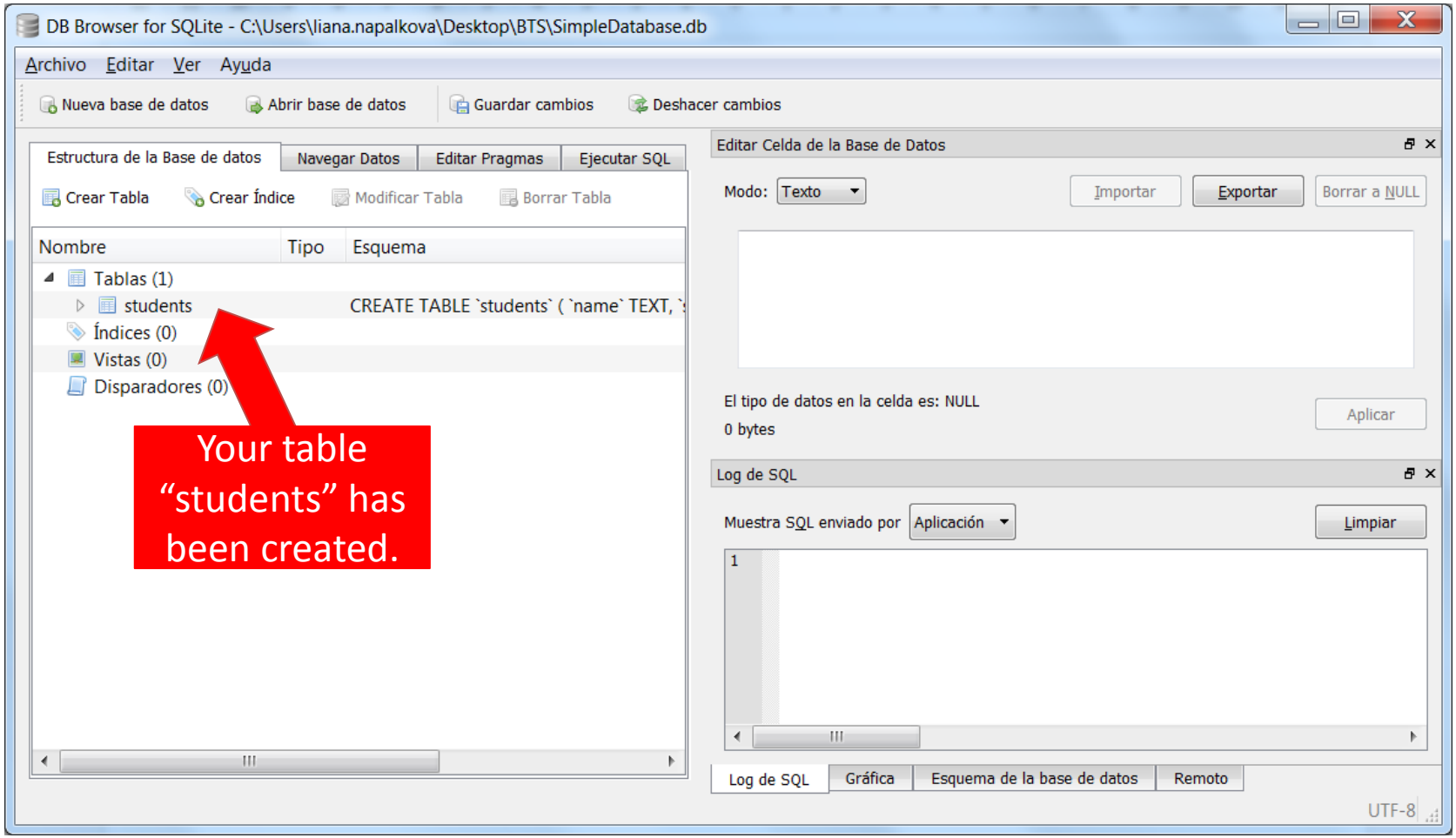


2. Create four columns

3. This query is created automatically

SQLite

Start simple



SQLite

Start simple

Now we will add some data to the table “students”.

name	surnames	email	country
John	Berry	jb@uni.edu	UK
Bianca	Mendez	bm@uni.edu	Venezuela
Bruno	Fiadino	bf@uni.edu	Italy

SQLite

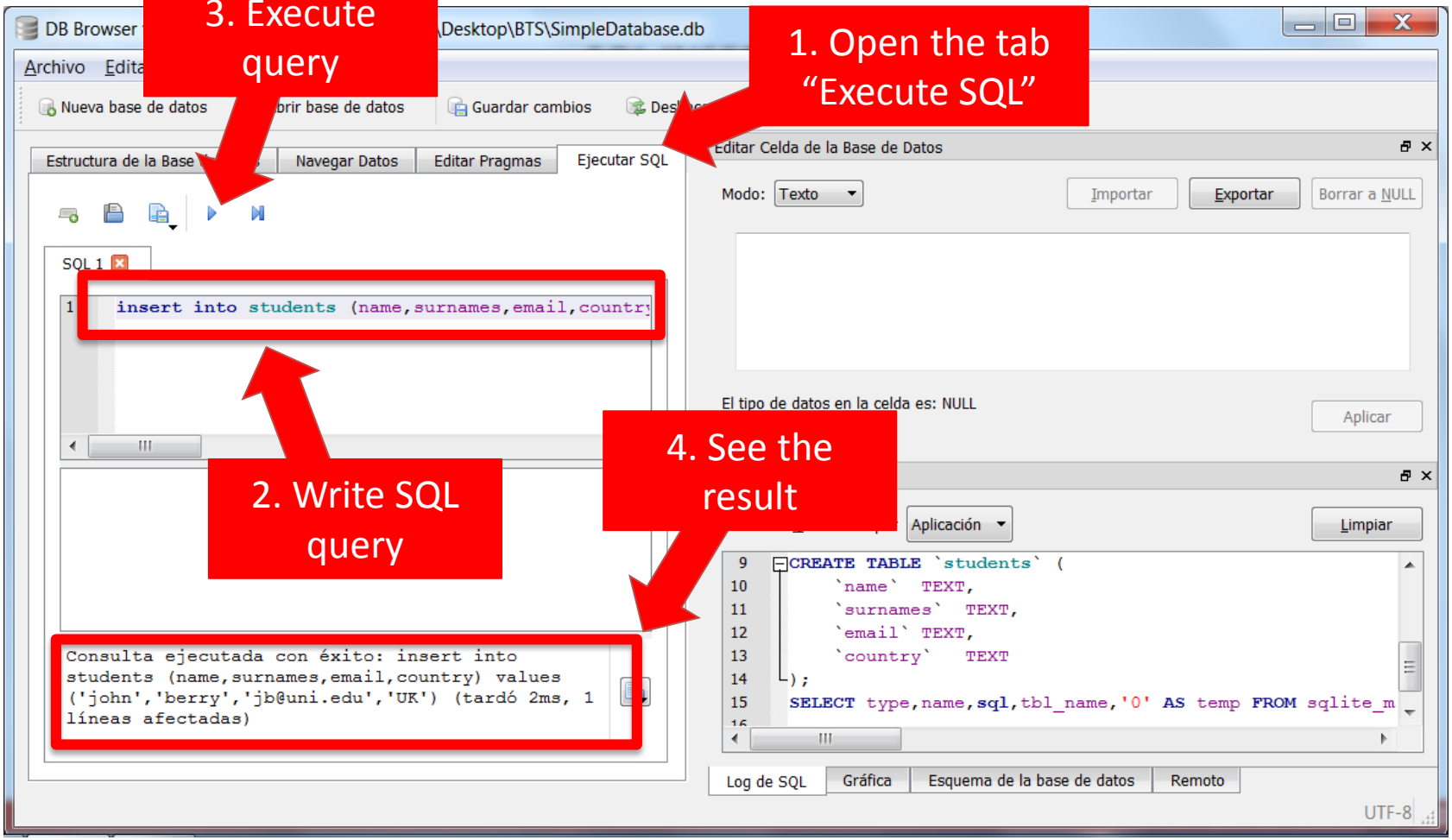
Start simple: SQL INSERT

The INSERT statement allows adding a new row into a table:

```
INSERT INTO students (name, surnames, email, country) VALUES ('john', 'berry', 'jb@uni.edu', 'UK')
```

