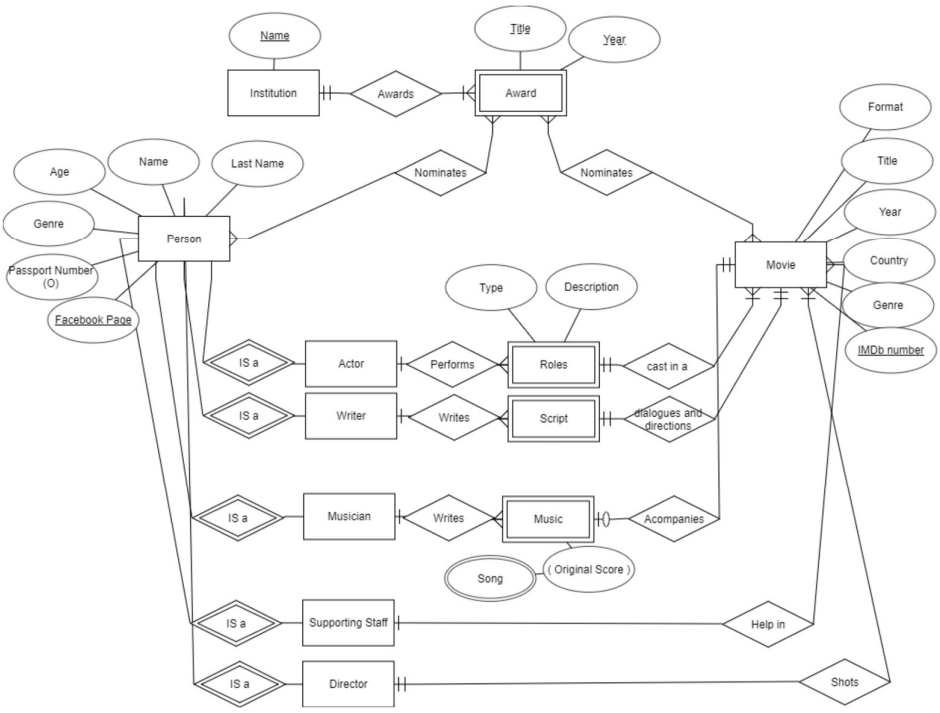
Coursework 1 DBMS

Part I: Entity-Relationship Diagram



STEPS

1. **Identify entity types / Primary Keys**
   - Institution. Its name is its Primary Key and it is a Foreign Key for Award.
   - Award. It is a weak Entity. Only can be identified if linked to an Institution. For instance, Best Movie Award in 2013 can only be identified if the Institution is as well declared.
   - Movie. Not easy to find a Primary Key. I have chosen IMDb (very well known database in the industry that identifies uniquely each movie).
   - Person. As mention in the text they are identified by their Facebook page. Note that in real world this can be tricky since not everybody may have one. Other id could be the Passport number (almost everybody has one despite not being mandatory)
   - Actor: IS A entity (speciality of a Person).

I have added other entities (IS A, specialties with special properties derived not from Person Entity) considered in the question (they all inherit the attributes and primary keys of Person):

- Writer.
- Musician.
- Supporting Staff (I have grouped here all supporting roles needed for a movie that can be nominated to an award: production designers, Hairstylers, Costume Designers,…)

- Director.

Finally, I have added some weak entities (linked to movies) that may be part of an award nomination (given to the person who created them.

- Roles. Weak entity, only can be determined through its link to a movie. Can be Leading or Supporting.
- Script. Weak entity, Linked to a movie.
- Music. Weak entity. Linked to a movie.

2. **Identify relationships (ISA / ID).**

Actor, Director, Supporting Role, Writer, Musician are ISA entities derived for Person Entity (Parent).

3. **Identify relationships – multiplicity**
   - Institution awards an Award. Many to Many relationship. One Institution (Oscars) can give many awards. An Award (Best Actress in a Leading Role) can be given by several institutions (for instance Oscars and BAFTA).
   - Award is nominated to a Person. Many to Many relationship. A person can be nominated with Many Awards (Best Actor, Best Director, Best Actor in a secondary role,…) and an Award can nominate several person (before they are awarded), but even the final award can be given to more than one person (for instance there are more than one Production Designer in a movie).
   - Award is nominated to Movie. Again a movie can be nominated with several awards (Best movie, Best foreign movie,…) and an award can nominate several movies (until one is awarded).
   - Person is an actor. Inherits their main attributes.
   - Actor performs a Role. One to many relationship: one actor can perform several roles in a movie (although it is not usual) but a role can only be perform by one actor (I consider that a character performed by 2 different actors are two different actors).
   - Role casts in a Movie. Role is a weak entity (only can be identified in the context of a movie). It is a mandatory (a role has to perform in a movie) and one to many relationship: a movie may have many roles but a role is performed in a specific movie.

4. **Participation constraints.**
   - There are several mandatory relationships:
     - Between Institution and Award. Every Award should be awarded / nominated by a Institution. Award is a weak entity that only can be identified when referred to an Institution.
     - An Actor has to perform a role, the nomination is to an actor playing an specific role in a movie.
     - A role has to be linked to a movie. As mentioned before, the role has to be performed in a movie.

5. **Draw ERD with entity types /relationships.** See first Image. I have used IDE1FX notation.

6. **Identify attributes entity – relationships.**
   - Institution. Name.
   - Award. Name, Year.

- Movie. Title, Year, Country /Nationality, IMDb, Genre, Format (for instance Film, Animation, Documentary,…).
- Person. Name, Last Name, Age, Genre, Passport N (Optional), Facebook page Id
- Actor: IS A entity (speciality of a Person).
- Role: Description and Type (Leading or supporting role).

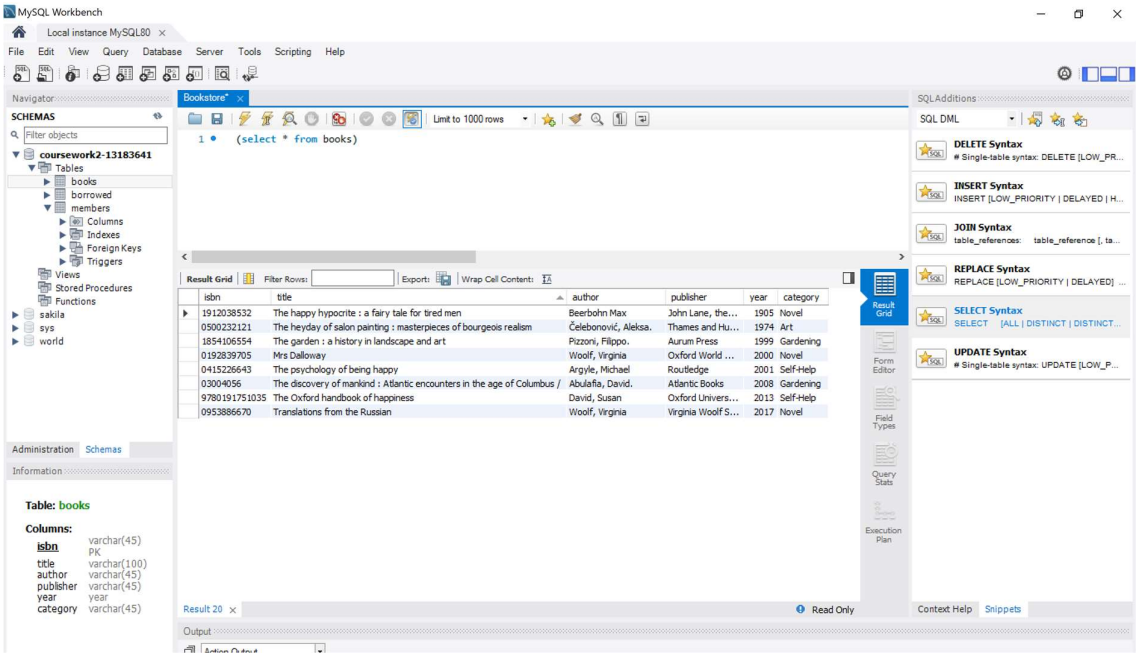
7. **Primary key for entity type.**
   - Institution: Name.
   - Award: Weak Entity. Hast to be referred to an Institution. To form the Key we have to add the Name of the Award and the Year. For example: BAFTA Best Actor in a Leading Role in 2019
   - Actor: Facebook Page (but note that in real life it would not be valid). IS A Entities derived from Actor inherit their primary key.
   - Movie: IMDb number is probably the best unique identifier (it can be equivalent to ISBN in books but IMDb is a private company owned by Amazon).
   - Roles / Script / Music: only can be uniquely identified by their link to a movie and an Actor/Writer/Musician.

