Exercises on Relational Databases in MySQL Workbench

This document contains exercises on Relational Databases. Complete the tutorial yourself and answer the questions marked by a Q.

1. What you should already know: relational databases

A relational data model has data that is represented by tables. We will use Structured Query Language (SQL) as the query language to manipulate the data of the database. Some main operations used for querying data are “restrict”, “project”, “group”, and “join”. Q: What does the output of an operator look like?

The (general) syntax of the SQL queries looks as follows:

```
SELECT column_name(s)

FROM table_name

WHERE condition

GROUP BY column_name(s)

HAVING condition
```

The JOIN operator is used when you want to retrieve data from multiple tables.

Aggregate functions are, for example, count(), avg(), sum(), min() and max().

For more information on all operators and functions, visit the following website:
https://www.w3schools.com/SQL/default.asp

2. Why MySQL?

For the exercises we will use MySQL as a relational database management system (RDMS). This database is open-source and and is available on over 20 platforms (including Windows, iOS, Unix and Linux). Other examples of relational database management systems include Microsoft SQL Server, Microsoft Access, Oracle, DB2, and so on.

In order to connect with MySQL, we need to use a server access tool that is able to communicate with MySQL Server, which is at core of MySQL. This server handles all database instructions (commands) and is available as a separate program for use in a client-server networked environment[1] and as a library that can be embedded into separate applications. Via a MySQL "client", installed on a computer, the commands are sent to MySQL Server. The server provides a query interface (SQL) to persist and manipulate the data. A "client" allows you to use that interface. There are multiple "clients" for MySQL. The one we will use is MySQL Workbench.

MySQL Workbench is a graphical tool or "client" program for working with and connecting to MySQL servers and databases. MySQL Workbench has tools to visually create physical database design models that can be translated to MySQL databases by forward engineering. We will use MySQL Workbench as a client application in this course. MySQL Workbench has a visual SQL editor and for this reason, SQL scripts can be run easily. (Note that MySQL is not case sensitive for the SQL keywords, e.g. "SELECT" and "select" are both possible as query syntax.)

3. MySQL Workbench Building Blocks

In the following sections of this tutorial, we first show you how to install a MySQL Server and the Workbench application tool. Next we teach you how to build a logical representation of a relational database using an ER diagram (see **Fig. 1**). This logical representation can be used to physically create the database. The resulting schema is basically the definition of all table objects, column types, relationships between the tables, triggers, functions, constraints, keys and indices. Third, we will teach you how to create a MySQL Schema in MySQL Workbench. A "schema" is a collection of database objects (e.g., tables for relational databases). Basically, in MySQL, a "schema" is a collection of tables that contain the physical data. One database can have multiple database schemas, which in turn contain multiple tables. Note that the same object (e.g., a table named "table1") can be used without conflict by different schemas. Schemas are not rigidly separated: a user can have access to objects in any of the schemas in the database they are connected to as long as they have the priviliges to do so. Schemas specify which fields will be present in the database and what will be their data types (e.g., table "*employee*" will have an "*employee_ID*" column represented by a string of 10 digits).

[1] A client-server network is designed for "clients" or "end-users" to access files and so on from another central computer called a "server". A client uses the "network" to connect with and speak to the server.