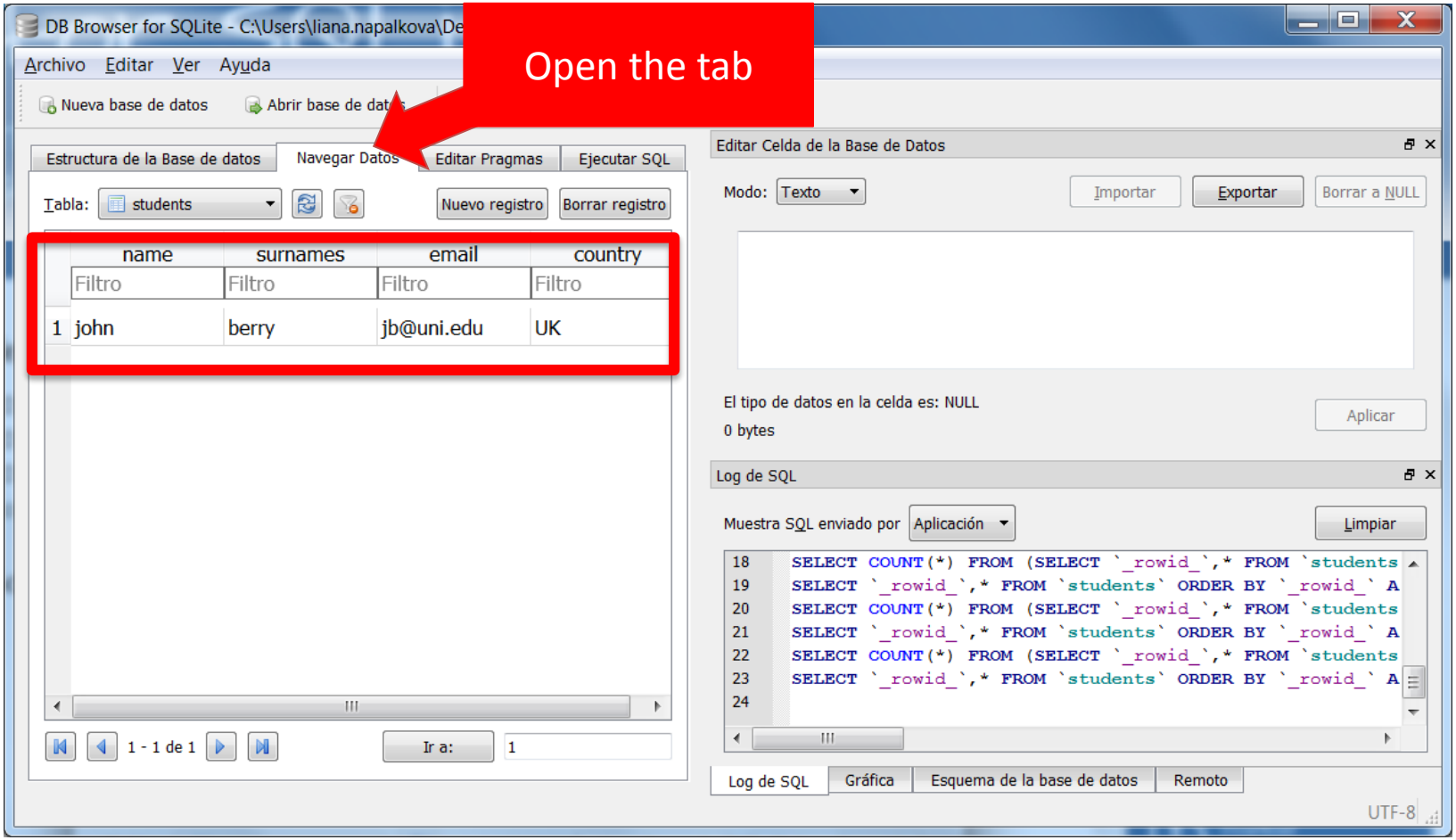
SQLite

Start simple: SQL INSERT



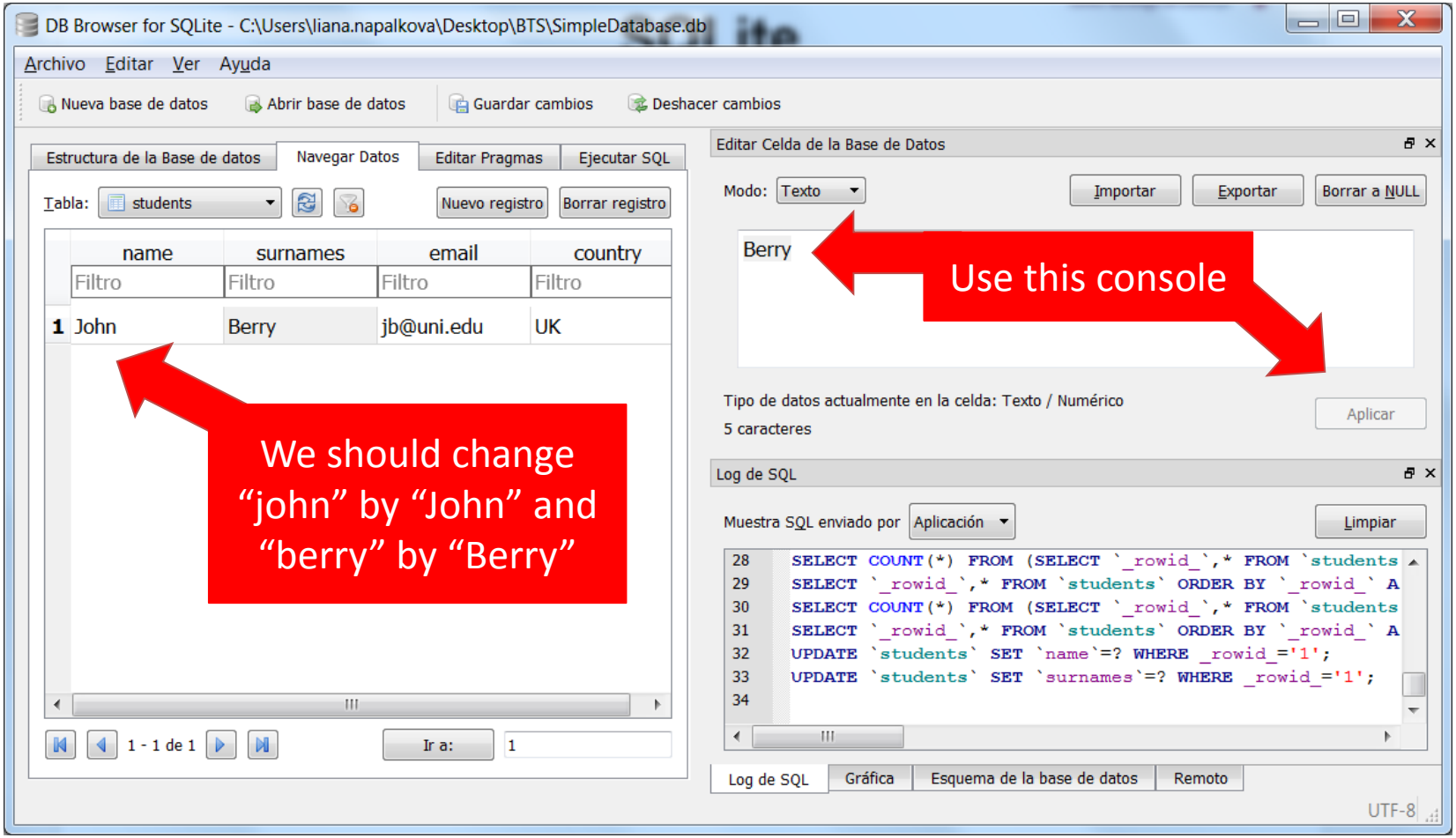
SQLite

Start simple: SQL INSERT



SQLite

Start simple: SQL INSERT



SQLite

Start simple: SQL INSERT

Please add the other two rows into the table “students”.

name	surnames	email	country
John	Berry	jb@uni.edu	UK
Bianca	Mendez	bm@uni.edu	Venezuela
Bruno	Fiadino	bf@uni.edu	Italy

```
INSERT INTO students (name, surnames, email, country)
VALUES (write the values of columns here)
```

SQLite

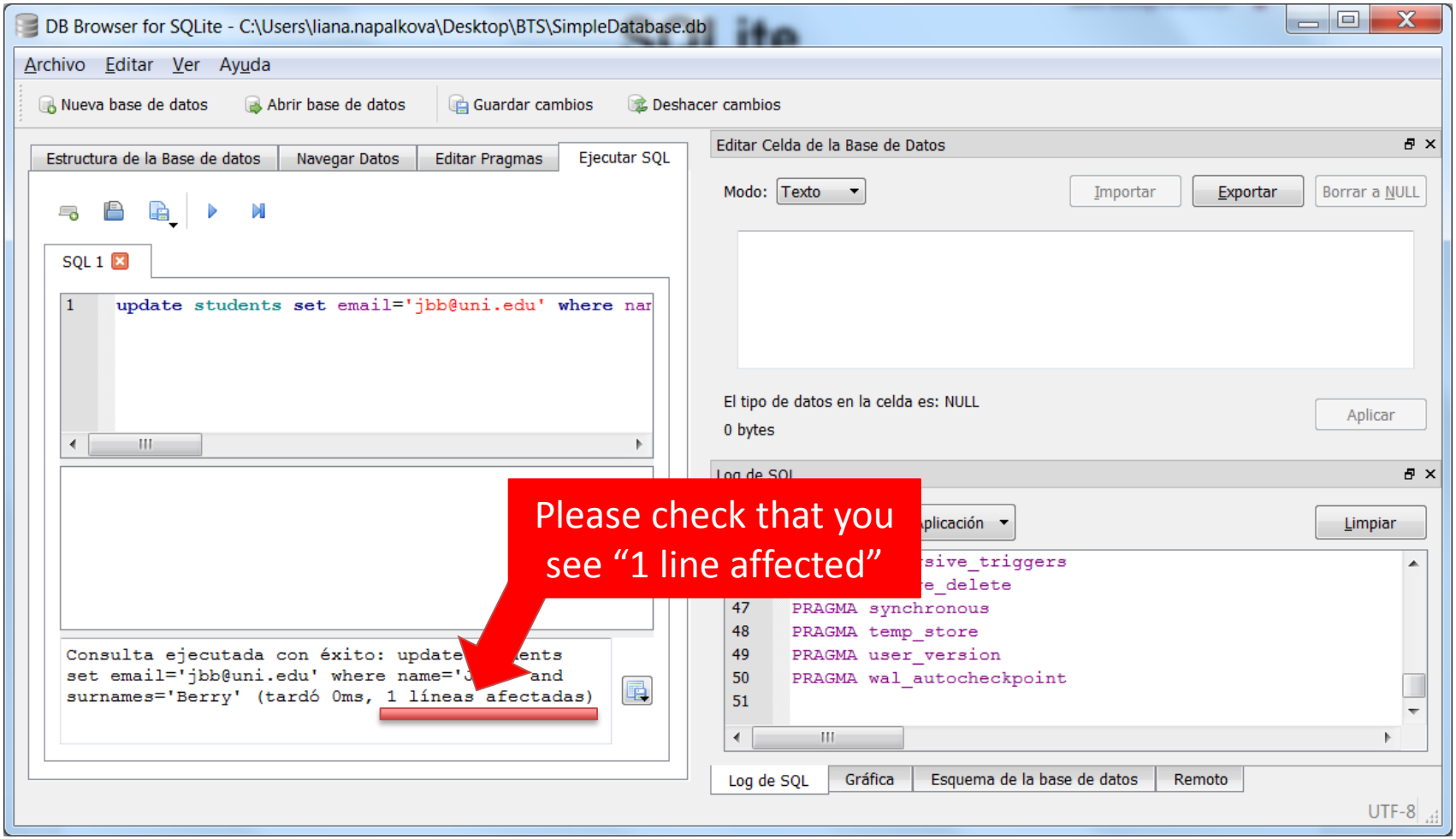
Start simple: SQL UPDATE

The UPDATE statement allows modifying a field with a where clause:

```
UPDATE students SET email='jbb@uni.edu' WHERE name='John' AND surnames='Berry'
```

SQLite

Start simple: SQL UPDATE



SQLite

Start simple: SQL SELECT

- The SELECT statement retrieves a group of records - you can either retrieve all the records or a subset of the records with a WHERE clause

- Please execute the following SELECT statements one by one and say which results do you see:

