



# MODULE 06 - Piscine Python for Data Science

SQL and Pandas

*Summary: This day will help you acquire skills with SQL.*

# Contents

<b>I</b>	<b>Foreword</b>	<b>2</b>
<b>II</b>	<b>Instructions</b>	<b>3</b>
<b>III</b>	<b>Specific instructions of the day</b>	<b>4</b>
<b>IV</b>	<b>Exercice 00 : Select</b>	<b>5</b>
<b>V</b>	<b>Exercice 01 : Subquery</b>	<b>7</b>
<b>VI</b>	<b>Exercice 02 : Preprocessing</b>	<b>8</b>
<b>VII</b>	<b>Exercice 03 : Selects and aggregations</b>	<b>10</b>
<b>VIII</b>	<b>Exercice 04 : Enrichment and transformations</b>	<b>12</b>

# Chapter I

## Foreword

Fun facts about pandas:

- Sequel or Ess Que Ell? How to pronounce SQL correctly? Some interviewers even reject interviewees if they pronounce it not in the “right” way. It seems that it is really important to know which one is correct.
- SQL was originally spelled as SEQUEL (Structured English Query Language), but it turned out later that it was the registered trademark by an aircraft company. The authors had to change it. Since then we have SQL (Structure Query Language). One of the authors [was asked the question](#) about which way of pronunciation is correct. He answered it:

Since the language was originally named SEQUEL, many people continued to pronounce the name that way after it was shortened to SQL. Both pronunciations are widely used and recognized. As to which is more “official”, I guess the authority would be the ISO Standard, which is spelled (and presumably pronounced) S-Q-L.

Thanks for your interest,

Don Chamberlin

- So one of the authors said that you can use both ways. S-Q-L is a more official way of doing it (presumably, huh?). What do other “authorities” think about it?
- It is said that the official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”). At the same time the official Oracle documentation says that the right way to pronounce it is “sequel”.
- Before diving into the exercises you should make an important choice – are you going to pronounce Sequel or S-Q-L? More informal or more formal? Where does your heart belong to?

# Chapter II

## Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Here and further we use Python 3 as the only correct version of Python.
- The python files for python exercises (module01, module02, module03) must have a block in the end: `if __name__ == '__main__':`
- Pay attention to the permissions of your files and directories.
- To be assessed your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your `.gitignore` to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google.
- Remember to discuss on the Intra Piscine forum.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!


# Chapter III

## Specific instructions of the day

- Use Jupyter Notebook to work with your code
- For each major subtask in the list of any exercise (black bullets), your ipynb file should have an h2 heading to help your peer easily navigate in your code
- No imports allowed, except those explicitly mentioned in the section “Authorized functions” of the title block of each exercise
- You can use any built-in function, if it is not prohibited in the exercise
- On this day you can use only the following methods of Pandas: `io.sql.read_sql` and `to_sql`, except it is prescribed explicitly in the exercise.
- Save and load all the required data in the subfolder `data/`

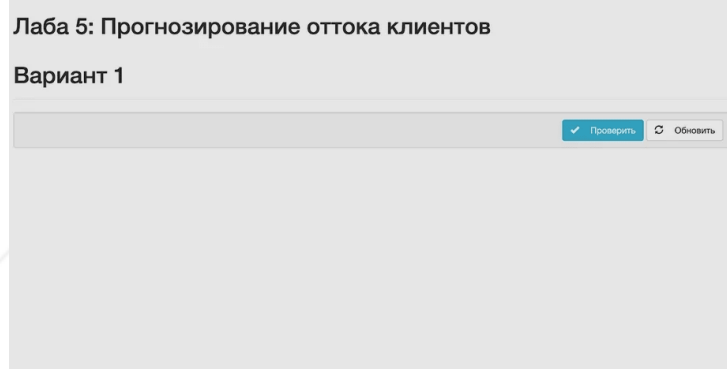
# Chapter IV

## Exercise 00 : Select

	Exercise 00
Select	
Turn-in directory : <i>ex00/</i>	
Files to turn in : <i>ex00_first_select.ipynb</i>	
Allowed functions : <code>import pandas as pd, import sqlite3</code>	

