Lab 2 – DD1334 – Preparations

We will use the same dataset as in Lab1, so please revise the instructions there for the setup. Create a new directory "lab2" with another copy of the mondial database inside for this lab.

Note: You can complete the lab alone or with one other student from the course. To pass the lab, you need to present your solutions in one of the lab sessions with your lab partner. Please note the deadlines and grading scheme depending on the timeliness and correctness of your submission described on the course Canvas page.

Important: Besides getting the results indicated below you and your lab partner may be asked by the TAs to explain what is happening and also solve some simple similar queries on the spot to show that you have understood the material!

Goals: This lab focusses on some slightly more advanced problems involving SQL in a Data Science setting and how to query and interact with a SQL database from python.

Before you start, please make sure you have read the relevant chapters of the book pertaining to SQL material we have covered in the lectures.

Imagined Scenario: Welcome, you have managed to work with the dataset from Lab 1, your task now is to use this database to understand and predict urban population growth!

Required software:
This lab will utilize Python, and python modules such as numpy, matplotlib, sqlite3 etc. You can either install python on your own machine or execute your scripts remotely on student-shell.sys.kth.se via SSH or use the lab computers. If you use ssh remotely from a linux/unix machine, add the "-X" option to be able to see matplotlib visualizations interactively, otherwise you can just save output as png images remotely and download them later – for example using a software such as rsync over ssh.

Lab 2 – How to submit solutions & get help

Please check the times for our lab sessions on Canvas. You will need to pass the lab by the date stated on our overview page to get an "A" and the grade decays thereafter with delays to submission.

Together with your lab partner, you can then book your short lab session help or examination appointment time slot with one of the tutors on canvas. You can use a booked lab slot either to just ask questions about labs you are currently in the process of working on, or to present your lab solution. Since these individual time slots with the tutors are only a few minutes in length, **you need to be well prepared**. Also, do not book multiple individual time slots in the same lab session.

For presenting your solution, you in particular need to

- make sure you have **reserved a lab session time-slot** with one of the available tutors for you and your lab partner on canvas. Instructions: https://community.canvaslms.com/t5/Student-Guide/How-do-I-sign-up-for-a-Scheduler-appointment-in-the-Calendar/ta-p/536
- upload your solution files **at the latest the day before the lab session** during which you want to present
- **Log into our lab session zoom video call** (see link on canvas) a few minutes before your individual lab session time slot. You will be put into the "waiting room" and the tutor will move you into an individual call with your lab partner at your assigned time slot. If you miss your timeslot, you may have to wait until the next lab session if no other time slots are free, so please be on time.
- **have all material of your solution ready to present via screen sharing** and be ready to run your code/solution live.
- **be ready to answer any additional questions** the tutor may have about your solution

Question 1 – Getting Data Ready for Analysis
a) Create a VIRTUAL VIEW called **PopData** about city population developments where we augment the yearly population data in citypops with additional information about cities and the economy of the countries within which they lie. We will use this view as the basis for our data science exploration.

Each row of PopData (one per year of data and city) should contain

from citypops:

- **year**: year of population data sample
- **name**: city name
- **population**: population for that year
- **country**: country code

Additionally add to each row, associated data from city:

- **longitude:** geographical longitude of city
- **latitude:** geographical latitude of city
- **elevation:** elevation above sea level of city

And add most recent information about country economy in which city lies from economy relation:

- **agriculture, service, industry:** percentages of gdp-composition
- **inflation:** percentage

Sample query and output on the view you should be getting:

```
sqlite> select * from popdata where name like 'Santiago%' limit 5;
year        name        population  country     longitude   latitude    elevation   agriculture  service     industry    inflation
----------  ----------  ----------  ----------  ----------  ----------  ----------  -----------  ----------  ----------  ----------
1990        Santiago    60959       PA          -80.97      8.11        101         3.7          78.4        17.9        4.1
2000        Santiago    74679       PA          -80.97      8.11        101         3.7          78.4        17.9        4.1
2010        Santiago    88997       PA          -80.97      8.11        101         3.7          78.4        17.9        4.1
1994        Santiago d  440084      C           -75.81      20.02       82          3.8          73.9        22.3        6
2002        Santiago d  423392      C           -75.81      20.02       82          3.8          73.9        22.3        6
```

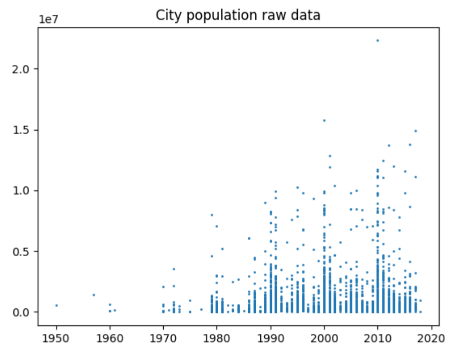
b) Sqlite3 has no direct support for materialized views. If you would use another DBMS, such as PostgreSQL, Mysql, etc, would a materialized view be a good idea for this task? Provide a short argument why or why not.

Python Interface Guideline:

Look into **figure.py** (download from Canvas) to understand how to extract basic (x,y) data points from a relation using python. Note in particular how **try … except … pass** is used to process potential SQL errors. Look into **menu.py** (download from Canvas) to see an example of a simple text menu interface to specify parameters for a particular SQL query. For the questions below, you should organize your solution into a single program with such a text menu entry for each sub-part to allow the end user to quickly run the different parts of the solutions you provide.