8. **Add attributes and primary keys to the ERD**. Already done in the first image.

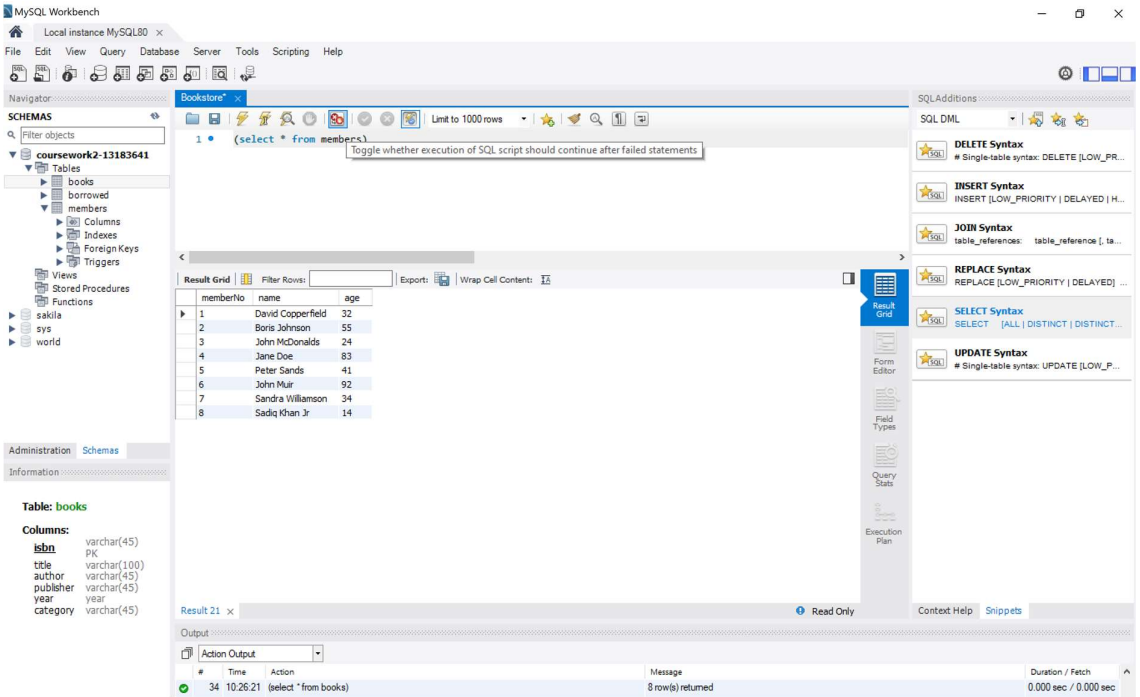
Database Management - 2020

Part II: SQL Queries (50 marks)

