Tutorial 4: SQL

Informatics 1 Data & Analysis

Week 6, Semester 2, 2017/18

This worksheet has three parts: tutorial *Questions*, followed by some *Examples* and their *Solutions*.

- Before your tutorial, work through and attempt all of the Questions in the first section. If you get stuck or need help then ask a question on *Piazza*.
- The Examples are there for additional preparation, practice, and revision.
- Use the Solutions to check your answers, and read about possible alternatives.

You must bring your answers to the main questions along to your tutorial. You will need to be able to show these to your tutor, and may be exchanging them with other students, so it is best to have them printed out on paper.

If you cannot do some questions, write down what it is that you find challenging and use this to ask your tutor in the meeting.

Tutorials will not usually cover the Examples, but if you have any questions about those then write them down and ask your tutor, or post a question on *Piazza*.

It's important both for your learning and other students in the group that you come to tutorials properly prepared. Students who have not attempted the main tutorial questions will be sent away from the tutorial to do them elsewhere and return later.

Some exercise sheets contain material marked with a star $\star$. These are optional extensions.

Data & Analysis tutorials are not formally assessed, but the content is examinable and they are an important part of the course. If you do not do the exercises then you are unlikely to pass the exam.

Attendance at tutorials is obligatory: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

Please send any corrections and suggestions to Ian.Stark@ed.ac.uk

Introduction

In the previous tutorial you constructed some queries around a set of relational tables dealing with air travel. In this tutorial, you will formulate queries in SQL for the same application domain, with similar (but not identical) tables. You will also interact with a database using *LibreOffice Base* to execute these queries.

LibreOffice Base

In this tutorial, you will be using the desktop application *LibreOffice Base*. To set things up on a DICE machine you should carry out the following steps.

(a) Download travel.odb from the course web page http://blog.inf.ed.ac.uk/da18/tutorials

(b) From the desktop start Applications → Office → LibreOffice Base.

(c) When the *Database Wizard* window appears, choose ⟨Open an existing database file⟩.

(d) Click ⟨Open⟩.

(e) Locate the travel.odb file you downloaded earlier and click ⟨Open⟩

(f) In the Database column on the left-hand side, click Tables.

You can now select individual tables to inspect their content. To start from the command line instead, enter libreoffice travel.odb in a terminal window at the appropriate directory.

The air travel data is now loaded and you are ready to start the tutorial. Make sure you regularly save any data you wish to keep, and print out results to bring to tutorials.

If you wish to try the tutorial on a non-DICE machine then you may want to install your own copy of LibreOffice from http://www.libreoffice.org. Versions are available for Linux, Windows and Mac. The Macs in the Main Library have LibreOffice installed, and the Windows machines in open access computer labs have the very similar *OpenOffice Base* (not the ones in the AT café area, though).

⋆ Extensions

There are three possible extension exercises: using the LibreOffice GUI; connecting by command line to a remote *PostgreSQL* database server run by the school; and linking together LibreOffice with PostgreSQL as a back-end server.

Details for all three of these follow the regular exercises. If you have completed the standard questions, then pick any extension and try it out.

While there are instructions on the extensions, these are in general more technically challenging than the main exercises. If you are attempting any of them, especially on your own computer, then I recommend sharing on Piazza information about what works and contacting others there trying the same things.

SQL Details

Remember the following SQL syntax points from lectures:

- SQL keywords are not case sensitive; identifiers like table or field names may or may not be. Keywords are often written in upper case, and it is good practice to maintain a consistent approach to the cases used in table and field names.
- SQL keywords never contain spaces and never require quotation; SQL identifiers can be quoted using double quotation marks "like this" and if quoted can contain spaces.
- Strings in SQL usually are case sensitive, and must always be quoted, using single quotation marks 'like this'.

Question 1: Queries in SQL

The tables in this database have the same design as the last tutorial. There are four tables: named Airport, Booking, Flight and Seat. Below you will find a series of English-language queries about this data; for each you should do the following:

- Formulate the specified query in SQL;
- Run the query on the travel database;
- Print out the SQL query and the result table so you can bring them to the tutorial.

