Database Technologies for Business Analytics
BEM2040
Practice – Week 4

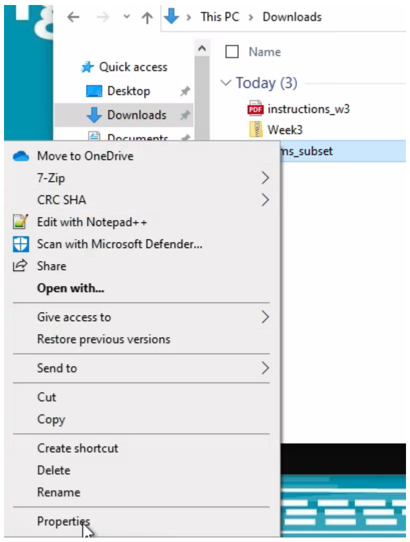
The following instructions can be followed by using university computers, a virtual desktop or your personal computer, if the software has been installed.

1. Download the file Week4.zip. We will be using:

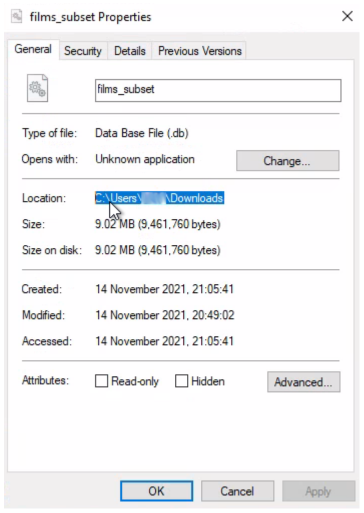
   – films_subset.db
   The tables in the database are as follows:

```
ACTOR:
        ACTOR_ID,
        ACTOR_NAME,

FILM:
        FILM_ID,
        FILM_TITLE,
        FILM_YEAR_END

FILM_ACTOR_PARTICIPATION:
        FILM_ID,
        ACTOR_ID

FILM_RATING_TYPE:
        RATING_TYPE_ID,
        RATING_TYPE_NAME

FILM_RATING:
        FILM_RATING_TYPE_ID,
        FILM_ID,
        FILM_RATING_VALUE
```

2. Decompress the files in your downloads folder.
3. Get the location of the database file: right-click the file in the explorer, select *Properties* and copy the text in *Location*:



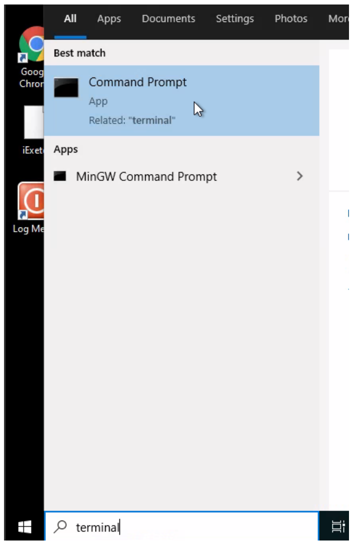
→



films_subset

Type of file: Data Base File (.db)
Opens with: Unknown application
Location: C:\Users\[redacted]\Downloads
Size: 9.02 MB (9,461,760 bytes)
Size on disk: 9.02 MB (9,461,760 bytes)
Created: 14 November 2021, 21:05:41
Modified: 14 November 2021, 20:49:02
Accessed: 14 November 2021, 21:05:41
Attributes: Read-only Hidden

4. We will be working with the command line tool for SQLite. To open it, type "terminal" in the windows starting search box and click on *Command prompt*:



The terminal would look like this, from a university computer (instead of ab000 you should see you user name):

```
Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. All rights reserved.
d:\ab000.ISAD>
```

5. Open the database with the sqlite3 command. Type *sqlite3*, then a blank space and then right click. The folder you copied should appear. Then complete it so that the entire line looks similar to this:

```
d:\ab000.ISAD>sqlite3 C:\Users\ab000\Downloads\films_subset.db
```

Press enter and you should be working on the database:

```
SQLite version 3.27.2 2019-02-25 16:06:06
Enter ".help" for usage hints.
sqlite>
```

6. Let's see the content of table *film*. Write the instruction

   ```
   select * from film;
   ```

   in the prompt:

   ```
   sqlite> select * from film;
   ```

   The instruction has to end in semicolon (;). The result would be as follows (only the last rows shown here):

   ```
   tt0081222|Nem Amantes, Nem Amigos|1970
   tt0081553|Wutai jiemei|1964
   tt0085649|Harry's Girls|1963
   tt0086441|Pipo|1970
   tt0090647|Anna|1967
   ```

7. Let's now obtain films for a specific year (1960). The instruction is:

   ```
   select * from film where film_year_end=1960;
   ```

   In the prompt:

   ```
   sqlite> select * from film where film_year_end=1960;
   ```

   ```
   tt0057651|Via Margutta|1960
   tt0058766|Yurei Hanjo-ki|1960
   tt0059029|Ching nu yu hun|1960
   tt0063898|Flucht aus der Hölle|1960
   tt0064260|Le songe des chevaux sauvages|1960
   ```

8. Now, let's look at table *actor*. The instruction

   ```
   select * from actor;
   ```

   In the prompt:

   ```
   sqlite> select * from actor;
   ```

   ```
   nm0104752|David Brandon
   nm0104772|Jane Alice Brandon
   nm0104799|Roy Brandon
   nm0104830|X Brands
   nm0104837|Jutta Brandstaedter
   ```

9. Let's add some rows:

   ```
   insert into actor(actor_id, actor_name)
   values ("a1", "Timothée Chalamet");
   ```

   ```
   sqlite> INSERT into actor(actor_id,actor_name)
   VALUES ("a1", "Timothée Chalamet");
   ```

   ```
   INSERT into actor(actor_id,actor_name)
   VALUES ("a2", "Rebecca Ferguson");
   ```

   ```
   sqlite> INSERT into actor(actor_id,actor_name)
   VALUES ("a2", "Rebecca Ferguson");
   ```

   ```
   INSERT into actor(actor_id,actor_name)
   VALUES ("a3", "Greta garbo");
   ```

   ```
   sqlite> insert into actor(actor_id,actor_name)
   values ("a3", "Greta garbo");
   ```

10. Check the table *actor* after the inserts:

```
sqlite>  select * from actor;
nm0104830|X Brands
nm0104837|Jutta Brandstaedter
a1|Timothée Chalamet
a2|Rebecca Ferguson
a3|Greta garbo
```

11. Now lets add headers to the output and activate the column mode to make the output more readable. Use the commands:

```
.header on
.mode column
```

```
sqlite> .header on
sqlite> .mode column
```

We can now see the column names (actor_id, actor_name, etc.) in the output when querying:

```
select * from actor;
```

```
sqlite> select * from actor;
nm0104830  X Brands
nm0104837  Jutta Brandstaedter
a1         Timothée Chalamet
a2         Rebecca Ferguson
a3         Greta garbo
```

12. Lets restrict the results to those actors whose name starts with 'Z':

```
select * from actor where actor_name like 'Z%';
```

```
sqlite> select * from actor where actor_name like 'Z%';
ACTOR_ID   ACTOR_NAME
---------  -------------------
nm0017920  Zhanna Aleksandrova
nm0027912  Zurab Anjaparidze
nm0033231  Zaida Araujo
nm0039559  Zinaida Asmolova
nm0059896  Zoltán Basilides
nm0065054  Zoran Becic
nm0067400  Zoltán Beke
nm0070524  Zoran Benderic
nm0073960  Zoraida Bergamini
nm0075444  Zvonimir Berkovic
nm0075678  Zeev Berlinsky
nm0079203  Zina Bethune
nm0090076  Zdenek Bláha
nm0091687  Zbigniew Bogdanski
nm0097175  Zvi Borodo
```

13. We can update one row, based on the id:

```
update actor set actor_name = 'Greta Garbo' where actor_id = "a3";
```

And check the result:

```
sqlite> select * from actor where actor_id = 'a3';
ACTOR_ID  ACTOR_NAME
--------  -----------
a3        Greta Garbo
```

14. We can order results of a query in ascending or descending directions.

    For example, by year:

    ```
    select * from film order by film_year_end;
    ```

    ```
    sqlite> select * from film order by film_year_end;
    tt0074953   Narda o el verano         1970
    tt0075009   On est au coton           1970
    tt0081222   Nem Amantes, Nem Amigos   1970
    tt0086441   Pipo                      1970
    tt0095436   Kemek                     1970
    ```

    Let's try in descending order:

    ```
    select * from film order by film_year_end desc;
    ```

    ```
    sqlite> select * from actor order by actor_year_born desc;
    tt0057651   Via Margutta                  1960
    tt0058766   Yurei Hanjo-ki                1960
    tt0059029   Ching nu yu hun               1960
    tt0063898  Flucht aus der Hölle           1960
    tt0064260  Le songe des chevaux sauvages  1960
    ```

    We can substitute * in a *select* query for specific columns:

    ```
    select film_id, film_title from film where film_year_end = 1965;
    ```

    ```
    sqlite> select film_id, film_title from film where film_year_end = 1965;
    tt0065066  A Tall, Stalwart Lancer
    tt0065110  Tonkina jedina ljubav
    tt0067664  Die reise nach Steiermark
    tt0067864  Tokoloshe
    tt0072579  Vision On
    ```

15. We can use the following operators:
    $<$,
    $>$,
    $<=$,
    $>=$,
    $=$,
    $<>$,
    IN,
    LIKE,
    IS NULL

    Try the following queries:

    – Films from 1965 or after:

    ```
    select * from film where film_year_end >= 1965;
    ```

    – Films where the title has the sequence of letters "sea":

    ```
    select * from film where film_title like '%sea%';
    ```

    – Films where the title ends in "country":

    ```
    select * from film where film_title like '%country';
    ```

16. We can delete a single row if we can identify it uniquely. Let's suppose we want to eliminate the actors we inserted before.,

    Let's first check the record:

    ```sql
    select * from actor where actor_id = 'a1';
    ```

    We then issue the following instruction to delete it:

    ```sql
    delete from actor where actor_id = 'a1';
    ```

    ```
    sqlite> delete from actor where actor_id = 'a1';
    ```

17. We can delete several rows. The following deletes all movies from 1970:

    ```sql
    delete from film where film_year_end = 1970;
    ```

    ```
    sqlite> delete from film where film_year_end = 1970;
    ```

18. We can verify the result by counting the number of films from that year:

    ```sql
    select count(*) from film where film_year_end = 1970;
    ```