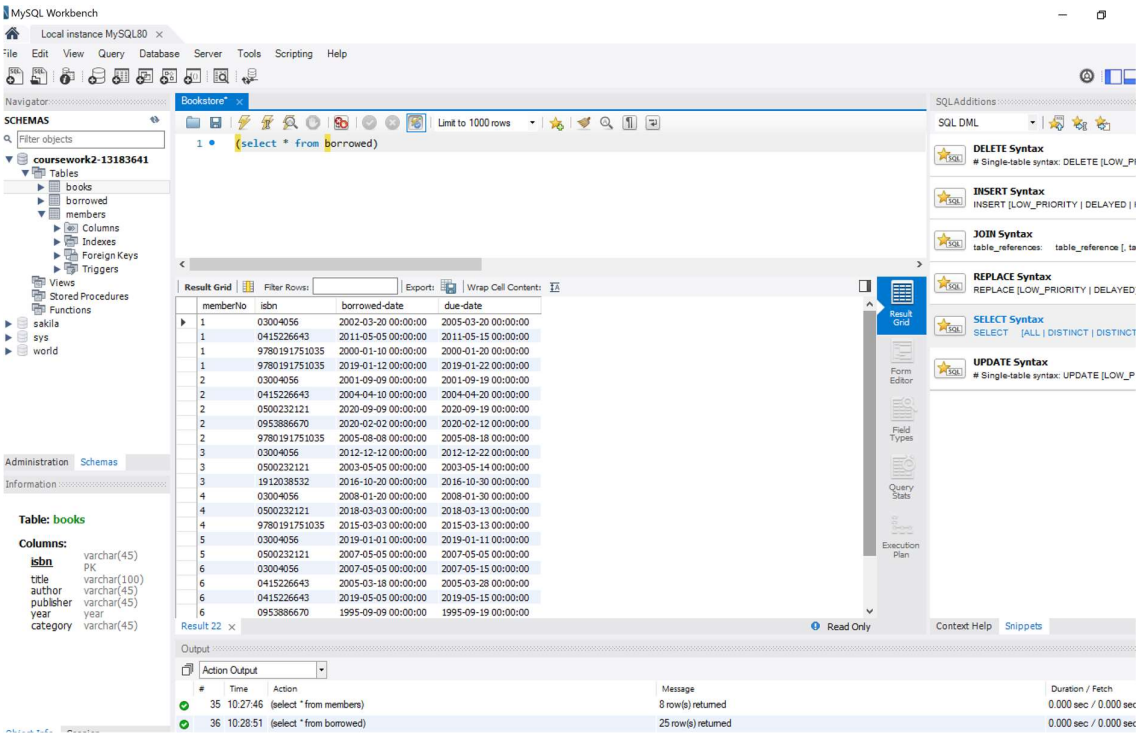
Books



Members

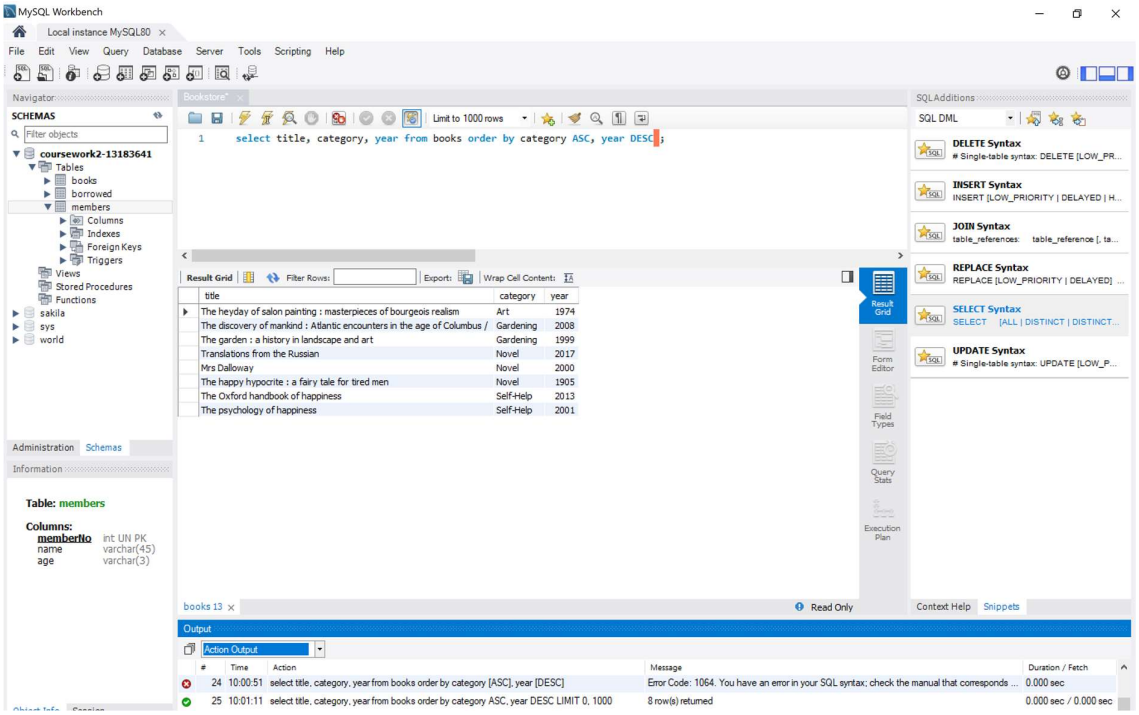


Borrowed



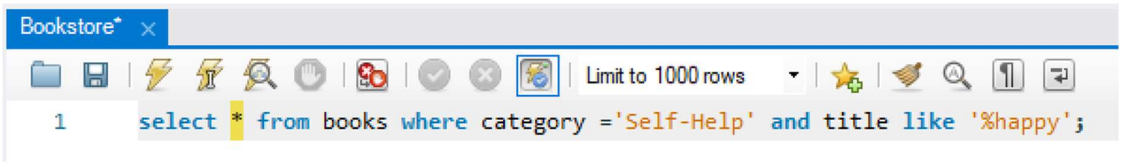
1. **List the title, category and year of publication of each book held in the library. The list should be ordered by ascending category, and within that by descending year of publication.**

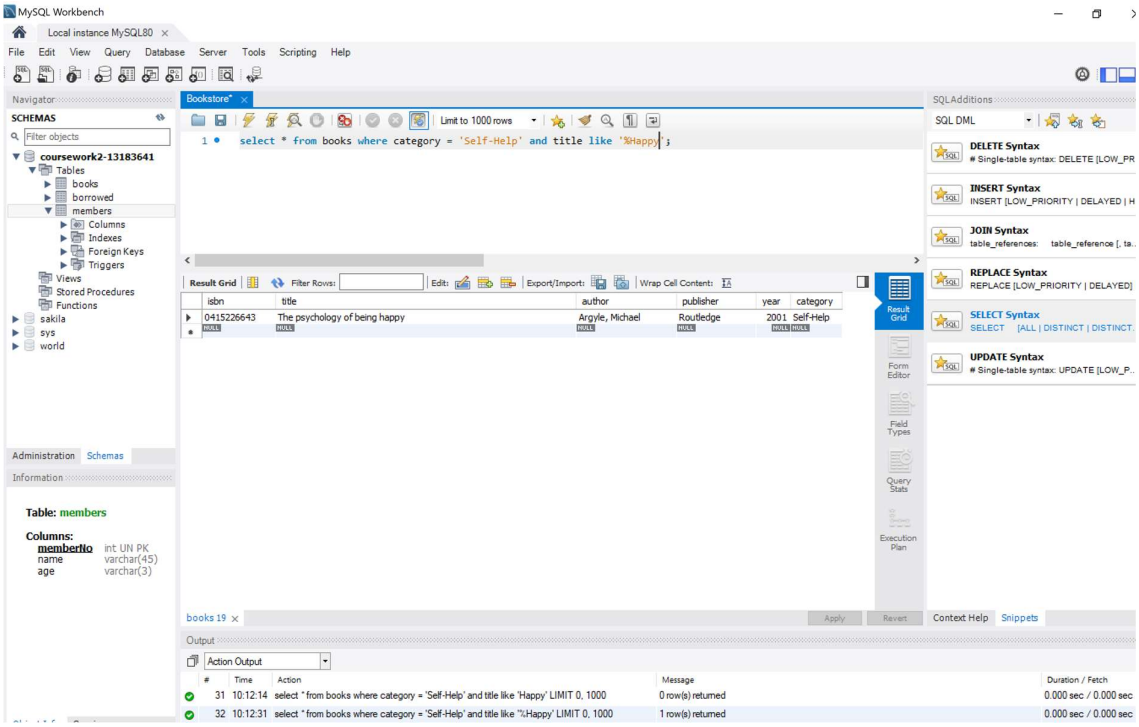
   Simple selection of 2 elements from one table (books) ordered firstly by category and year as second criterion.



2. Find the titles of books in the "Self Help" category that have "happy" somewhere in their title.

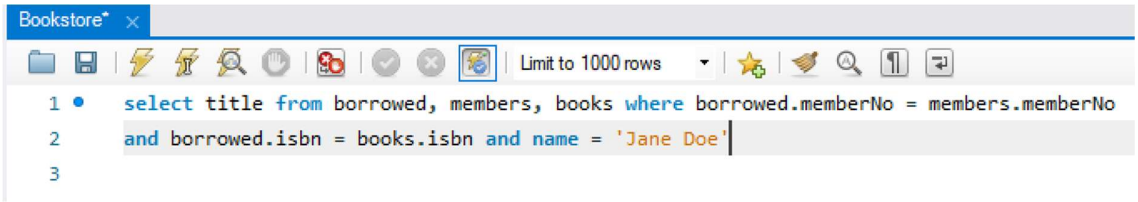
I select title (in the screenshot I have included * all but it is exactly the same) from one single table (books) and put 2 conditions linked by the AND operator and I introduce like to find strings that include the word happy.