To run the query on the travel database, proceed as follows.

- Click on Queries in the left-hand Database column of the travel-window.
- From the list shown in the Tasks panel, choose Create Query in SQL View....
- Enter your query in the white space in the window that opens.
- Type ⟨Ctrl+S⟩ to save the query with the name of your choice.
- Go back to the travel-window and double-click on the query name. Results will then appear in a new window.
- To edit any query, right click on it in the Query panel and choose Edit in SQL View....

To print the result of a query, open a new text document with *LibreOffice Writer*, and drag the query from the travel window into the new document. You'll be given the option to choose some columns: it's simplest to click the "copy all" button (·>|).

To print the SQL source of a query, edit it in SQL view then select all text and again drag it to a *LibreOffice Writer* document, or the editor of your choice.

When editing your SQL source, you may notice that *LibreOffice* has capitalised keywords and generously added quotation marks around every identifier in sight. This does not change the meaning of the query, but can make it more portable between different database servers.

Queries

(a) Retrieve rows from the Airport table for all the airports in London. The schema of the output table should be same as that of the Airport table.

(b) List all bookings by British and French passengers. The schema of the output table should be same as that of the Booking table.

(c) Retrieve the names of all booked passengers.

(d) Find the flight number, departure airport and arrival airport for every British Airways flight in the database.

(e) Retrieve the name of every passenger together with their flight number and the associated flight company.

(f) Tabulate all flights from airports in London. The output schema should be same as that of the Flight table.

(g) Obtain ticket numbers and names of all passengers departing from London.

(h) Retrieve the flight number and flight company of all flights from London to Paris.

⋆ **(i)** List the ticket numbers and names of all passengers travelling in Business class.

⋆ **(j)** Find the names and nationalities of all the Business class passengers travelling from London to Paris.

Question 2: Discussion

So far in this course you have used Relational Algebra, Tuple-Relational Calculus and SQL to formulate queries on tables. From your experience, is SQL more similar to Relational Algebra or Tuple-Relational Calculus? List differences between the real-world query language SQL and the more mathematically abstract languages of Relational Algebra and Tuple-Relational Calculus. What would you say are the reasons for each of these differences?

⋆ Extension 1: Graphical Query Editing

LibreOffice Base has a graphical user interface to generate and execute queries, with its *Design View* and query creation *Wizard*. The Wizard provides prompts to build simple queries, which can then be edited in the Design View to create more complex ones.

Query Creation Wizard

We'll build a query to fetch the names of all airports in Paris. In the main database window, click on the Queries icon in the Database column, and then in the Tasks section click on Use Wizard to Create Query.... This opens the Query Wizard window.

1. Field Selection

(a) Choose the Airport table from the drop-down list of tables.

(b) Select from the Available fields list.

(i) Click on the name field and use the > button to move it to the list Fields in the Query.

(ii) Do the same thing to add the city field. Click ⟨Next⟩.

2. Sorting order It's possible to sort the results of the query using one or more fields. In this case, there's no need to for sorting, so you can leave the default - undefined - option selected. Click ⟨Next⟩.

3. Search conditions These allow us to indicate which database records we are interested by describing the values of relevant fields. Selection may involve one or more tests.

(a) We are using only query condition, so it makes no difference whether we use the default Match all of the following option or the alternative Match any of the following.

(b) We are looking for airports in a particular city, so select Airport.city from the Fields dropdown and leave the condition as the default is equal to. You might be interest to look in the Condition dropdown to see what possibilities the wizard offers.

(c) Type Paris as the value to match. Click ⟨Next⟩.

7. Aliases The wizard automatically skips steps 4–6 as we don't need those. It also makes a sensible automatic assignment of aliases for the fields involved. Click ⟨Next⟩.

8. Overview Name the query as Query_Paris_Airports and select the Modify Query option. Click ⟨Finish⟩.

Run the query You should now see a new window for viewing and editing the query. This has a panel showing details of the tables used in the query, and grid with fields from the query and how they are used. You can run the query by selecting Edit → Run Query, pressing F5, or clicking on the icon with a pile of disks and a downward arrow. This will produce another panel at the top containing the result of the query. It should now display a table listing the airports Charles de Gaulle and Orly as both in Paris. Save the query.