✓ `SELECT * FROM students`

✓ `SELECT name, surnames FROM students`

✓ `SELECT * FROM students WHERE country='UK'`

SQLite

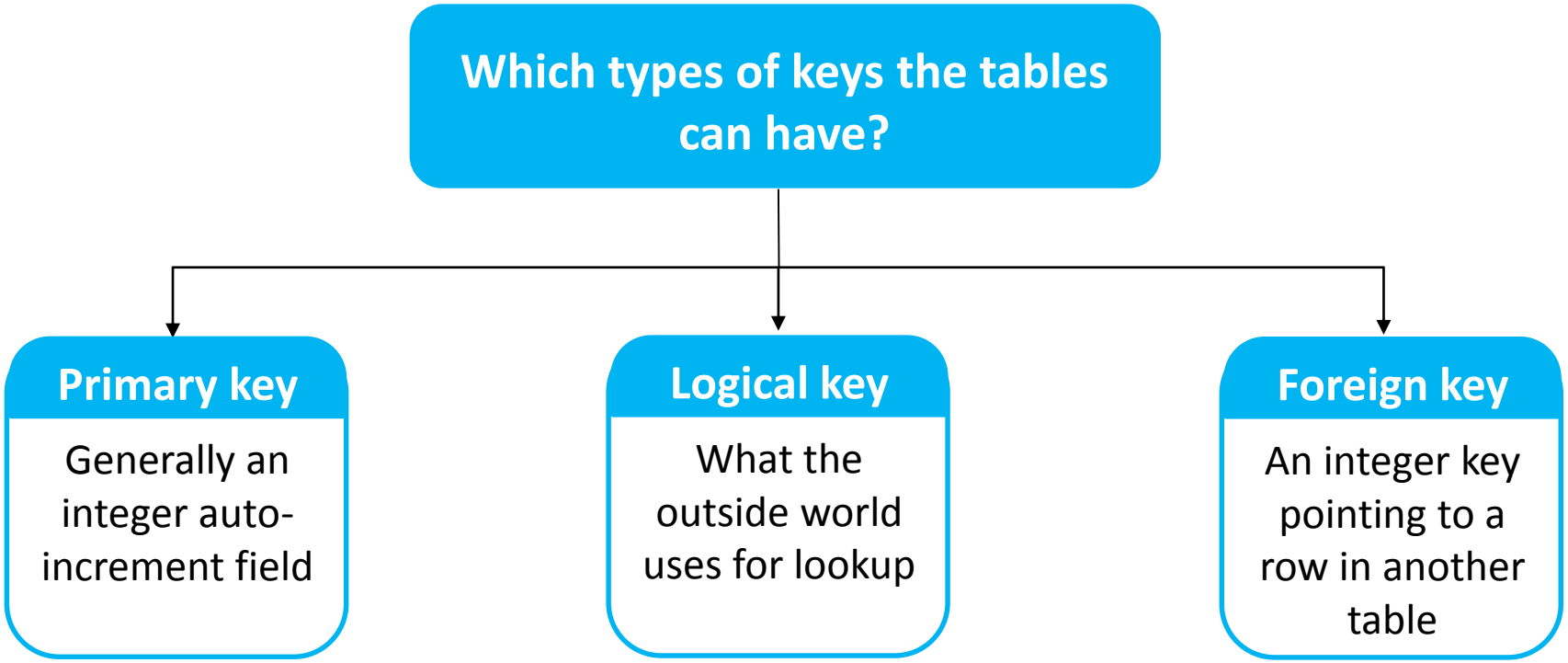
Start simple: SQL DELETE

The DELETE statement eliminates a row in a table based on a selection criteria:

```
DELETE FROM students WHERE name='John'
```

SQLite

Keys


Which types of keys the tables can have?
Primary key
Generally an integer auto-increment field
Logical key
What the outside world uses for lookup
Foreign key
An integer key pointing to a row in another table

SQLite

Keys

Database “university”

students

s_pk	name	surnames	email	country	c_pk
1	John	Berry	jb@uni.edu	UK	1
2	Bianca	Mendez	bm@uni.edu	Venezuela	1
3	Bruno	Fiadino	bf@uni.edu	Italy	2

courses

c_pk	title
1	Course #1
2	Course #2

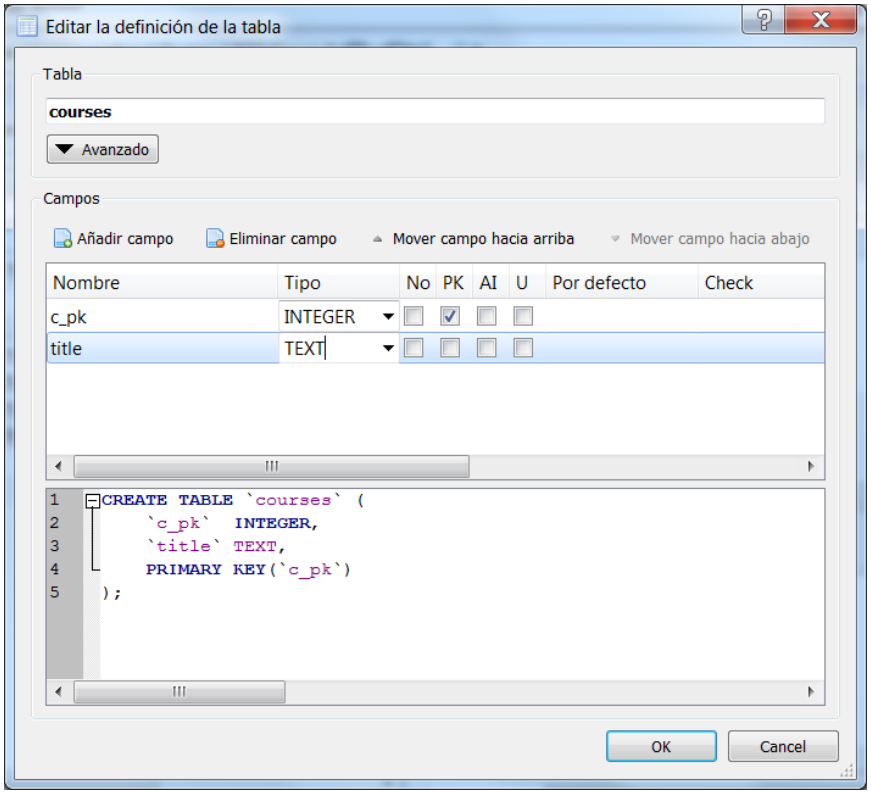
SQLite

Keys

Let's first create a new table "courses":

c_pk	title
1	Course #1
2	Course #2

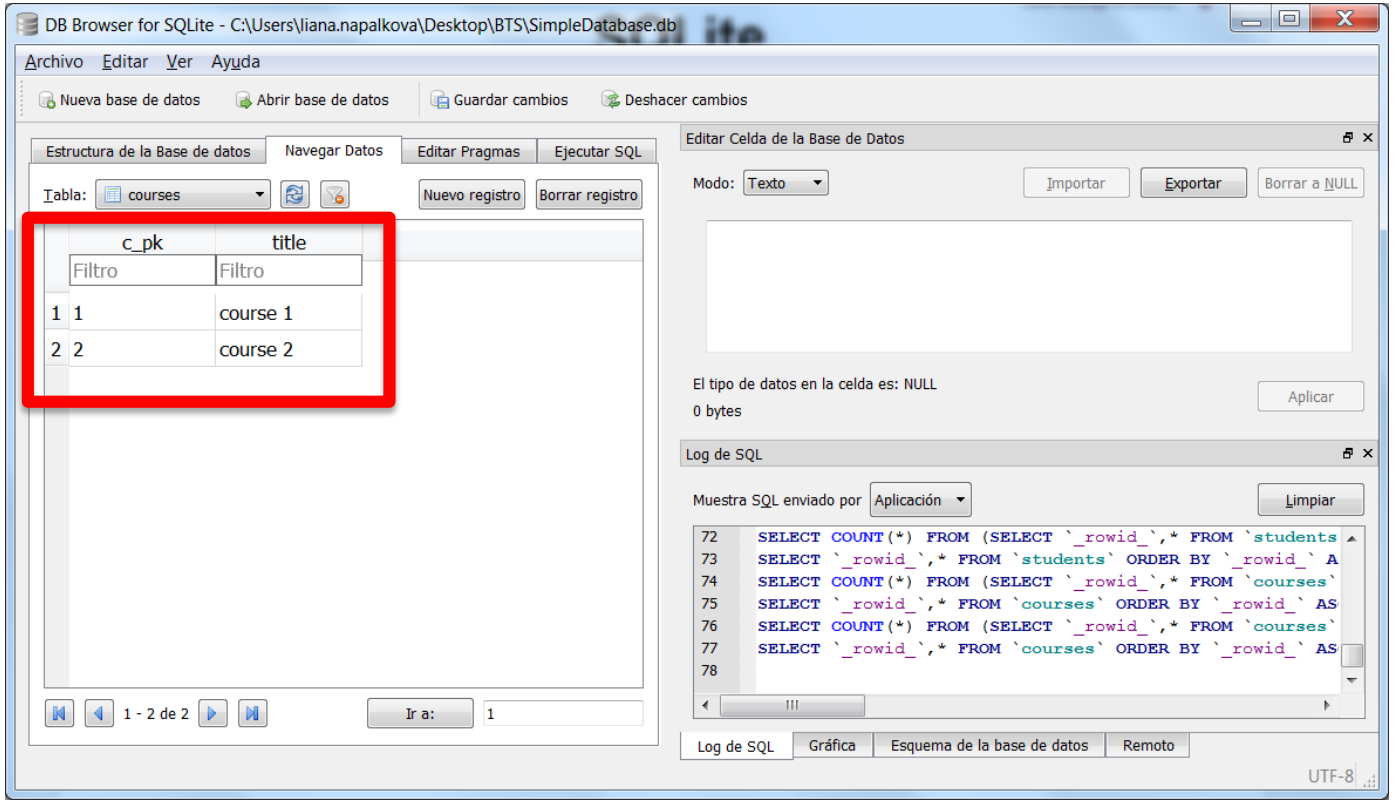
Please notice that "c_pk" is marked as a primary key (PK).



SQLite

Keys

If everything is correct, you will see two rows in the table “courses”.