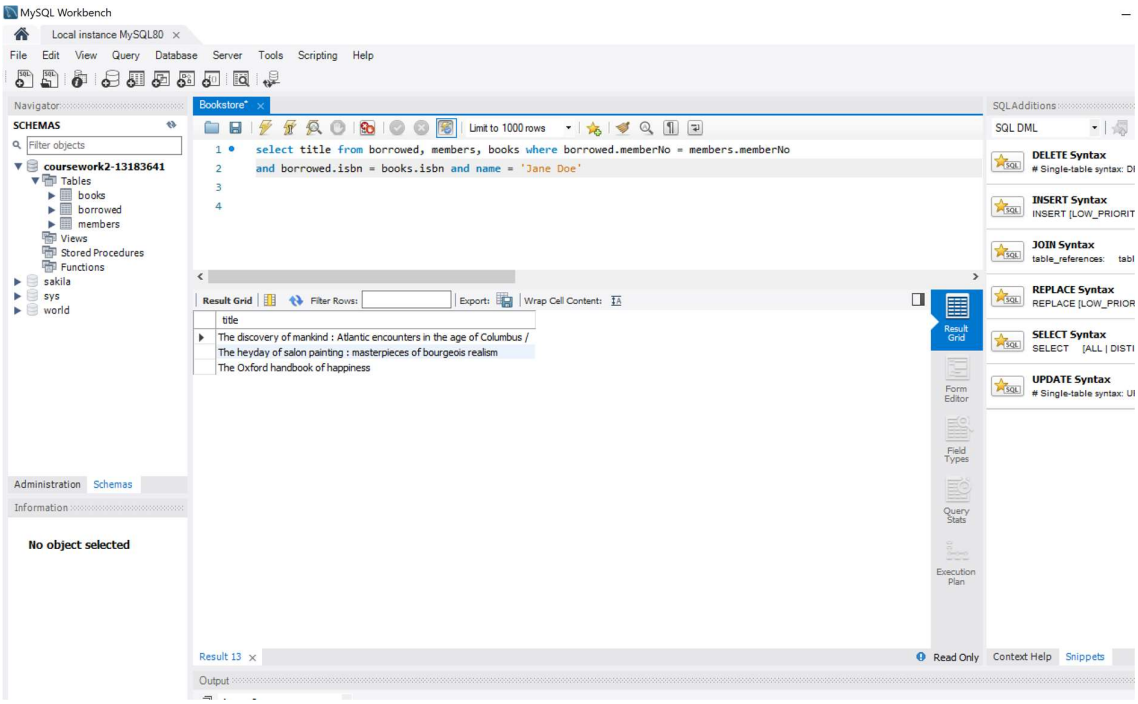




3. Find the titles of books borrowed by "Jane Doe".

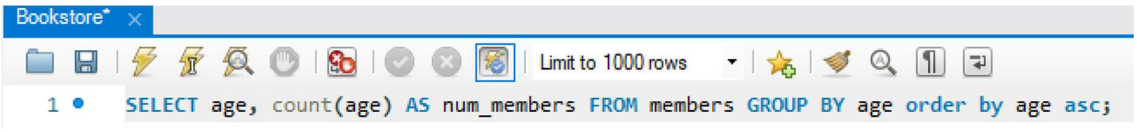
In this case I have to link the 3 main tables (books, members and operators) using members and isbn to discard duplicates and adding the additional condition that the name of the borrower has to equal the string ‘Jane Doe’.

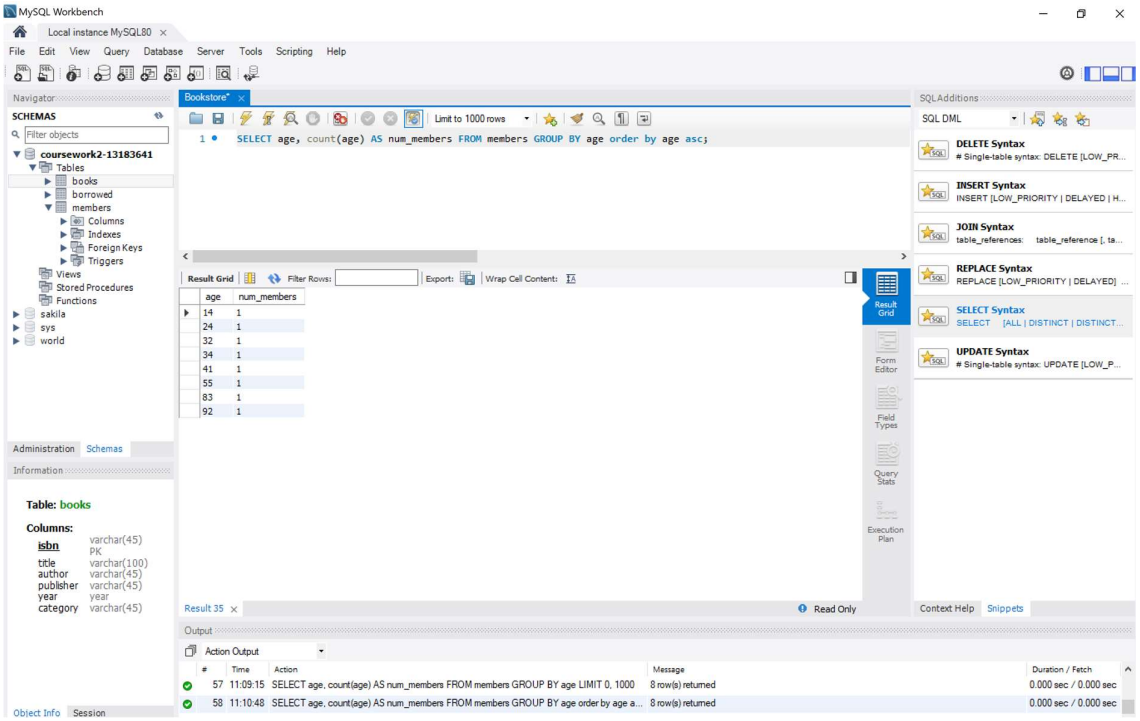




4. **Find the age profile of members, i.e., for each age, find the number of members of that age.**

   I use count () to consolidate data from members adding them by age. I rename the column AS num_members and finally, accordingly to our age consolidation I group the different ages and sort them in ascending order.