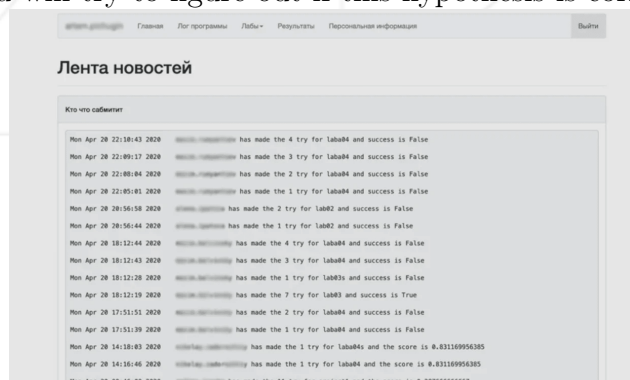
Sometimes having only Python in your toolbox can be limiting. You remember that during the first day of the Piscine you were working with command-line tools because they can be more efficient for some tasks. During this day you will work with SQL. Why can it be useful for you? Sometimes your data might be not in convenient CSVs or JSONs but stored in a database, and you have to extract it from there somehow. Also, SQL can be used in Apache Spark and in Hive (but with HQL) – tools that are used for processing big data.

Download [the SQLite database](#). During the day you will work with different tables of it using Pandas. They are all connected and refer to the same project. It is a real dataset of an educational company. They have their own platform where every student can check if their solution is correct and get some other feedback. The table checker stores the logs of when and which labs the users checked.



The company decided to create a new page on the platform – Newsfeed where those logs are visible to all the students in the program. The logs of the page visits are stored in another table – pageviews. The hypothesis is that the page would create peer pressure and the students would start working on the labs earlier. That could be good because

they could make more iterations and try different approaches. In this series of exercises, you will try to figure out if this hypothesis is correct.



But let us start from something super simple. In this exercise, you will need to get the filtered data from a table in the database. Why is it important to filter the data in the query but not after in Pandas? Because tables can be enormous. If you try to get the whole table, you will not be able to process it. Always keep this thought in mind.


The first way of filtering is by choosing only those columns that you really need. The second is by choosing the rows that you really need.

In more details:

- put the database in the subfolder data in the root directory of the day
- create a connection to the database using the library sqlite3
- get the schema of the table pageviews using `pd.io.sql.read_sql` and the query `"PRAGMA table_info(pageviews);"`
- get only 10 first rows of the table pageviews to check how the table looks like
- get the subtable using only one query where:
  - only uid and datetime are used
  - only users' data (`user_*`) is used and not admins' data
  - it is sorted by uid ascendingly
  - index column is datetime
  - datetime is converted to `DatetimeIndex`
  - the name of the dataframe is pageviews
- close the connection to the database

# Chapter V

## Exercise 01 : Subquery

	Exercise 01
Subquery	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <b>ex01_subquery.ipynb</b>	
Allowed functions : <b>import pandas as pd, import sqlite3</b>	

Ok, let us make something more complicated. Have you heard about subqueries? Like a query inside a query. Why can it be useful for you? In general, you might want to make some aggregations over a select that you had made before. Beware though that nested queries run first and only then the main query runs.


Here is what you need to do:

- create a connection to the database using the library `sqlite3`
- get the schema of the table checker
- get only 10 first rows of the table checker to check how the table looks like
- count how many rows satisfy the following conditions using only one query with any number of subqueries:
  - count the rows from the pageviews table but only with the users from the checker table with:
    - \* `status = 'ready'`, we do not want to analyze the logs that are in status checking
    - \* `numTrials = 1`, we want to analyze only the first commits, because only they can tell us when a student started working on a lab
    - \* labnames should be from the list: `'laba04', 'laba04s', 'laba05', 'laba06', 'laba06s', 'project1'`, only they were active during the experiment
  - store into the dataframe checkers with the column `cnt`
- close the connection



# Chapter VI

## Exercise 02 : Preprocessing

	Exercise 02
Preprocessing	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <b>preprocessing.ipynb</b>	
Allowed functions : <b>import pandas as pd</b>	

In this exercise, you will create a so-called datamart. It is a table that can be used for analytics purposes. Usually, it is created by joining different tables together. In the exercise we collect different data about our users: when they did their first commits, when they visited Newsfeed for the first time, etc. It will help us analyze it later.