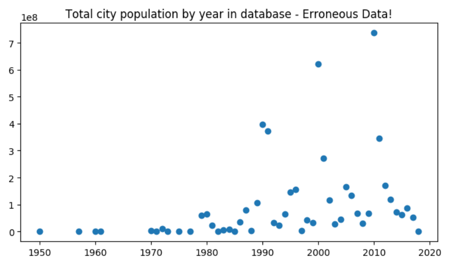
Question 2 – Data Exploration in Python

a) Generate a python function and SQL query and scatterplot of the raw yearly population data for all cities with x=year, y=population for that city and year. The result should look something like this:



To explore, look at the data for a few cities of interest as well by creating appropriate SQL queries.

b) Generate python function to create a scatterplot of x=year against y=(total sum of populations of all cities at that year) in PopData by creating an appropriate SQL query (use group by!). Your imagined junior data scientist colleague is suggesting that there must have been massive up and down swings in the total city populations due to people moving from cities to and from the countryside in the last few decades. What is wrong with this reasoning?



c) Let us attempt to predict a city's population by fitting a line y = ax + b to the y=population, x=year data per city. To accomplish this task, we will combine functions from SQL with the linear regression tools in python.

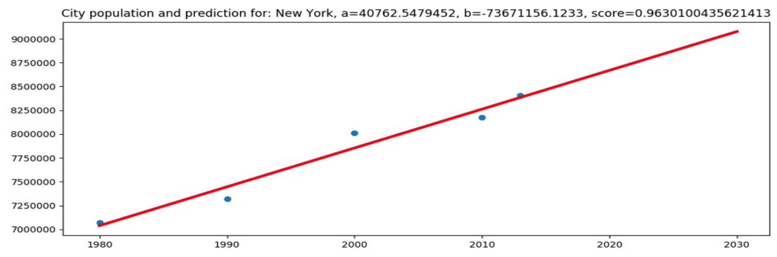
To compute the linear fit, you can use for example scikit-learn:
https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html
If you use arrays "xs" and "ys" as in figure.py to extract the year/population data, the following should work to compute a linear regression fit for one city:

```
import numpy as np
from sklearn.linear_model import LinearRegression
regr = LinearRegression().fit(np.array(xs).reshape([-1,1]), np.array(ys).reshape([-1,1]))
score = regr.score(np.array(xs).reshape([-1,1]), np.array(ys).reshape([-1,1]))
a = regr.coef_[0][0]
b = regr.intercept_[0]
```

Where a, b are the parameters from above and 'score' is the goodness of fit score (1=perfect linear trend, 0=worst).

Write a python program that requests a city country code and city name and produces 1) a plot of the available yearly population data and 2) an overlay of the linear population trend as shown below:

Example output:



d) Building on c), you are now creating a new python function that iterates though all cities one by one and creates a new relation “linearprediction” with attributes “name, country, a, b, score” with the linear prediction parameters for each of the cities. When the table already exists, the program should first drop the table and create it again. Make sure your schema for the new relation is appropriate by investigating the original schema of the the appropriate name, country columns and setting appropriate data types for a, b, score. Make sure your table has appropriate constraints specified so that invalid tuples with NULL entires are rejected and that check that the score value is not null and lies in the interval [0, 1].

When you iterate through all (city, name) pairs, make sure you use a loop using the command “fetchone()” https://docs.python.org/3/library/sqlite3.html#sqlite3.Cursor.fetchone. Why may fetchall() be inappropriate for a scalable high-quality implementation? Make sure that your code handles data issues such as Nulls etc without crashing and does not produce multiple entries for the same city. A warning: Some city names have special characters such as ‘. Read: https://docs.python.org/3/library/sqlite3.html to understand how to correctly execute queries with string parameters to avoid problems. Also review what a “cursor” is and what is represented by the corresponding python object to understand any issues you may run into.

e**)** Create a python function and menu entry that uses pure SQL commands to:

- create a new relation “prediction” with the attributes (name, country, population, year) that will hold one entry per city and year with the linear predictions for each year from 1950 to 2050 of population trends according to the parameters of our linearpredictions relation
- Use SQL to select data from linearprediction and simple arithmetic (recall y = ax + b) together with SQL INSERT statements to populate the predictions table with one tuple per year and city. The idea here is to run a python loop **over the years** with ONE SQL INSERT command per year to create all tuples for a given prediction YEAR (do not create one SQL command per tuple, which would be very inefficient)! Let the optimized internal code of SQLite3 do the work for you. Think about how you can achieve this.

f) Add another python function and menu entry which uses our new prediction table to visualize the population trends for all cities and years as a scatterplot and add the predicted mean and max predicted population for cities overall. What trends do you observe?

g) Create your own hypothesis about the population data and generate a python program that allows the user to explore this hypothesis in terms of a few sample queries and visualizations with some user specified query parameters that confirm or disprove the hypothesis. Your program should correctly process errors (use try … except … pass error handling) that may be thrown if data is not found or the input is invalid. You can use all the information in popdata and create new relations/views as desired.

Some ideas: You could group the cities into quartiles according to a size, growth-trend etc and study them. Which city population grow or decline most linearly? You could ask if there are correlations between (longitude, latitude) position of a city and population growth. What are the mean and standard deviation properties of city size over different longitude-latitude rectangles? What can we say about the cities with rapidly declining populations? Is there a connection to the type economy in which they lie in terms of service vs agriculture percentages of gdp?

To pass this question, you don’t need a complex investigation or large amount of queries, but you should 1) clearly define your hypothesis, 2) identify a few SQL queries and data visualizations that export information from the database via these SQL queries to support or reject your own initial hypothesis.