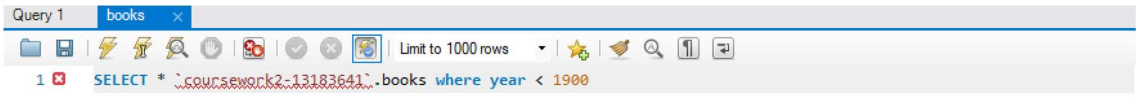


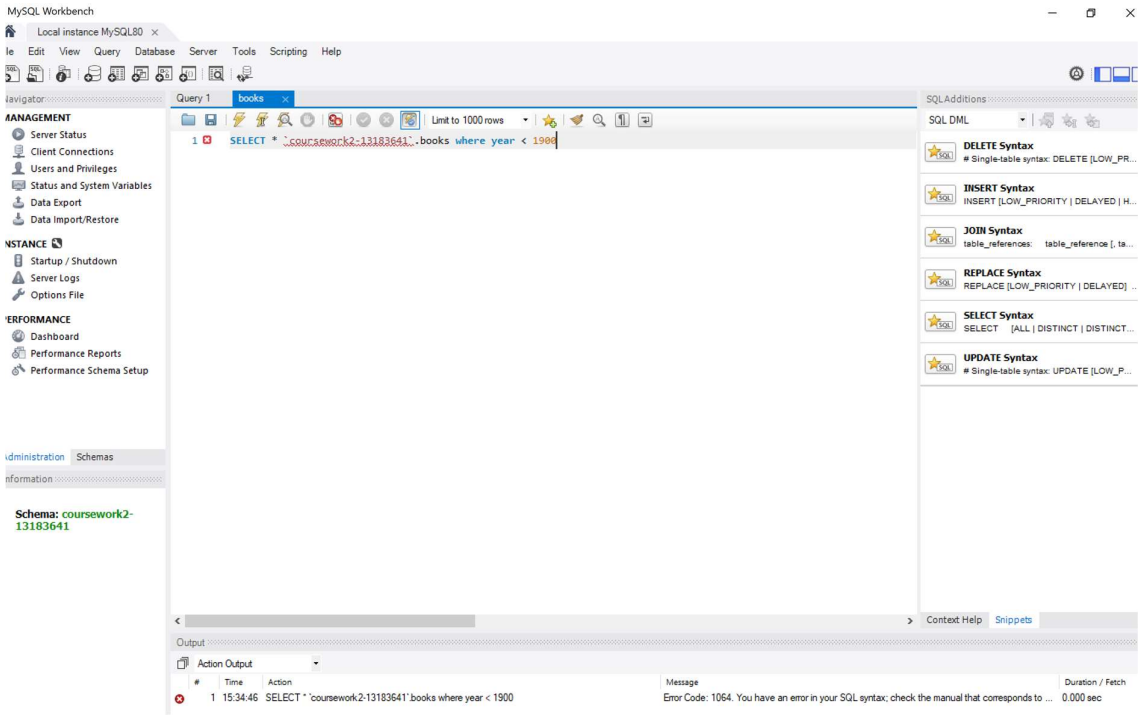


5. **Find the titles of books published before 1900.**

MySQL configuration does not allow selecting dates before 1900. I do not want to change Year configuration into varchar (it could have been a possibility). I should have changed the overall configuration of MySQL using date time instead of smalldate time.

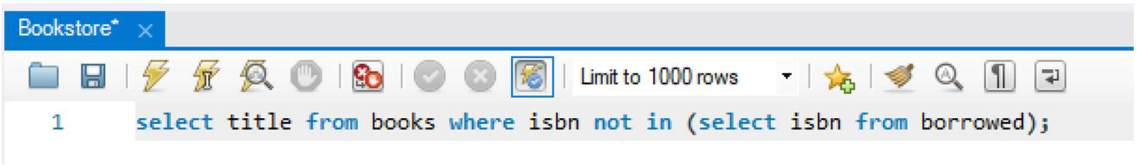
The query itself is not difficult at all: just select books in the DB and include the condition where year is less that 1900.

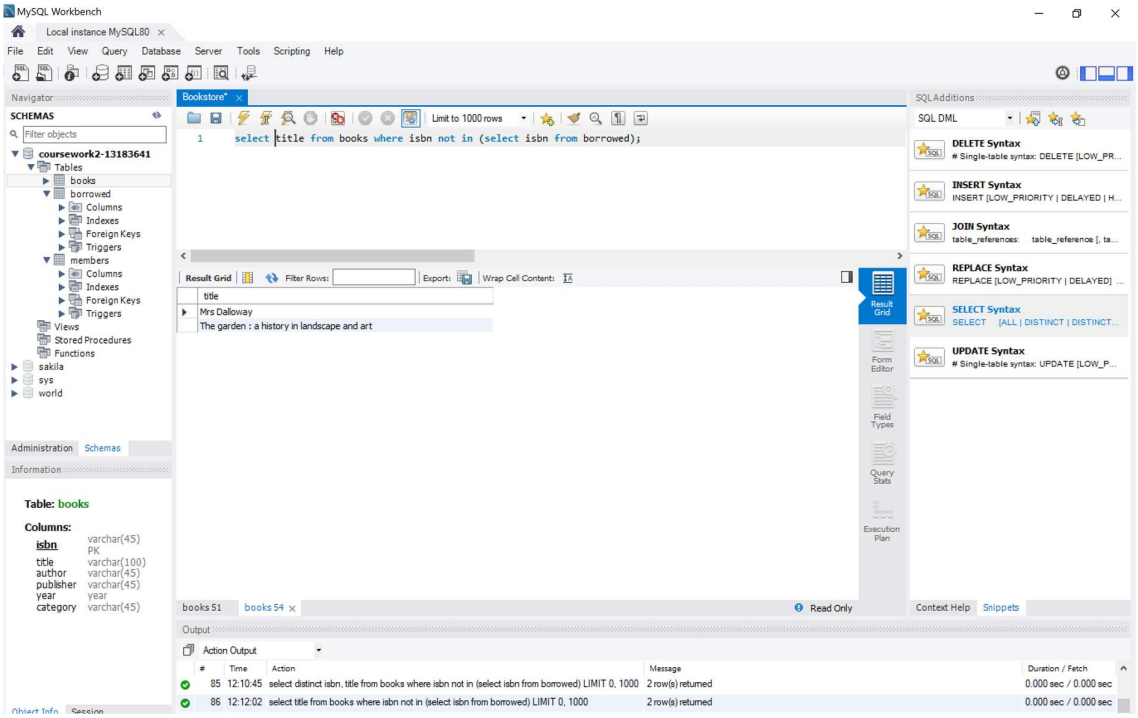




6. Find the titles of books that have never been borrowed.

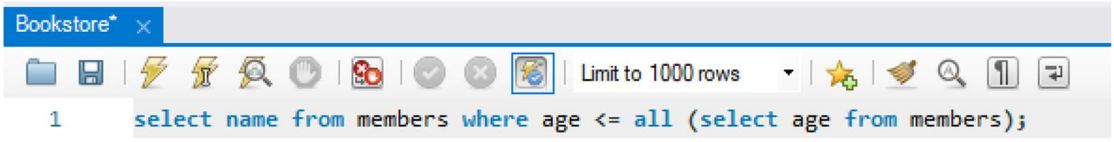
Books are identified by ISBN. I select the title from books table and add the condition that its isbn does not match any included in the borrowed tables using NOT IN.