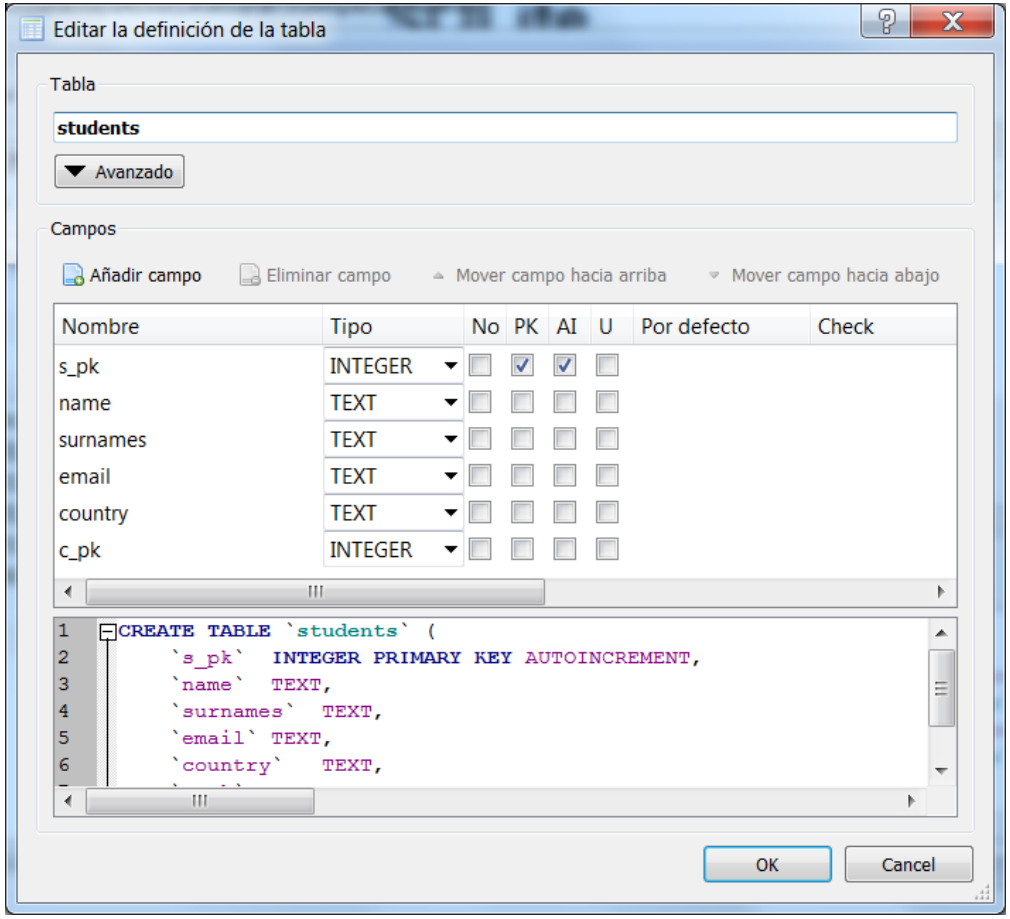


SQLite

Keys

Now modify the table “students” in order to add an auto-incremented primary key “s_pk” and a new field “c_pk”.

The field “c_pk” will be used as a foreign key to link the table “students” with the table “courses”.



SQLite

Keys

Please modify the values of the foreign key “c_pk” in the table “students” so that it looks as follows:

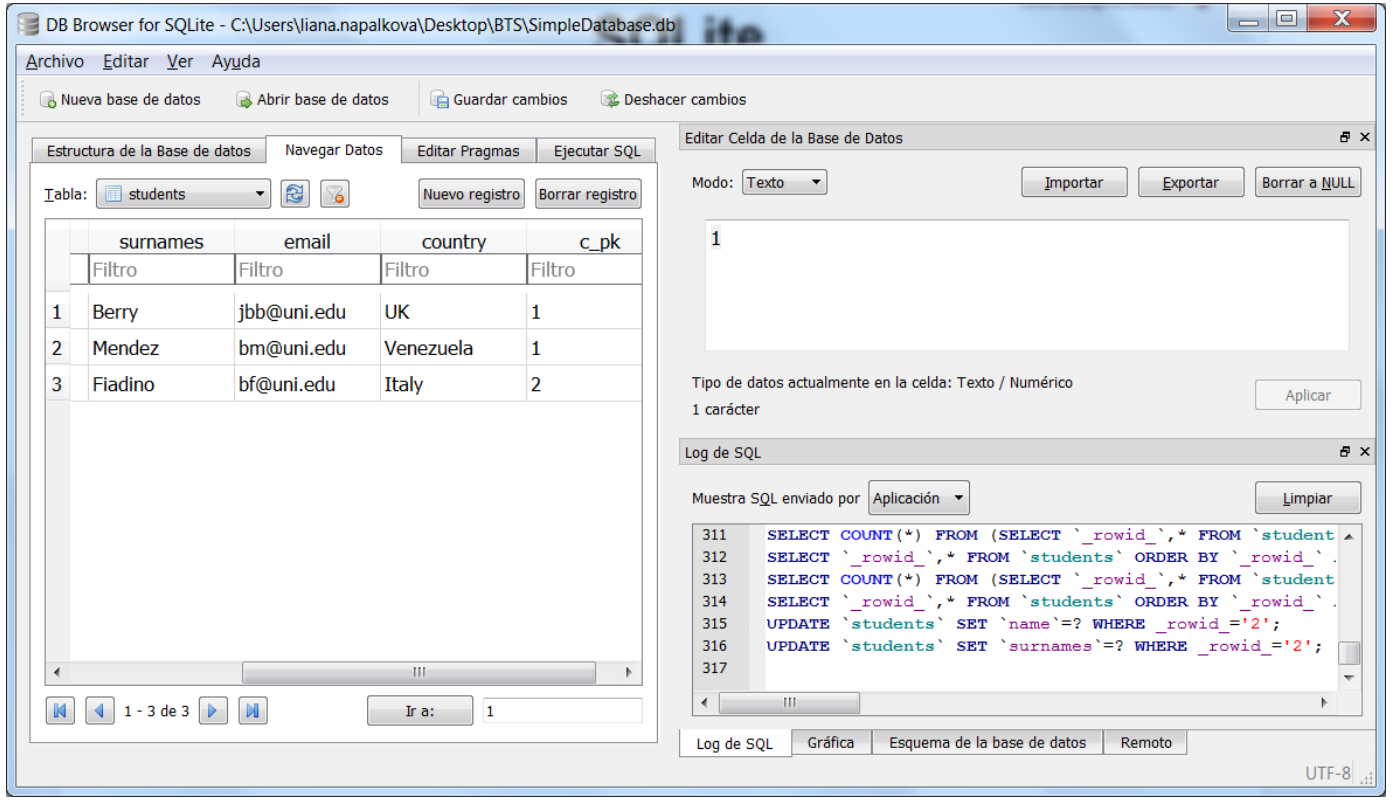
students

s_pk	name	surnames	email	country	c_pk
1	John	Berry	jb@uni.edu	UK	1
2	Bianca	Mendez	bm@uni.edu	Venezuela	1
3	Bruno	Fiadino	bf@uni.edu	Italy	2

SQLite

Keys

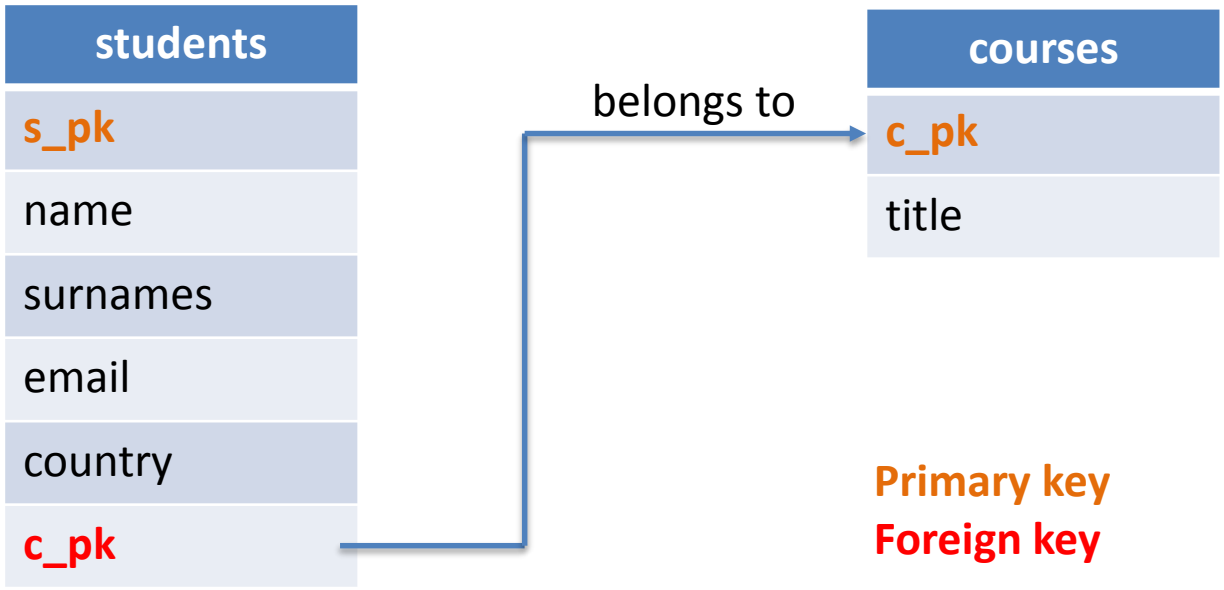
If everything is correct, your table “students” should now look like this:



SQLite

Relationship building

- We have just created the following relationships between the tables “students” and “courses”:

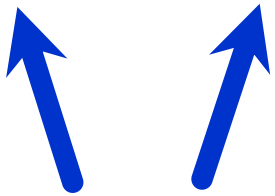


SQLite

SQL JOIN

- The JOIN operation links across several tables as part of a select operation.
- You must tell the **JOIN** how to use the keys that make the connection between the tables using an **ON clause**.

```
SELECT students.name, courses.title FROM students JOIN courses ON students.c_pk = courses.c_pk
```



What we want to see

The tables that hold the data

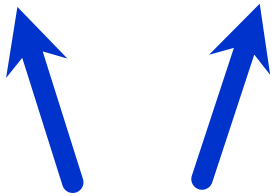
How the tables are linked

SQLite

SQL JOIN

- Please execute this statement yourself and tell the result that you see.

```
SELECT students.name, courses.title FROM students JOIN courses ON students.c_pk = courses.c_pk
```