Query Design View

This three-part window is the *Design View* and can be used to create and edit queries. You can toggle between Design View and SQL using the View menu or the icon of a database table overlaid with a set-square (triangle).

The panel in the middle shows the different tables used in a query, listing all their fields. It can also show links between these fields. The panel at the bottom has a column for each field used in the query. So far we have name and city mentioned, with 'Paris' listed as a criterion for the city field to show we are selecting only tuples with that value.

We're now going to build up to a more complex query, listing the nationalities of passengers travelling to Paris with Air France.

Step 1: Hide the city field Since we are only listing airports in Paris, the city field isn't really needed in the result table. Hide this by unticking the Visible box in the city column of the grid. Now run the query again to see the outcome of this.

Step 2: Add the table of flights Use the menu entry Insert → Add Table or Query (or press F7) and select the Flight table. Click ⟨Add⟩ then ⟨Close⟩. You should now see two tables in the central panel. They are linked by a line — we'll explore that later.

Step 3: Select flight information Add the fields flightNo and flightCompany to the query grid. You can either double-click on the field name in the central panel, or drag and drop them to a free column in the grid. We want to select flights operated by Air France, so in the flightCompany column write 'Air France' as the criterion.

Step 4: Link two tables The central panel has automatically inserted a link between the Airport.airportId and Flight.depAirport fields. You may need to move the tables around to see this properly. This means that the query will select only flights leaving from the selected airport. Run the query and you should see three Air France flights, AF1232, AF1235 and AF1343, all leaving from Charles de Gaulle airport.

We can change this to show flights arriving in Paris instead. First, right-click the connecting link between the tables and select Delete. Second, click on the arrAirport field and drag to connect it to the airportId field of the Airport table. When you release the mouse button, a line connecting the two fields in the two windows appears. This sets a condition in the query the content of the two fields must match: if you toggle to the SQL view this should be visible. Run the query again and the output table will be updated, this time showing four flights.

Step 5: Add and connect booking information Now add the Booking table. Again, there is an automatic link added to the Flight table, prompted by the foreign key constraint between the tables. Double-click the nationality field from the Booking table to add it to the query.

Step 6: List the nationalities Finally, untick the Visible box for all columns except nationality. Run the query again, and you should have a list of all nationalities recorded for passengers booked to fly out of Paris with Air France. Notice that this includes repeat elements: as this is SQL, we have a *multiset* rather than a set. Save the query.

Step 7: Count nationalities As a final step, suppose that we wish to report how many different nationalities there are among these passengers. Go back to the design view and look for the Function row in the grid. For the nationality column, click on that row and select Count from the dropdown that appears. Run the query.

This gives the answer 4, because the earlier table had exactly four rows. However, this is a multiset and includes repetitions. To count the number of *different* nationalities, you need to switch to the SQL view. Do that, and modify the query by inserting DISTINCT before "Booking"."nationality". Run the query again and check you get the answer 2.

Exercise

Go back to Question 1 and implement some of the queries there using either the wizard or the design view. For example, try out queries (e), (h) and (i).

Based on your experience with the example and exercise above, how do you evaluate the *Query Creation Wizard* and the *Query Design View*? Can they create the same range of queries? If not, then what are the differences? How do you find them compared to writing SQL directly?

⋆ Extension 2: PostgreSQL

The School of Informatics runs a dedicated database server for teaching, on the pgteach machine. This runs the *PostgreSQL* open-source database software, also called simply *Postgres*. The inf1da database on the server has copies of the travel and film tables from this tutorial.

For direct access to the PostgreSQL command line, open a terminal window on a DICE machine and enter the following.

```
psql96 -h pgteach inf1da
```

Here psql96 invokes version 9.6 of the PostgreSQL client, which matches the server version; while -h pgteach indicates the server machine and inf1da the target database.