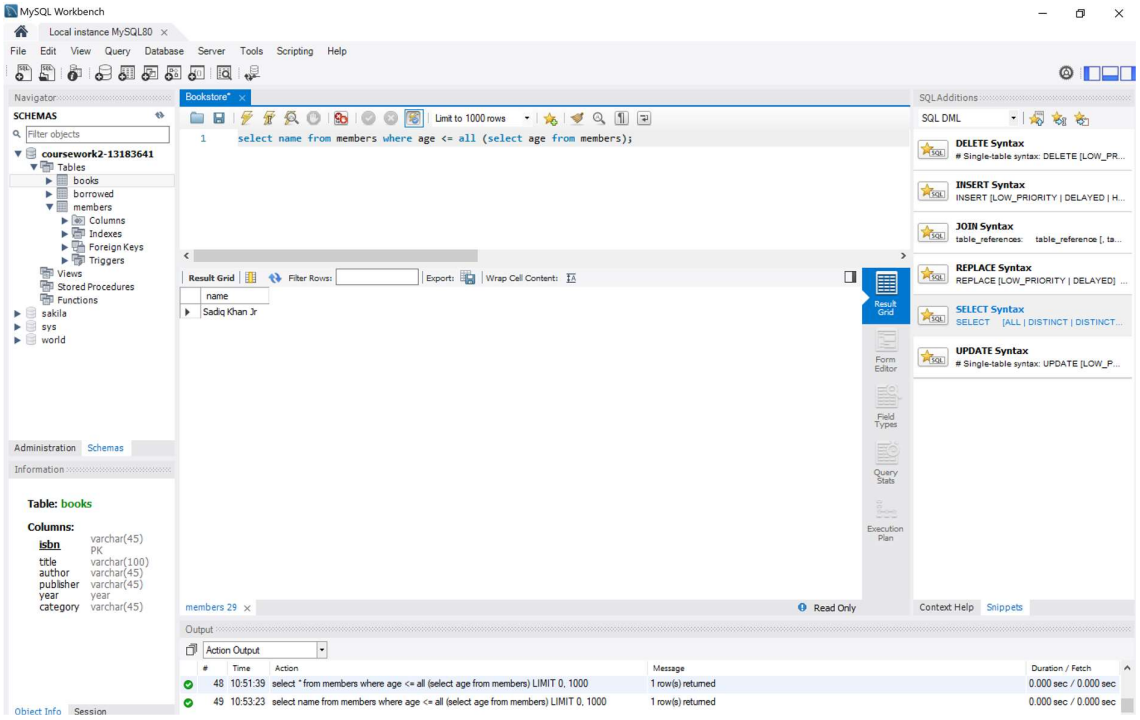




7. Find the name of the youngest member of the library.

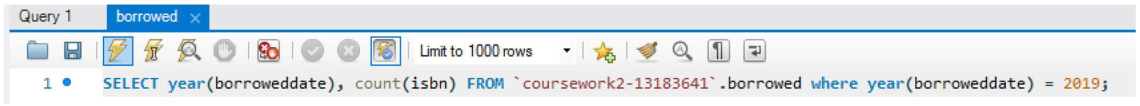
In this case I select column name from members table and add the condition that its age is the minor value of all ages from members using all and comparing to another query that includes ages from all members. As it is a very simple table there is only one member with that age.

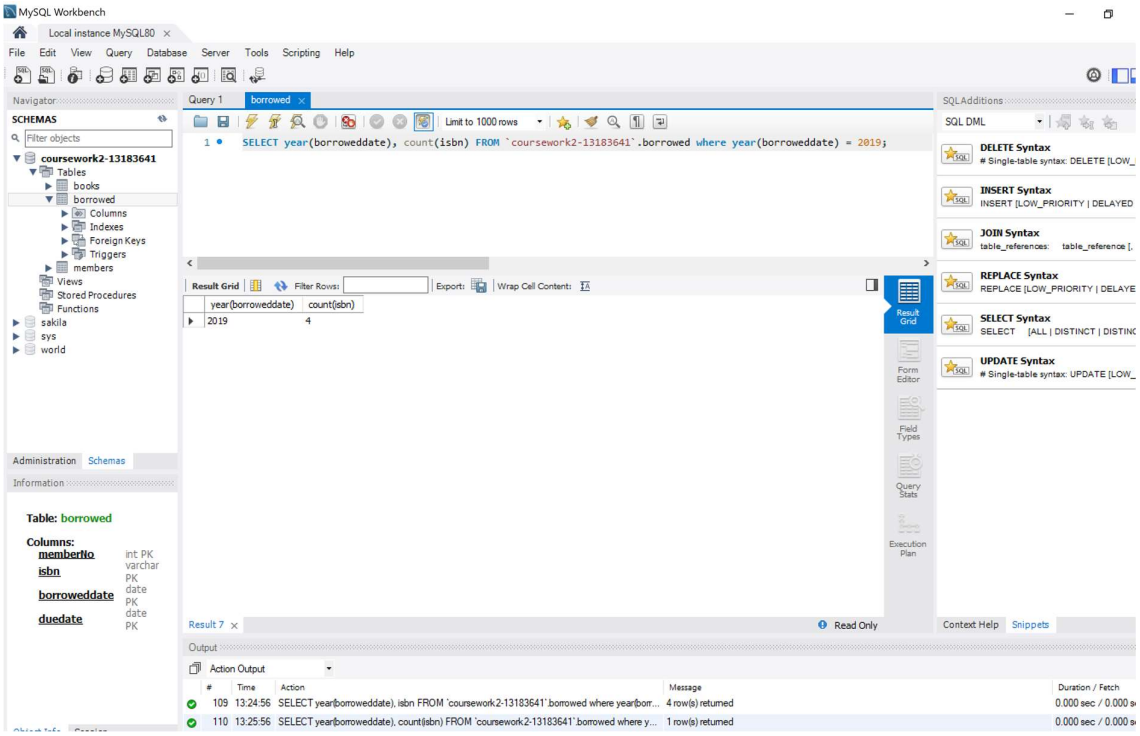




8. Find the total number of books borrowed in 2019.

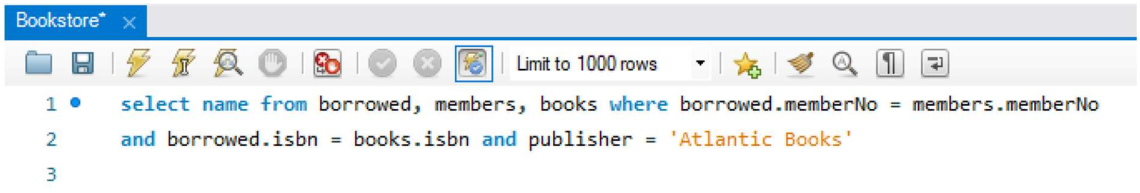
In this case I use the year() command to group all books based on borroweddate and count them by ISBN in the borrowed table. Finally I add the condition where the borrowed year equals 2019.