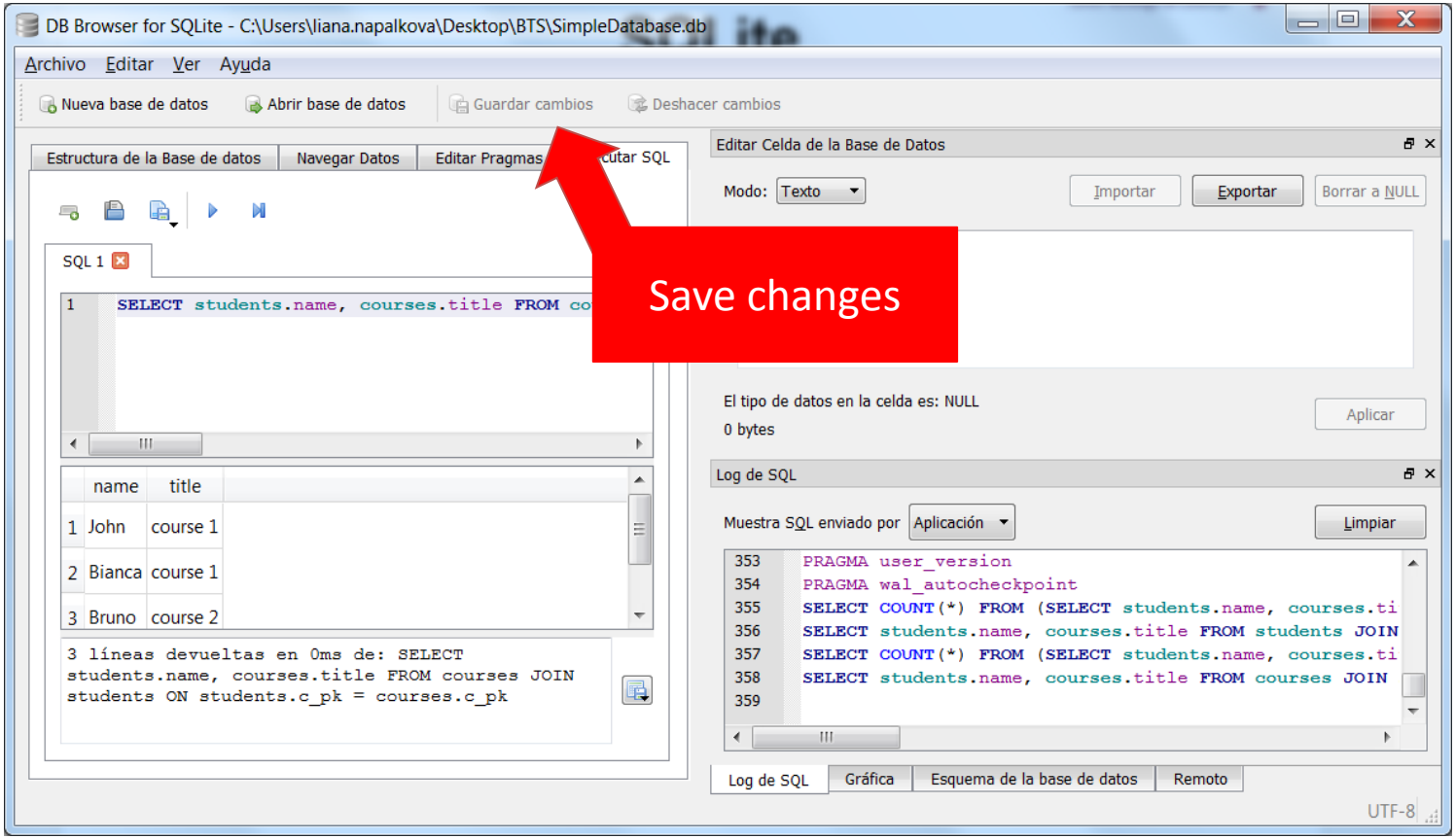
What we want to see

The tables that hold the data

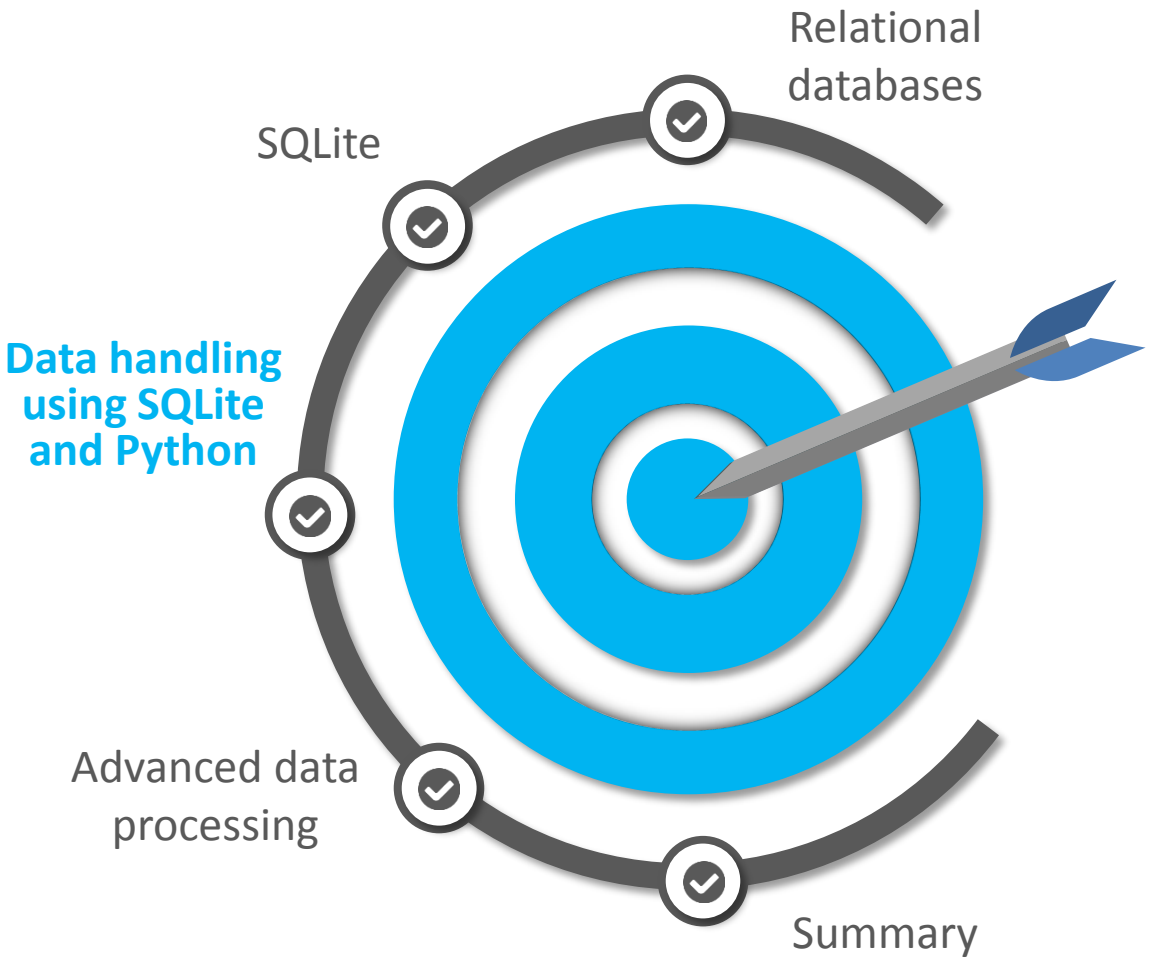
How the tables are linked

SQLite

- Click on the button “Save” in order to save your database and tables.



Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

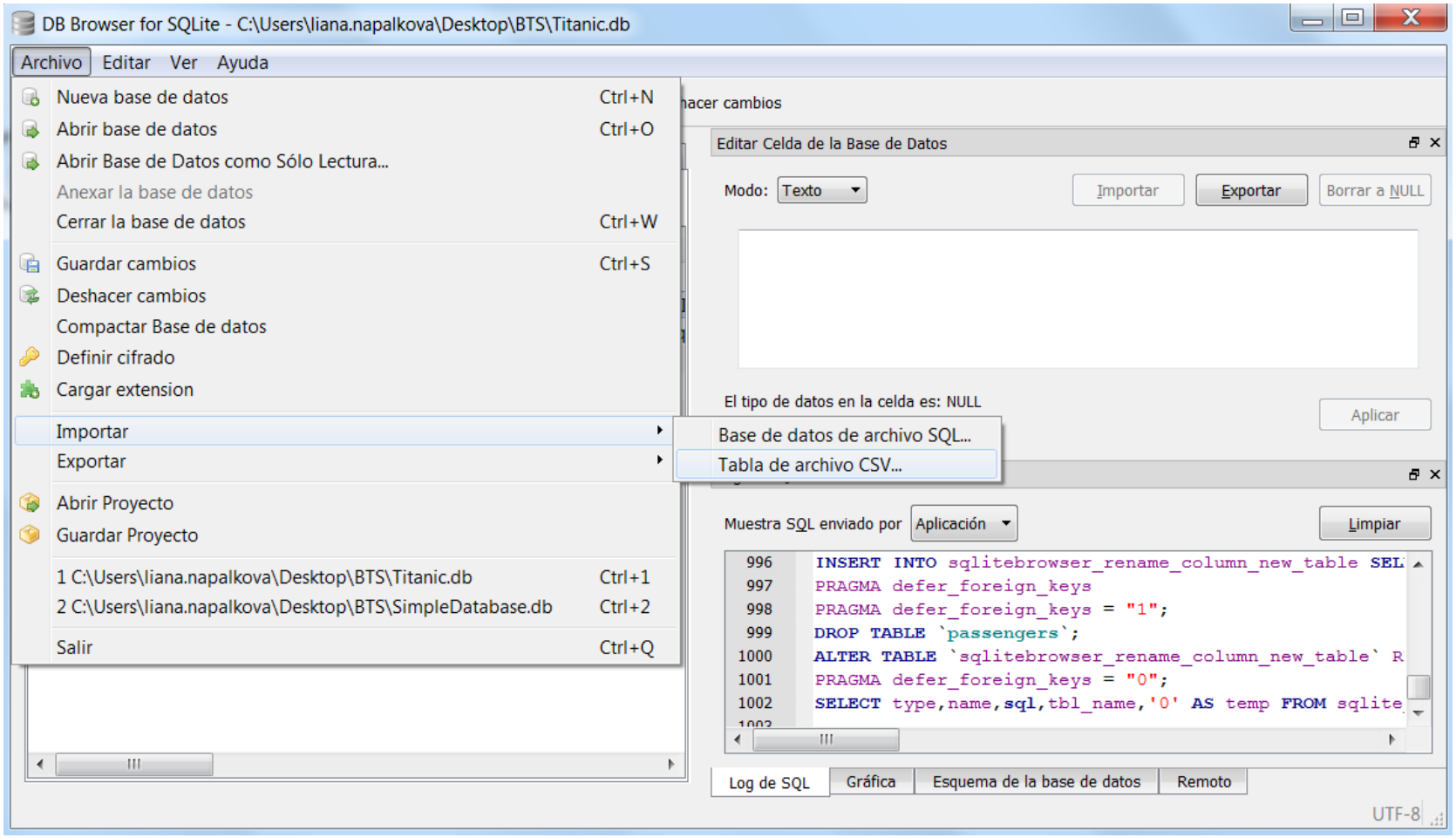
Data handling using SQLite and Python

Loading data into SQLite database

- Now we will learn how to load the data from CSV file into SQLite database.
- First of all, please create a new database called “Titanic” and save it in “Session_3” in your local folder “DataScienceFoundations”.
- In SQLite go to File->Import->CSV