This will give you a secure connection to the SQL server and load the correct database. You can now type two different kinds of command:

- SQL commands, like **select** ∗ **from** actor; Make sure you end with a semicolon ';' If you use more than one line the prompt changes from inf1da=> to inf1da->.
- Commands to the server itself. These start with a backslash '\'. For example: \dt will list all the tables; \dtS will list all the system tables used to manage the database server itself; and \? will give help on all these commands. Type \q to quit the server.

These should be enough to explore the system and try out all your SQL queries from Question 1.

To access the server from a non-DICE machine, it's probably simplest to ssh into the DICE server student.ssh.inf.ed.ac.uk, then on to student.login, and finally run psql96 there.

Exercises

These queries use the film tables in the inf1da database.

Note: PostgreSQL automatically transforms all unquoted identifiers, such as table or field names, into lowercase. Some of the inf1da tables have mixed-case field names. You will need to use quotation for these, such as actor."actorId".

(a) Create a query to return all actors that have worked with the director Christopher Nolan.

(b) Create a query that lists all actors in the database by name and year they first appear in the database. Sort the list in ascending order by year.

⋆ Extension 3: Link LibreOffice Base to PostgreSQL

It is also possible to connect to the pgteach database using LibreOffice Base and then use that to write queries. Before doing so you need to set up a password on your PostgreSQL account. Connect through the command line as in the previous section and create a password by entering this SQL command:

```
alter role username with password 'new password here';
```

Fill in your own username and choice of password. For example:

```
alter role s1234567 with password 'correcthorsebatterystaple';
```

You should see the shell respond with **ALTER ROLE** as confirmation.

From here on, the instructions are different depending on whether or not you are accessing from a DICE machine inside Informatics. From a DICE machine, do as follows.

- Open LibreOffice Base. Choose ⟨Connect to an existing database⟩ in the *Database Wizard* window and select ⟨JDBC⟩ in the drop-down menu below.
- For the Datasource URL type in postgresql://pgteach/inf1da. The JDBC Driver Class you need is org.postgresql.Driver, so type that and click ⟨Test Class⟩. Once that works, click ⟨Next⟩.
- After that enter your DICE username, tick Password required and click ⟨Test Connection⟩. This should request the password you set earlier and then announce success. Click ⟨Next⟩.
- In the final window of the wizard no changes need to be made. Click ⟨Finish⟩ and LibreOffice will ask you where to store a database file. This records the connection information and any queries you add so that you can access them later.

If using a non-DICE machine then you will need to do two additional pieces of configuration. The first is to set up a VPN (virtual private network) connection to access the server, which is secured inside the Informatics network. The second is to install some Java code with a driver that will connect LibreOffice to the PostgreSQL server.

- Set up an OpenVPN network connection to Informatics. To do this, follow the guidance on the Informatics computing help site http://computing.help.inf.ed.ac.uk/openvpn . For more general information on remote connections to Informatics, read the online guide at http://computing.help.inf.ed.ac.uk/remote-working .

  The Informatics OpenVPN service offers several alternative configuration files. They should all work fine, but the standard choice here is Informatics-via-AT.ovpn .

- Install the Java PostgreSQL driver. You will need to identify which version is appropriate for your system. Open the LibreOffice startup application — or, if that is not available, LibreOffice Writer. Navigate to the Java Options settings panel:

  Menu entry Tools ⟶ Options ⟶ LibreOffice ⟶ Advanced.

  Make sure ⟨Use a Java runtime environment⟩ is ticked. Wait a while for the white text box to populate: it should shortly show one or more Java runtime environments. Note the version number of the one currently selected: it will probably begin with 1.6, 1.7 or 1.8.

  You need to install a driver file matching this version of Java. While keeping the LibreOffice settings panel open, go to https://jdbc.postgresql.org/download.html and read the *Current Version* section. Download the PostgreSQL driver version it recommends, probably one of JDBC 4.2, JDBC 4.1 or JDBC 4.0. Store the file somewhere you can find it again.