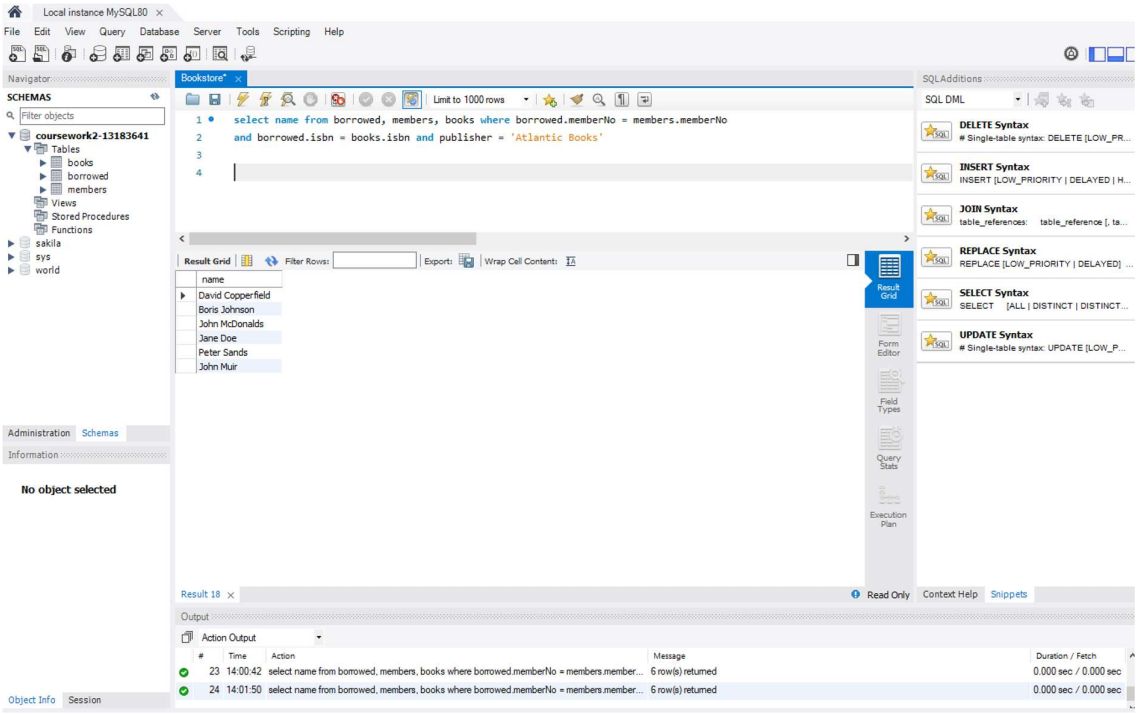




9. Find the names of members who have borrowed any book published by "Atlantic Books".

Here I am linking the 3 main tables using its main keys (memberNo and Isbn). Out of that data that does not have duplicates I chose the name and add the condition using AND that the publisher equals the string ‘Atlantic Books’.

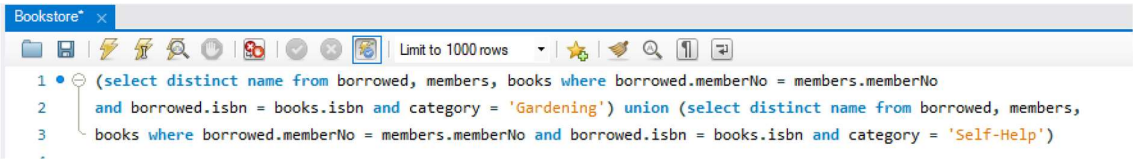


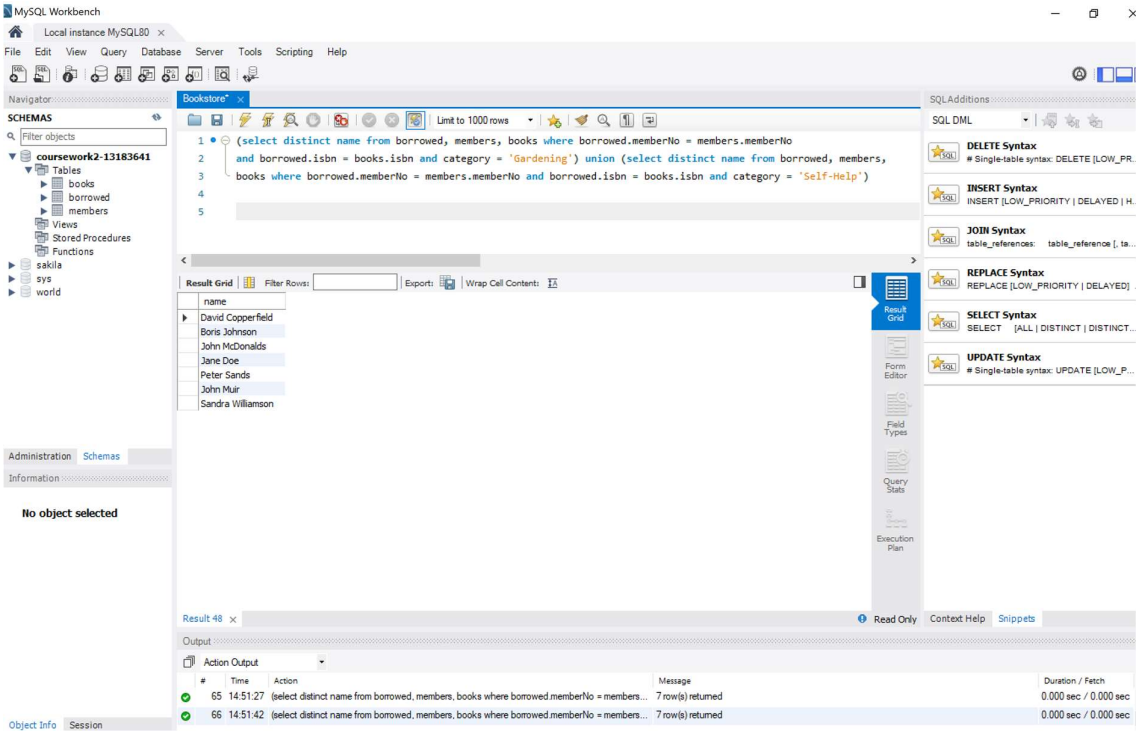


10. For each category, find the names of members who have borrowed more than five books in that category.

11. Find the names of members who have borrowed books from both of the categories "Gardening" and "Self Help".

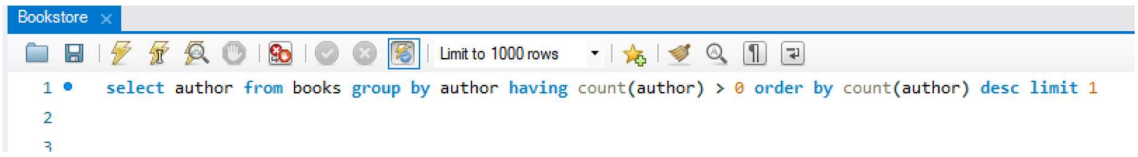
I use the Union to merge 2 different queries. Firstly, out of the 3 main tables linked I chose the category ‘Gardening’. Secondly, Using UNION I add another similar query where the books have been selected by category ‘Self-Help’. Union adds both querys.