Data handling using SQLite and Python

Loading data into SQLite database

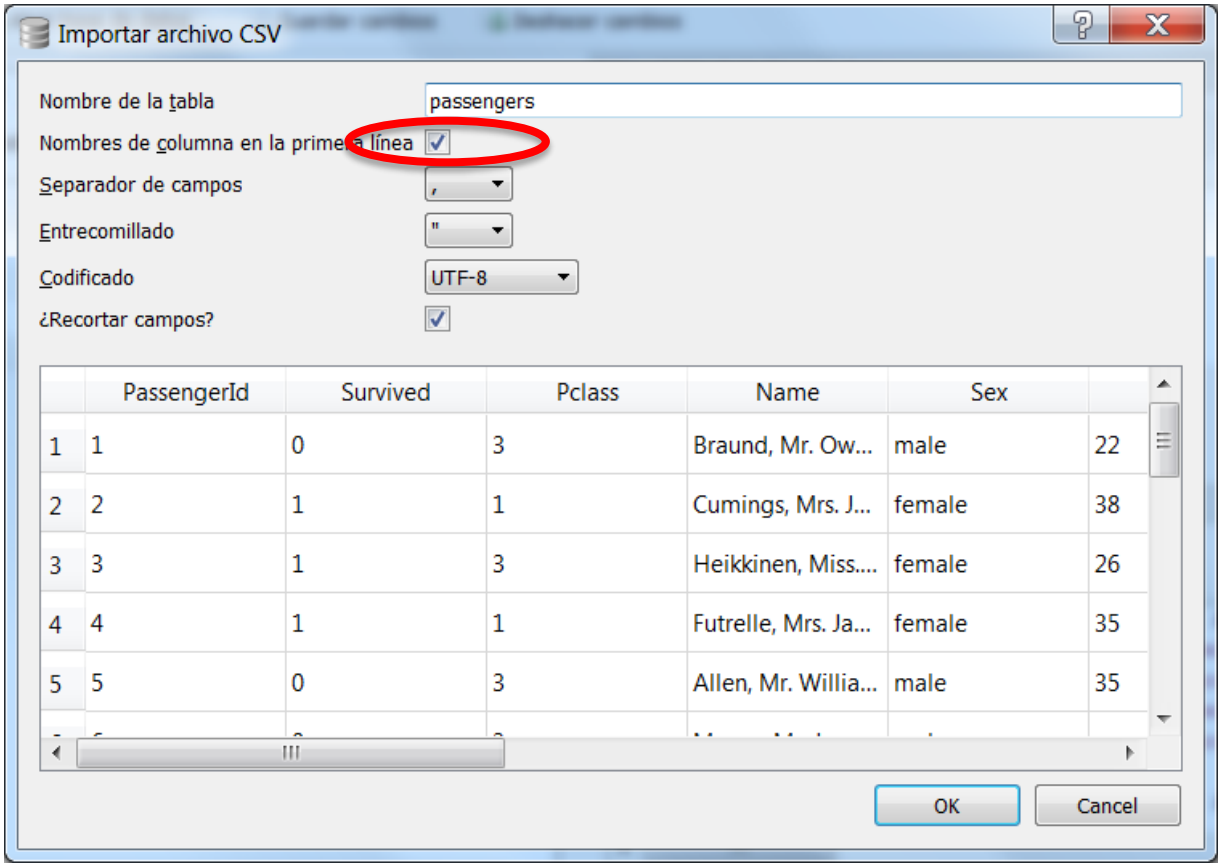


BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

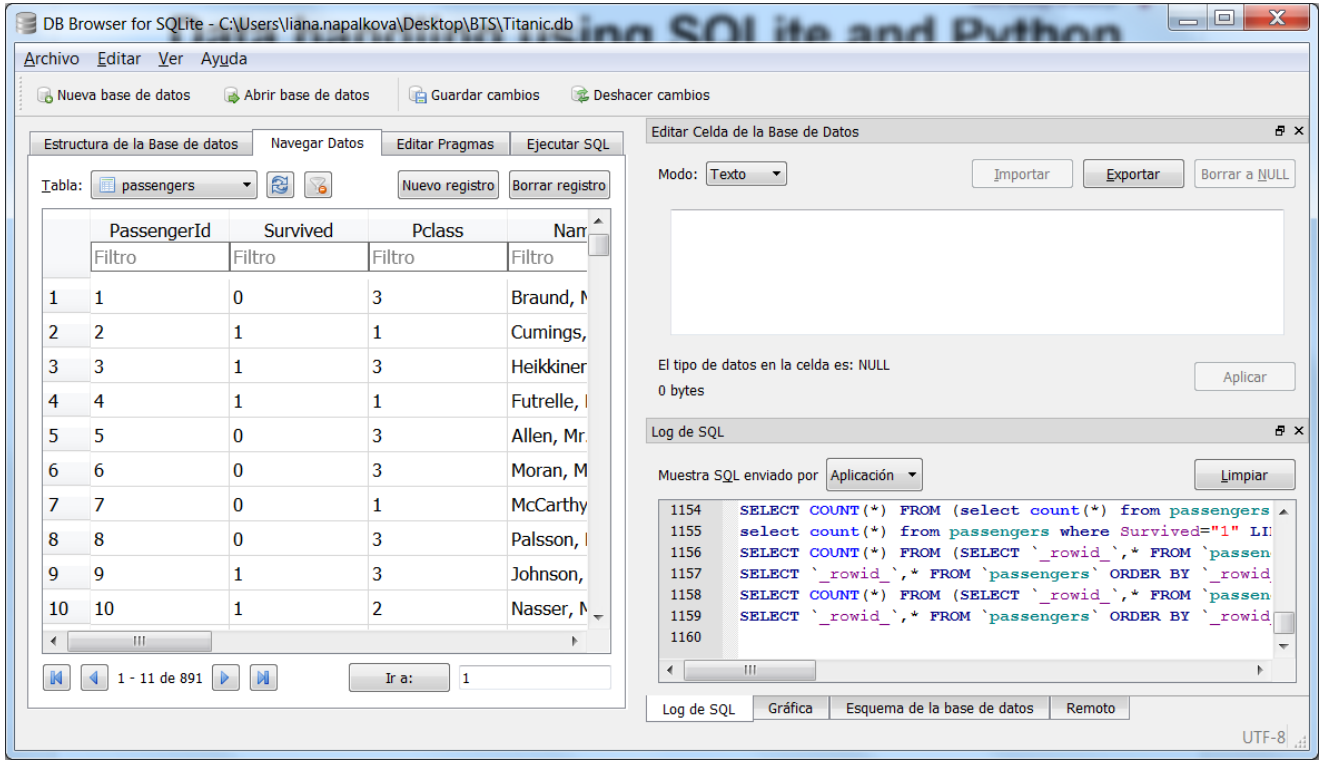
Loading data into SQLite database



Data handling using SQLite and Python

Loading data into SQLite database

- We have just created a new database called “Titanic” and loaded titanic dataset into the table “passengers”.



BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

Loading data into SQLite database

- By default all column types are TEXT.
- Please modify them as follows and save the database.

```
PassengerId   -> Integer
Survived      -> Integer
Pclass        -> Integer
Name          -> Text
Sex           -> Text
Age           -> Real
SibSp         -> Integer
Parch         -> Integer
Ticket        -> Text
Fare          -> Real
Cabin         -> Text
Embarked      -> Text
```

Data handling using SQLite and Python

Loading data into SQLite database

- Execute the query to be sure that everything is correct:

```
select count(*) from passengers

select count(*) from passengers where Survived=1

select count(*) from passengers where Survived=0
```

BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

Retrieving data from SQLite database

- Now we will retrieve the data from our “Titanic” database using Python and Pandas.

- Synchronize your folder “BTS_MasterInBigData” with the remote repository “BTS_MasterInBigData”:

  https://github.com/LianaNapalkova/BTS_MasterInBigData.git

- Go to the folder “Session_3” and import the file “1_retrieve_data_from_SQLite_database.ipynb” into your Jupyter Notebook.

Data handling using SQLite and Python

Retrieving data from SQLite database

To retrieve the data from DB, we execute the following steps:

1. Establish a connection to the SQLite database by creating a Connection object.
2. Create a Cursor object using the cursor method of the Connection object.
3. Execute the SELECT statement.
4. Call the fetchall() method of the cursor object to fetch the data.
5. Loop the cursor and process each row individually.

BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

Retrieving data from SQLite database

- Change the `database` by the path to `Titanic.db` on your local file system:

```
database = "C:\\Users\liana.napalkova\Desktop\BTS\Titanic.db"
```

- Execute the code “1_retrieve_data_from_SQLite_database.ipynb” in your Jupyter Notebook.

- Which results do you get?

BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

Retrieving data from SQLite database

- Now let's retrieve data from the database "Titanic" using Pandas.
- Go to the folder "Session_3" and import the file "2_retrieve_data_from_SQLite_database_pandas.ipynb" into your Jupyter Notebook.
- Change the `database` by the path to `Titanic.db` on your local file system:

```
database = "C:\\Users\liana.napalkova\Desktop\BTS\Titanic.db"
```

- Execute the code.
- Push your Jupyter Notebook and database file into the folder "Session_3" of your repository "DataScienceFoundations".