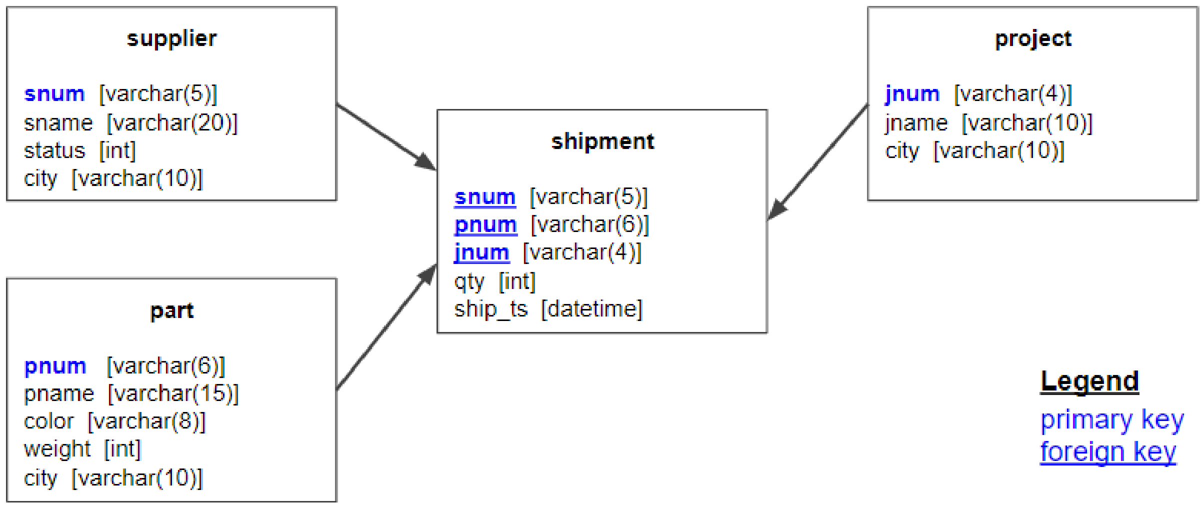


Fig. 1: Example of a **SPJ database** an Entity-Relationship diagram. Entities are "supplier", "part", "shipment" and "project". Primary keys are indicated in bold and foreign keys in bold/underlined.

The **main MySQL Workbench building blocks** are:

- **MySQL Server**: The core of MySQL is the MySQL Server. This server handles all database instructions (i.e., commands) and is available as a separate program for use in a client-server networked environment and as a library that can be embedded into separate applications. A client program (MySQL Workbench) is used to connect to MySQL Server and to use the query interface.
- **MySQL Connection**: links MySQL Workbench application to a MySQL Server. Actions (e.g., SQL queries) within Workbench are then performed against a connected MySQL Server.
- **MySQL Model**: MySQL models are used to build ER diagrams and physical MySQL databases or "schemas" (with "real" data in the tables).
- **MySQL Schema**: a "schema" or "database" is the physical implementation of a logical model in MySQL Workbench. You can create a schema by forward engineering (creating schema from MySQL Model) or immediately.
- **Database design**:
  - "Logical" model: Develop database model based on the requirements. There are no physical implementations. For example, creating an Entity-Relationship Model (ER) helps to model entities/attributes and relationships.
  - "Physical" model: The logical model is implemented and physical tables containing real data are created.

4. Getting started: Installation of MySQL Server & Workbench

This section contains guidelines for installing MySQL Server & Workbench on your own computer (to remake exercises or make new exercises at home). Note that the Server and Workbench are already installed on the computers in the computer lab. **You can skip subsection 4.1 in the tutorial session and immediately use the pre-installed LocalServer.**

4.1 Download MySQL Server (skip during tutorial session):

Download MySQL Server (latest version 8.0.18) via MySQL Installer. Make sure that you select the right operating system (iOS, Windows, Linux).

Step 1: Choose as setup type "Developer Default"

Step 2: Installation. The MySQL installer will install all MySQL products, including the MySQL Workbench. Click "Execute".

Step 3: Product configuration. Click "Next".

Step 4: High availability. Choose "standalone MySQL Server". Configuration type: select "Development Computer". As connectivity, choose "TCP/IP", Port: 3306 and X Protocol Port: 33060. Click "Next".

Step 5: Authentication method. Click "Next".

Step 6: Accounts and Roles. Choose a MySQL Root Password for the root account.

Step 7: Choose MySQL80 as Windows Server name. Click "Next".

Step 8: Apply configuration: click "Execute".

Step 9: Connect to server.