What you need to do in this exercise (read the full task):


- create a connection to the database using the library `sqlite3`
- create a new table `datamart` in the database by joining the tables `pageviews` and `checker` using only one query
  - the table should have the following columns: `uid`, `labname`, `first_commit_ts`, `first_view_ts`
  - `first_commit_ts` is just a new name of the column `timestamp` from the `checker` table, it shows the first commit of the particular lab of the particular user
  - `first_view_ts` is the first visit of a user in the table `pageviews`, timestamp when a user visited Newsfeed
  - `status = 'ready'` still should be a filter
  - `numTrials = 1` still should be a filter
  - `labnames` still should be from the list: `'laba04'`, `'laba04s'`, `'laba05'`, `'laba06'`, `'laba06s'`, `'project1'`
  - the table should contain only the users (`uids` with `user_*`) and not the admins
  - `first_commit_ts` and `first_view_ts` should be parsed as `datetime64[ns]`

- using Pandas methods create two dataframes: test and control
  - test should have the users that have the values in first\_view\_ts
  - control should have the users that have missing values in first\_view\_ts
  - replace missing values in the control by the average of first\_view\_ts of the test users, we will use this value for the future analysis
  - save both tables into the database, you will use them in the next exercises
- close the connection

A small piece of advice – do it step by step, from simple to more complex. It will help you to debug your queries.

# Chapter VII

## Exercice 03 : Selects and aggregations

	Exercise 03
Selects and aggregations	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <code>selects_n_aggs.ipynb</code>	
Allowed functions : <code>import pandas as pd</code>	

What we did before was just data preparation. We have not got any insights about the data. It is time to change it. Remember we had the hypothesis that the users would start working on the labs earlier, if they saw Newsfeed? It means that the key metric for us is a delta between when a user started working on a lab (their first commit) and the deadline of the lab.


What you need to do in this exercise:

- create a connection to the database using the library `sqlite3`
- get the schema of the table `test`
- get only 10 first rows of the table `test` to check how the table looks like
- find among all the users the minimum value of the delta between the first commit of the user and the deadline of the corresponding lab using only one query
  - do it by joining the table with the table `deadlines`
  - the difference should be displayed in hours
  - do not take into account the lab 'project1', it has longer deadlines and will be an outlier
  - the value should be stored in the dataframe `df_min` with the corresponding `uid`
- do the same thing but with the maximum using only one query, the dataframe name is `df_max`

- do the same thing but with the average using only one query, this time your dataframe should not include the uid column, the dataframe name is df\_avg
- we want to test the hypothesis that the users who visited Newsfeed just a few times have the lower delta between the first commit and the deadline, to do this you need to calculate the correlation coefficient between the number of the pageviews and the difference
  - using only one query create a table with the columns: uid, avg\_diff, pageviews
  - uid is the uids that exist in the test
  - avg\_diff is the average delta between the first commit and the lab deadline per user
  - pageviews is the number of Newsfeed visits per user
  - do not take into account the lab 'project1'
  - store it to the dataframe views\_diff
  - use the Pandas method corr() to calculate the correlation coefficient between the number of the pageviews and the difference
- close the connection

## Chapter VIII

### Exercise 04 : Enrichment and transformations

	Exercise 04
Enrichment and transformations	
Turn-in directory : <i>ex04/</i>	
Files to turn in : <code>enrichment.ipynb</code>	
Allowed functions : <code>import pandas as pd, import numpy as np, import requests</code>	

So... let us finally find out if the Newsfeed affected the behavior of the students. Did they start to work on the labs earlier? Remember that we have two prepared tables in the database test and control. We are going to conduct something similar to A/B-test. We need to calculate what the delta between the first commit and the deadline had been before they visited the page for the first time and after that. The same thing we need to do for the control group too.

In other words, each user in the test has their own timestamp of the first Newsfeed visit. We want to calculate the average delta (first commit - deadline) before that timestamp and after that timestamp. The same thing we will do for the users in the control group. You may say: “but they did not visit the Newsfeed at all”. That is correct, and we decided earlier to use the average timestamp of the first view from the test group for the control group users.

If the delta before the first Newsfeed visit is significantly different compared to the delta after it in the test group, and we will not see the same effect in the control group, then creating the page was a great idea. We can roll it out to the whole group.

In more details:

- create a connection to the database using the library `sqlite3`
- using only one query for each of the groups create two dataframes: `test_results` and `control_results` with the columns `time` and `avg_diff` and only two rows
  - `time` should have values: after and before
  - `avg_diff` contains the average delta among all the users for the time period before each of them made their first visit to the page and after it

- take into account only the users that have observations before and after
- we still do not use the lab 'project1'
- close the connection
- have the answer: did the hypothesis might be true and the page affected the students' behavior?