BTS | Barcelona Technology School

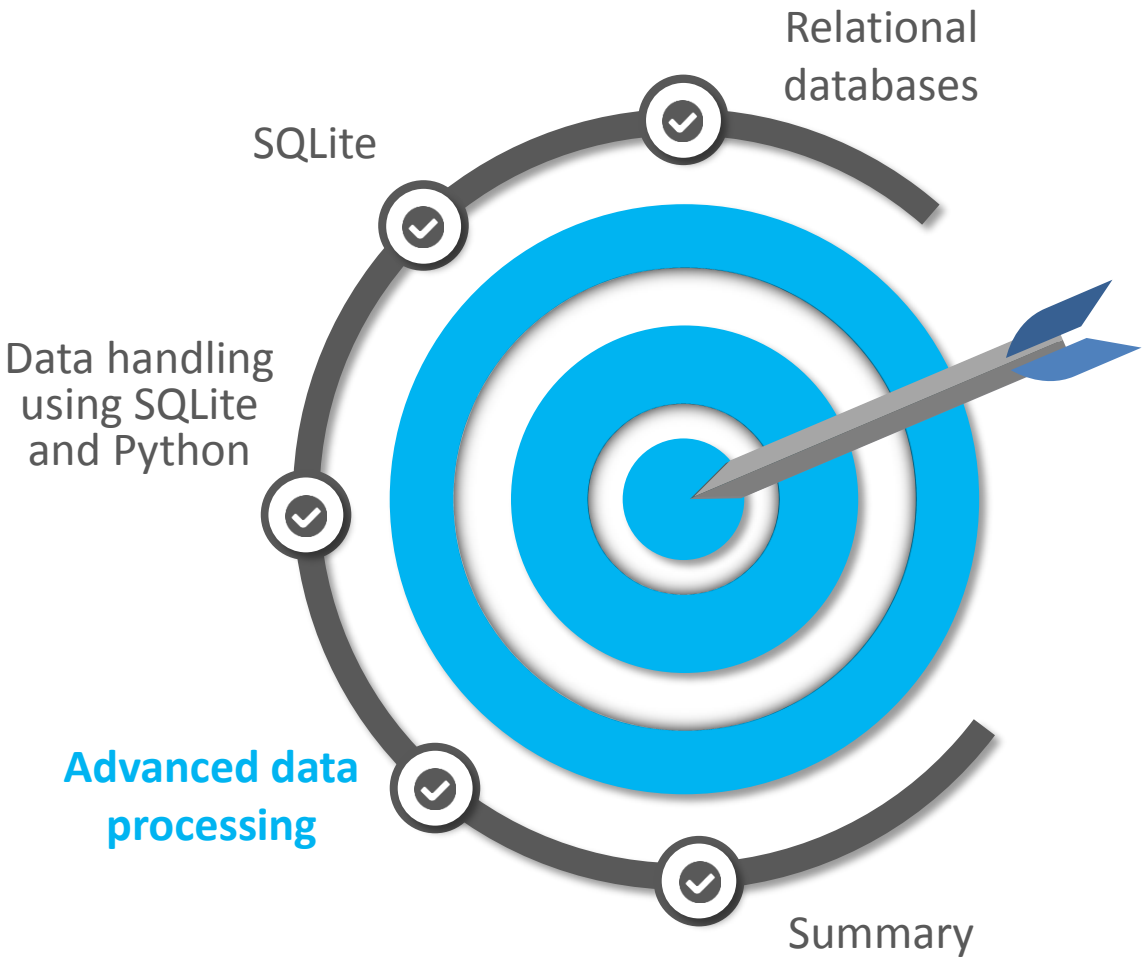
eurecat Centre Tecnològic de Catalunya

Data handling using SQLite and Python

Task

- Create a new database “BikeSharing” for storing bike sharing data (see Session 2 to find the dataset).
- Load bike sharing data into your database.
- Create a new Jupyter Notebook and write a code for retrieving data from your database into Pandas DataFrame.
- Push your Jupyter Notebook and database file into the folder “Session_3” of your repository “DataScienceFoundations”.

Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

Advanced processing of data

- Open any notebook that you used to analyze Titanic dataset.
- Execute the script in order to load the Titanic dataset into Pandas DataFrame.
- Now we will learn how to do the advanced data processing.

Advanced processing of data

- Take a look at the column “Name”.
- It has entries like “Braund, Mr. Owen Harris”, “Heikkinen, Miss. Laina”, etc.
- What can you see?

Advanced processing of data

- Take a look at the column “Name”.
- It has entries like “Braund, Mr. Owen Harris”, “Heikkinen, Miss. Laina”, etc.
- What can you see?
- We can see that this column contains information about the title of a person: Miss, Mrs, Mr, Master, and others.
- How can we create a new column that would contain these values?

Advanced processing of data

```
import re

# Define function to extract titles from passenger names
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    # If the title exists, extract and return it.
    if title_search:
        return title_search.group(1)
    return ""

# Create a new feature Title
df['Title'] = df['Name'].apply(get_title)
```

Which result do you get by running this code?:

```
df['Title'].unique()
```

Advanced processing of data

- The replace function allows substituting values by another values.
- For example, it is possible to substitute a set of values by a single value as follows:

```
# Replace all rare titles by Rare
df['Title']   =   df['Title'].replace(['Lady','Countess','Capt',
'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'],
'Rare')
```

- It is also possible to substitute one value by another value:

```
# Make other replacements
df['Title'] = df['Title'].replace('Mlle', 'Miss')
df['Title'] = df['Title'].replace('Ms', 'Miss')
df['Title'] = df['Title'].replace('Mme', 'Mrs')
```

Advanced processing of data

- Another mapping approach is to use the **loc** function as follows:

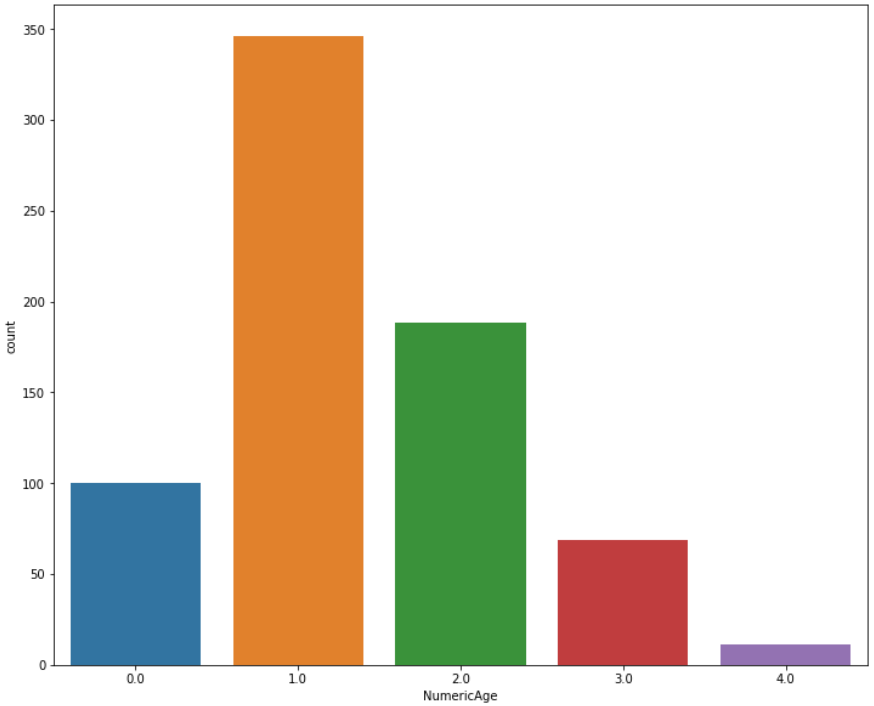
```
df.loc[df['Age'] <= 16, 'NumericAge'] = 0
df.loc[(df['Age'] > 16) & (df['Age'] <= 32), ' NumericAge'] = 1
```

- Please finish the mapping for all age ranges.

- Please create a counterplot for “NumericAge”.

Advanced processing of data

```
df.loc[df['Age'] <= 16, 'NumericAge'] = 0
df.loc[(df['Age'] > 16) & (df['Age'] <= 32), ' NumericAge'] = 1
df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'NumericAge'] = 2
df.loc[(df['Age'] > 48) & (df['Age'] <= 64), 'NumericAge'] = 3
df.loc[df['Age'] > 64, 'NumericAge'] = 4
```



Advanced processing of data

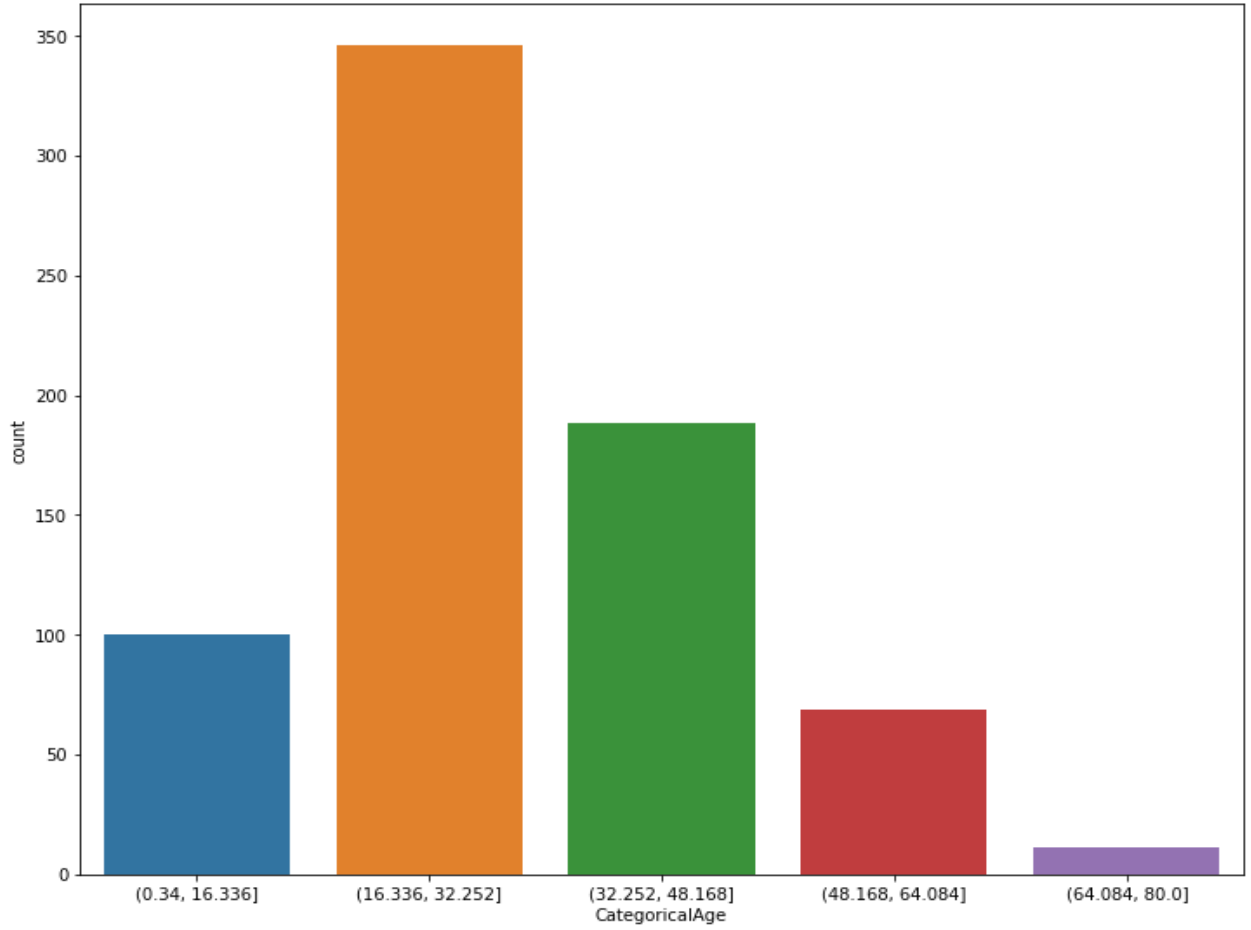
- There is another way to convert a numerical variable into a **categorical variable**.
- Run this code:

```
df['CategoricalAge'] = pd.cut(df['Age'], 5)
```

- Create a counterplot for “CategoricalAge”.
- Which age group is the most frequent among the passengers of Titanic?