- Go back to the LibreOffice settings panel and click ⟨Class Path⟩. In the popup window click ⟨Add Archive⟩ and select the JAR file you downloaded with the PostgreSQL driver. Click ⟨OK⟩ and close LibreOffice.

- Now open LibreOffice Base. Choose ⟨Connect to an existing database⟩ in the *Database Wizard* window and select ⟨JDBC⟩ in the drop-down menu below.

- For the Datasource URL enter postgresql://pgteach.inf.ed.ac.uk/inf1da. In the box labelled JDBC Driver Class type in org.postgresql.Driver and click ⟨Test Class⟩. Once that works, click ⟨Next⟩.

- After that enter your DICE username, tick Password required and click ⟨Test Connection⟩. This should request the password you set earlier and then announce success. Click ⟨Next⟩.

- In the final window of the wizard no changes need to be made. Click ⟨Finish⟩ and LibreOffice will ask you where to store a database file. This records the connection information and any queries you add so that you can access them later.

After the connection is established, the database will open and you should be able to see the tables by clicking on Tables in the Database column on the left-hand side. Go back and try the exercises from Extension 2, working with the Film database.

Examples

This section contains some examples of SQL queries designed to answer English-language questions. The examples are very similar to the main tutorial questions, but using a database of information about films rather than air travel.

Download the database file film.odb from the course web page: http://blog.inf.ed.ac.uk/da18/tutorials. You will notice that film.odb contains similar (but not identical) tables to those presented in the *Examples* section of Tutorial 3. There are four tables — Actor, Film, Performance and Director. For each of the English-language queries below, questions do the following:

- Formulate the specified query in SQL;
- Run the query on the film database;
- Print out the SQL query and the result table.

See the instructions from Question 1 for more detail on how to enter and edit SQL queries in LibreOffice.

Queries

(a) Retrieve details of all films that were released in 2010. The output schema should be the same as that of the Film table.

(b) Retrieve details of all actors that are not in their thirties. The output schema should be the same as that of the Actor table.

(c) Retrieve the names of all directors.

(d) Retrieve the names of all American directors.

(e) Find out the names of all British actors above the age of 40.

(f) List all performances by an actor, giving for each one the actor's name and the film in which he or she appeared.

(g) Find out the names of all actors that have played the part of Bruce Wayne (Batman).

(h) Retrieve the names of all actors that have played the part of Bruce Wayne, together with the year the corresponding films were released.

(i) Retrieve all actors from the film Inception. The output schema should be the same as that of the Actor table.

(j) Find out the names of all actors that have performed in a film directed by Christopher Nolan.

(k) Retrieve the titles of all films in which Leonardo Di Caprio and Kate Winslet have co-acted.

(l) Assuming that the ids of actors and directors are used consistently across the tables, retrieve details of all actors that have directed a film.

Solutions to Examples

These are not entirely "model" answers; instead, they indicate a possible solution. Remember that not all of these questions will have a single "right" answer. There can be multiple appropriate ways to formulate a query.

If you have difficulties with a particular example, or have trouble following through the solution, please raise this as a question in your tutorial.

(a) Retrieve details of all films that were released in 2010. The output schema should be the same as that of the Film table.

```
select * from Film where yr = 2010
```

filmId	title	yr	directorId
DSP10	Despicable Me	2010	PCF97
INC10	Inception	2010	CN345
SHI10	Shutter Island	2010	SCO78

(b) Retrieve details of all actors that are not in their thirties. The output schema should be the same as that of the Actor table.

```
select * from Actor where age<30 or age>39
```

actorId	name	nationality	age
CB379	Christian Bale	British	40
DP423	Dev Patel	British	24
ELP87	Ellen Page	American	27
EMG32	Ewan McGregor	British	43
HBC54	Helena Bonham Carter	British	48
JD801	Judi Dench	British	80
LDC21	Leonardo DiCaprio	American	40
MKE12	Michael Keaton	American	63

(c) Retrieve the names of all directors.

```
select name from Director
```

name
Darren Aronofsky
Tim Burton
Christopher Nolan
Danny Boyle
James Cameron
Michael Keaton
Pierre Coffin
Martin Scorsese
Sam Mendes