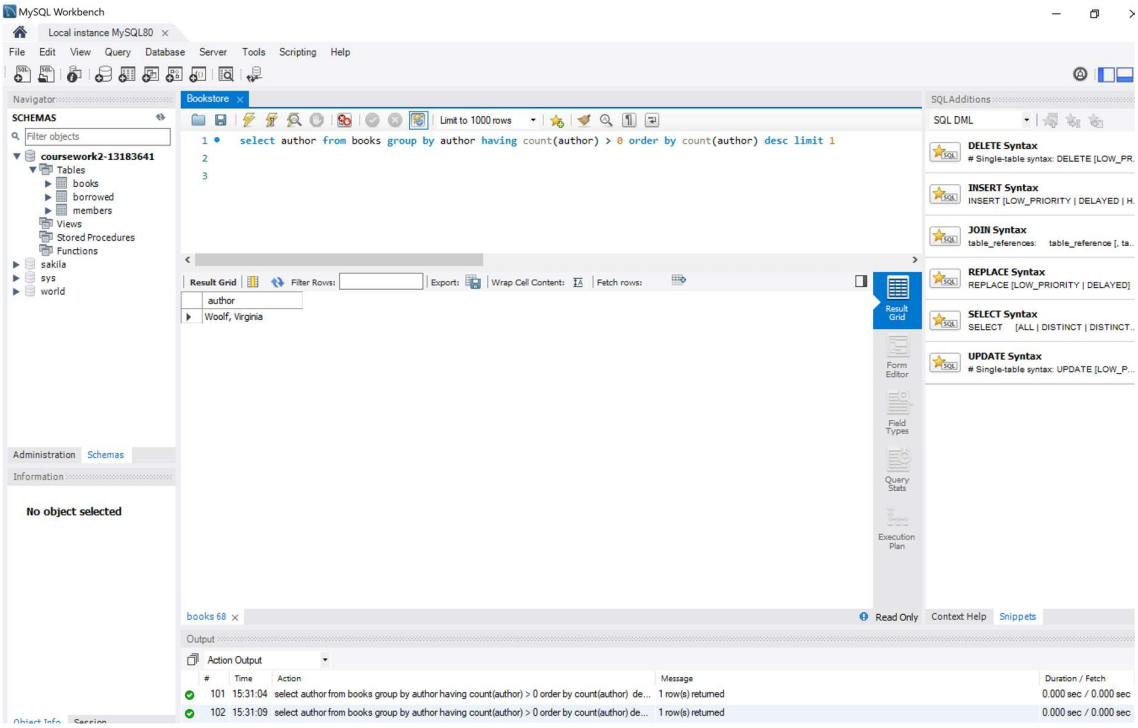




12. Find the most prolific author in the library, i.e., the author who has authored the most books.

Simple query extracting data from one single table. I group the books by author and add them using HAVING COUNT > 0. I sort the books in descending order and I use LIMIT 1 to pick the highest value.





13. Find the names of members who have borrowed the same book more than once.

Submit a pdf or text file containing a listing of the SQL queries and their results **(See attached files)**. Write a sentence or two on each query describing what you did.

Database Triggers.

1. What triggers are used for?

Triggers in Databases are used to automate a set of instructions to ensure that data consistency and integrity constraints are maintained across the DB whenever it is stored. It is used as well to automatize some tasks when some conditions are met. It is executed as a side effect if a database is modified. They are normally activated after a specific event, normally through a command issued in the Data Manipulation Language (DML) of the DB. As most DML use at least 4 basic commands (INSERT, SELECT, UPDATE and DELETE), the triggers can be used BEFORE, AFTER or INSTEAD OF these commands. It basically avoids erroneous entries and enables integrity in the DB.

Specific examples where triggers are used are: avoid invalid transactions (request validation before updating), enforce referential integrity in distributed DB, generate values automatically to populate some fields (for instance managing minimal inventory in a warehouse), prevent DML operations during certain times (e.g. out of business hours), enforce security rules log events in the DB, just to name some of them.

2. The general structure of a trigger.

The basic structure is:

a) Specify the moment when the trigger is fired. It can be decomposed into 2 steps:
   a. *An event* that activates the trigger.
   b. *Condition* to be met so the trigger is executed.
b) Actions to be executed when the trigger is fired.

Applied to a DBMS DML it is reflected in 3 steps:

a) Triggering SQL statement. As mention before, more common statements are INSERT, UPDATE, DELETE and additionally AFTER, BEFORE and INSTEAD OF the event that fires the trigger.
b) Trigger Restriction. Normally to set the frequency and scope where the trigger is applied.
c) Trigger Action. Performs a set of transactions as a consequence of the previous DML statement.

3. Description of different kinds of events and triggers.

In general, triggers can be classified in:

- *Row Level Trigger*. We have to specify when creating the trigger and its When condition.
- *Statement Level Trigger*. It is the default option. It will be executed once after the DML statement and it can be applied to one /several rows or the whole table.

Each major DBMS has its specificities. For instance, in Oracle there are 3 kinds of triggers:

- *DML triggers*. They have been formerly described: they run when a DML event is executed (normally Insert, Delete or Update). They run After, Before or Instead of when they are executed.

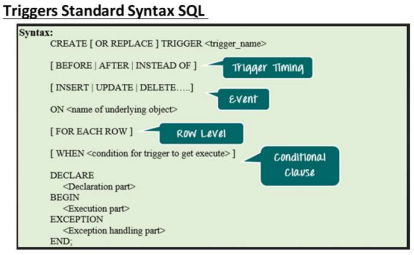
- *Schema triggers*. Fire on events that occur in objects defined within a schema, for instance: Alter, Analyse, Audit, Create, Drop, Grant, Rename, Revoke,…

- *Database triggers*: created on a Database and linked to events happening in that DB. Some examples are: After Startup, Before Shutdown, After Logon, Before Logoff, After Suspend,…

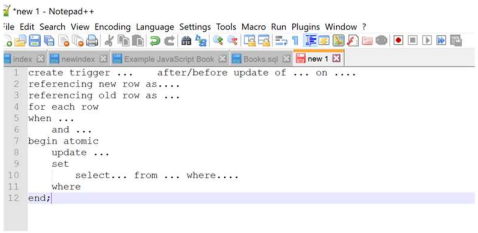
4. Databases supported by triggers.

All major DBMS support triggers but there are some differences over the SQL standard syntax. Below this standard (supported by IBM DB2 and MySQL with slight differences) is described. Other DBMS have a nonstandard syntax or may not implement all features. Below I describe some differences as well.

Triggers Standard Syntax SQL



IBM DB2 Example



Triggers in major DBMS

	Existence	Differences on SQL standard
Oracle	Yes	Keyword does not appear in referencing statement. Atomic does not appear after beguin Uses colon for the update statement. Subqueries are not allowed
Microsoft SQL Server	Yes	ON is used instead of AFTER Reference Clause is omitted Tuple used instead of New and Old rows
IBM DB2	Yes	Supports SQL standard
MySQL	Yes	Supports SQL standard

MySQL Example

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
       FOR EACH ROW
       BEGIN
           IF NEW.amount < 0 THEN
               SET NEW.amount = 0;
           ELSEIF NEW.amount > 100 THEN
               SET NEW.amount = 100;
           END IF;
       END;//
mysql> delimiter ;
```

5. Problems or disadvantages associated with triggers.

Triggers can be very untrusty and tricky due to the difficulties to follow the actions that they may fire. In general, it is always preferable to use relationships using foreign keys or appropriate use of stored procedures to enforce integrity rather than use to triggers. The main reason to avoid their usage is that triggers may fire other triggers, creating a 'trigger storm' with circular references.

New built-in solutions have been incorporated into new databases allow us to limit the length of these chains of triggers or to maintain replicas of databases or materialised views. They provide better solutions than to triggers.

Finally, triggers may be fired in other processes unintendedly like backups (some DB allow 'as not for replication' limits to avoid this).

In short, they can be very useful but it is always better to avoid them if proper alternatives exist.

6. References.

- *Melton, J. and Simon, A. (2002). SQL:1999. San Francisco, Calif.: Morgan Kaufmann.*
- *SILBERSCHATZ. (2019). DATABASE SYSTEM CONCEPTS: MCGRAW-HILL EDUCATION.*
- *Yarger, R., Reese, G. and King, T. (1999). MySQL & mSQL. Sebastopol (California): O'Reilly.*