Advanced processing of data

```
plt.figure(figsize=[12,10])
sns.countplot(df['CategoricalAge'])
plt.show()
```



Advanced processing of data

- We can use the function “apply” in order to run a user-defined function row-wise or column-wise.

- For example, let’s create a new feature “Person” that has the following unique values: “female”, “male”, “child”.

```
def get_person(passenger):
    age,sex = passenger
    return 'child' if age < 16 else sex

df['Person'] = df[['Age','Sex']].apply(get_person,axis=1)
```

Advanced processing of data

- Let's imagine that you should create a prediction model (a simple regression model) for predicting "Survived" based on features of a passenger.

- To be able to create the prediction model, all your data should be pre-processed. For example, to apply the regression model, all non-numeric values should be converted into numeric values:

- You can do it using **mapping** - the function "map" allows mapping values to another values, for example:

```
df['Sex'] = df['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
df['Title'] = df['Title'].map(title_mapping)
df['Title'] = df['Title'].fillna(0)
```

Advanced processing of data

- Finally, we will learn the function **groupby**.
- It is used for grouping the data by column names.
- For example, let's group the data by "Person" and estimate the average value of "Survived" for each group.

```
df[["Person", "Survived"]].groupby(['Person'],as_index=False).mean()
```

	Person	Survived
0	child	0.590361
1	female	0.756458
2	male	0.163873

- Create a barplot to visualize this data.

Advanced processing of data

- What else can you do using groupby?
- You can group by multiple columns and apply multiple aggregations, e.g. “sum”, “mean”, “min”, “max”.

```
df.groupby(df.index_col).agg({'Age':'mean',
                              'Survived':'sum'})[['Age','Survived']]
                        .reset_index()
```

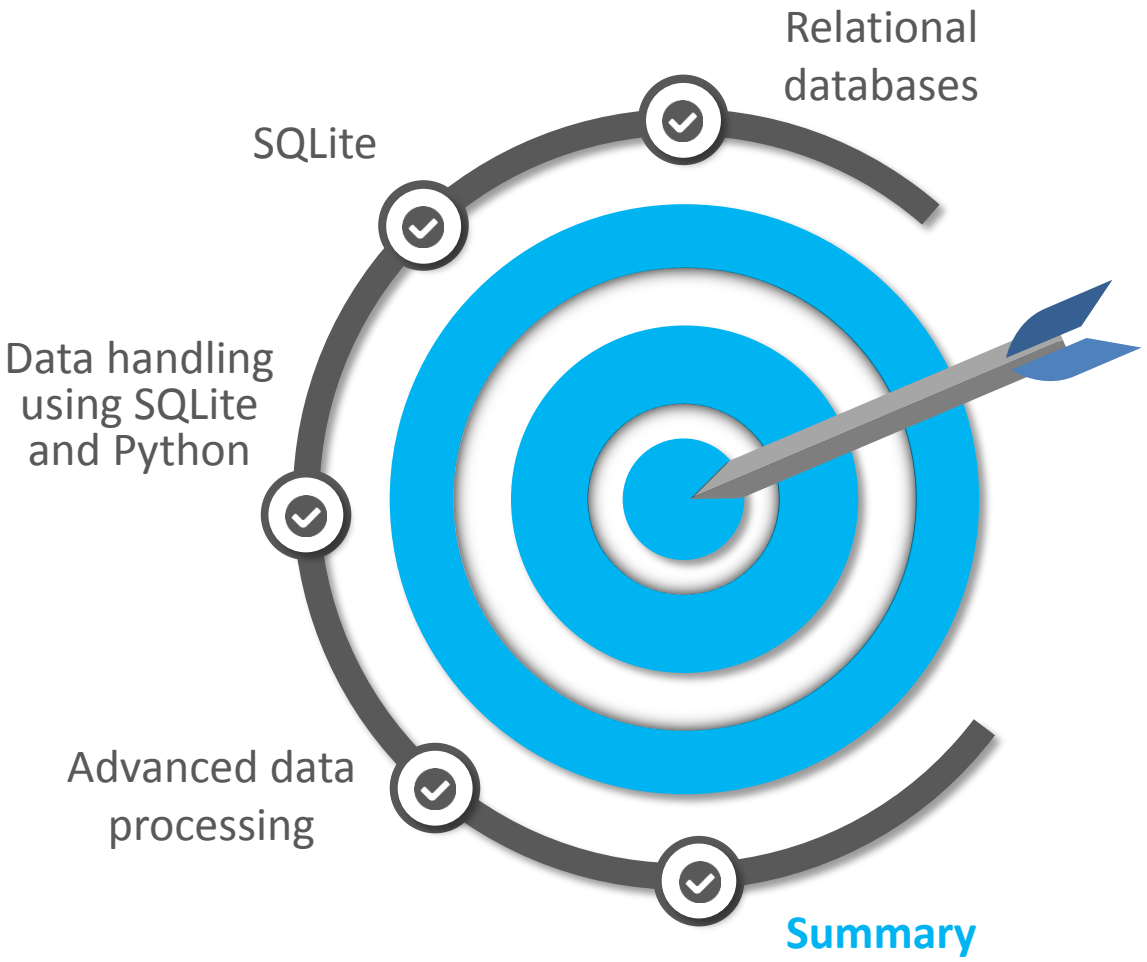
	Person	Embarked	Age	Survived
0	child	C	8.245556	14
1	child	Q	7.200000	1
2	child	S	5.737500	34
3	female	C	33.750000	53
4	female	Q	25.136364	26
5	female	S	31.716561	124
6	male	C	34.960938	26
7	male	Q	39.500000	3
8	male	S	32.608309	59

Advanced processing of data

Task

- Ask 5 more complex questions to the Titanic dataset and use the functions **apply**, **groupby**, **replace**, **loc**, **map**.
- For example, you may create new features like we did with “Person” or “Title”. This process is called **feature engineering**.
- Create 5 charts that show your findings.
- Push your Jupyter Notebook into the folder “Session_3” of your repository “DataScienceFoundations”.

Contents



Today's objective

- How to work with relational database - SQLite.
- How to perform advanced processing of data.

Summary

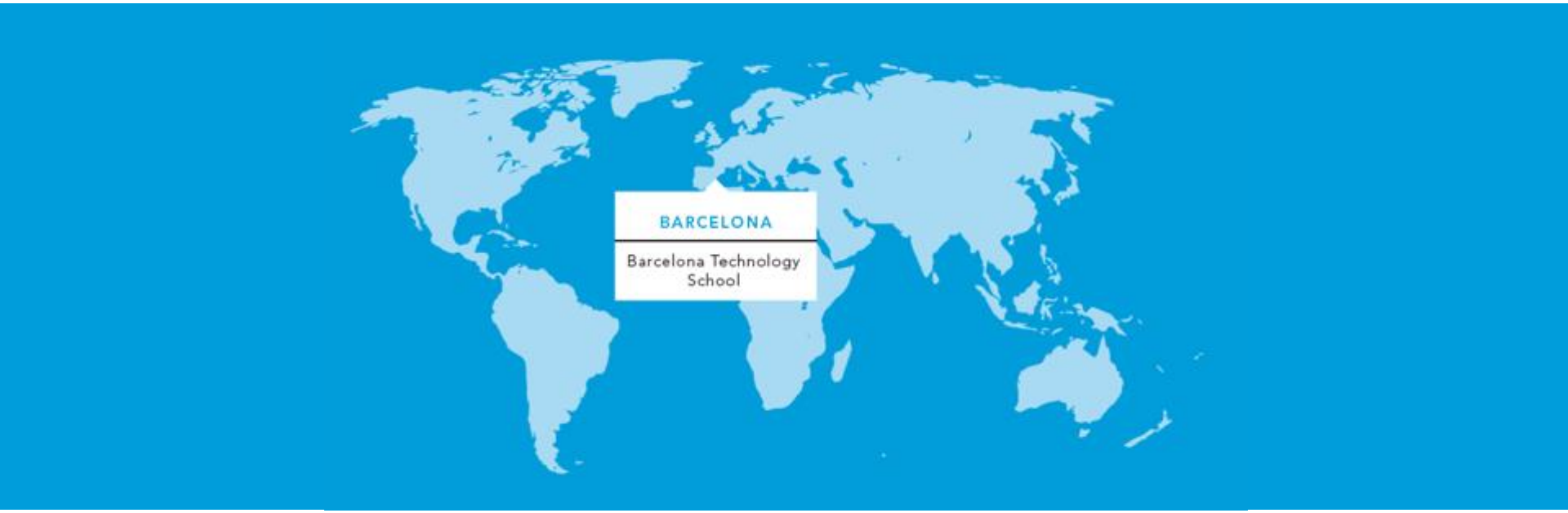
- Why do we need to **clean** the data before performing any analysis?
- Why handling of **empty fields** is important?
- Why asking **clear questions** to the data is important?
- Why adding **docstrings** into your code is important?
- Why a **proper visualization** should be selected to communicate the retrieved insights to a client/general public?
- What is feature engineering and why do we need it?
- Why do we need to convert all non-numerical variables into numerical variables before applying **the prediction model, like regression model**? How this conversion can be done?

BTS | Barcelona Technology School

eurecat Centre Tecnològic de Catalunya

Individual assignment (graded)

- Select any dataset from Kaggle, UCI repository (https://archive.ics.uci.edu/ml), or any City Council open data source, etc.
- For this assignment the data should be in CSV format and should not exceed 10 Mb.
- Load the data into SQLite DB (check that data types are correct).
- Retrieve the data from Python using Pandas.
- Ask and answer 20 questions to your dataset.
- Create visualizations to communicate your major findings (at least 5 visualizations).
- Put comments and docstrings to your code.
- Push your Jupyter Notebook (in *.ipynb format) and the database file (in *.db format) into the folder "Session_3_graded_assignment" of your repository "DataScienceFoundations."


BARCELONA
Barcelona Technology School