(d) Retrieve the names of all American directors.

```
select name from Director where nationality = 'American'
```

name
Darren Aronofsky
Tim Burton
Michael Keaton
Martin Scorsese

(e) Find out the names of all British actors above the age of 40.

```
select name from Actor
where nationality = 'British' and age>40
```

name
Ewan McGregor
Helena Bonham Carter
Judi Dench

(f) List all performances by an actor, giving for each one the actor's name and the film in which he or she appeared.

```
select name, title from Actor A, Performance P, Film F
where A.actorId = P.actorId and P.filmId = F.filmId
```

name	title
Christian Bale	The Dark Knight
Christian Bale	The Dark Knight Rises
Dev Patel	Slumdog Millionaire
Ellen Page	Inception
Ewan McGregor	Big Fish
Ewan McGregor	Trainspotting
Helena Bonham Carter	Big Fish
Judi Dench	Skyfall
Joseph Gordon-Levitt	The Dark Knight Rises
Joseph Gordon-Levitt	Inception
Kate Winslet	Revolutionary Road
Kate Winslet	Titanic
Leonardo DiCaprio	Inception
Leonardo DiCaprio	Revolutionary Road
Leonardo DiCaprio	Shutter Island
Leonardo DiCaprio	Titanic
Michael Keaton	Batman Returns

(g) Find out the names of all actors that have played the part of Bruce Wayne (Batman).

```
select distinct name from Actor A, Performance P
where A.actorId = P.actorId and P.part = 'Bruce Wayne'
```

Alternate solutions are possible, like this one using a nested query:

```
select distinct name from Actor
where actorId in (select actorId from Performance where part = 'Bruce Wayne')
```

name
Christian Bale
Michael Keaton

(h) Retrieve the names of all actors that have played the part of Bruce Wayne, together with the year the corresponding films were released.

```
select A.name, F.yr from Actor A, Performance P, Film F
where A.actorId = P.actorId and P.part = 'Bruce Wayne' and P.filmId = F.filmId
```

Here is another presentation with explicit joins — the tables mentioned and all tests are just the same, only arranged to show one way of joining them to evaluate the query.

```
select A.name, F.yr
from Actor A
  join Performance P on A.actorId = P.actorId
  join Film F on P.filmId = F.filmId
where P.part = 'Bruce Wayne'
```

name	yr
Christian Bale	2008
Christian Bale	2012
Michael Keaton	1992

(i) Retrieve all actors from the film Inception. The output schema should be the same as that of the Actor table.

```
select A.* from Actor A, Performance P, Film F
where A.actorId = P.actorId and P.filmId = F.filmId and F.title = 'Inception'
```

actorId	name	nationality	age
ELP87	Ellen Page	American	27
JGL81	Joseph Gordon-Levitt	American	33
LDC21	Leonardo DiCaprio	American	40

(j) Find the names of all actors that have been in a film directed by Christopher Nolan.

```
select distinct A.name from Actor A, Performance P, Film F, Director D
where A.actorId = P.actorId and P.filmId = F.filmId
   and F.directorId = D.directorId and D.name = 'Christopher Nolan'
```

name
Christian Bale
Ellen Page
Joseph Gordon-Levitt
Leonardo DiCaprio

(k) List the titles of all films in which both Kate Winslet and Leonardo Di Caprio appeared.

```
select distinct F.title from Film F, Actor A1, Actor A2,
                             Performance P1, Performance P2
where A1.name = 'Leonardo DiCaprio' and A2.name = 'Kate Winslet'
   and A1.actorId = P1.actorId and P1.filmId = F.filmId
   and A2.actorId = P2.actorId and P2.filmId = F.filmId
```

title
Revolutionary Road
Titanic

(l) Assuming that the ids of actors and directors use the same scheme, retrieve details of all actors that are also in the database as having directed a film.

select A.∗ **from** Actor A, Director D
where A.actorId = D.directorId

actorId	name	nationality	age
MKE12	Michael Keaton	American	63