4.2 Connect to a MySQL connection

Launch the MySQL Workbench application. The Home Window opens up on your screen.

Go to "MySQL Connections" and double click on "My Server". Enter the password. In the computer lab, you can use the MySQL server which is available when you open the Workbench. However, this server will only be pre-installed in the lab P.010. Double click on the **LocalServer** and enter the **password "SP010user"**. You are now able to start working on this server.

4.3 Visual SQL editor

The visual SQL editor window opens up once you double clicked on a connection.

To create tables, you will need to execute **SQL scripts**. First, take a look at the visual SQL editor. This editor consists of a set of editors: query, schema, table, and so on. All of these panels allow you to build/edit/run queries (in SQL), create/edit data, view/export results (e.g., to a csv file). When writing code in SQL, you will be assisted by color syntax highlighting, context help, code completion, and so on.

The screenshot in **Fig. 3** shows you how the visual editor looks like in Workbench:

1: Home screen tab;

2: Connection tab: each separate connection made to the server is represented by a different tab;

3: SQL query tab;

4: Main menu bar;

5: Main toolbar: create SQL tab for new queries, create SQL script file in a new query tab;

6: Shortcut actions;

7: Sidebar panel with Navigator and Information labels. The Navigator shows the Schemas (or databases);

8: SQL Additions area.

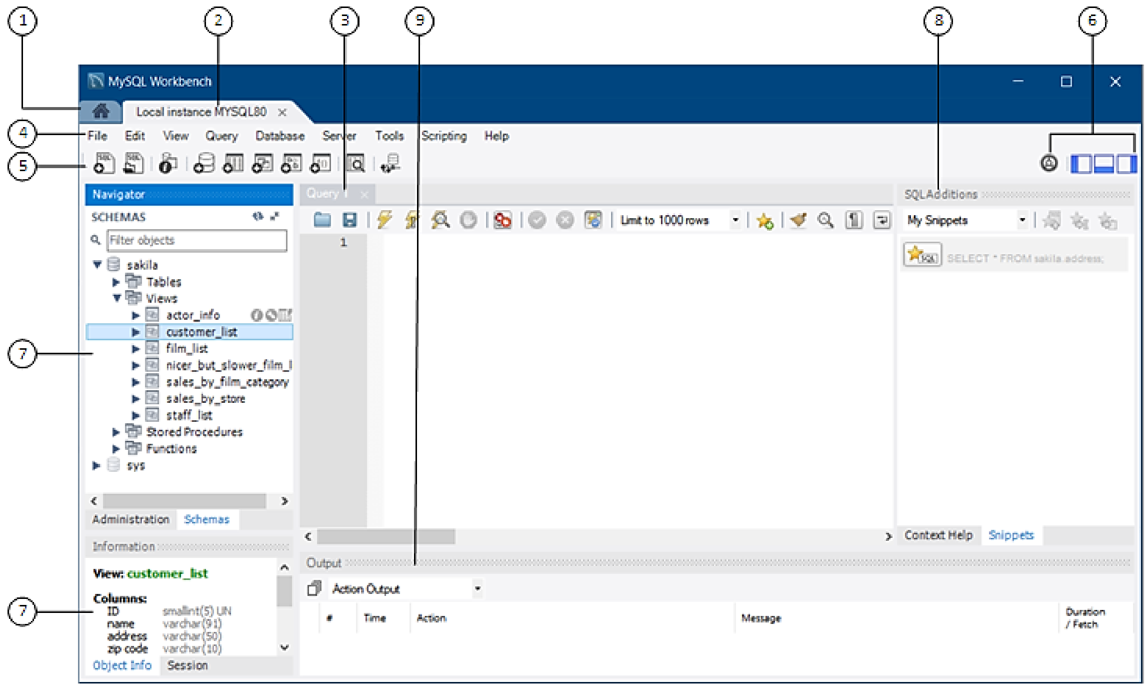
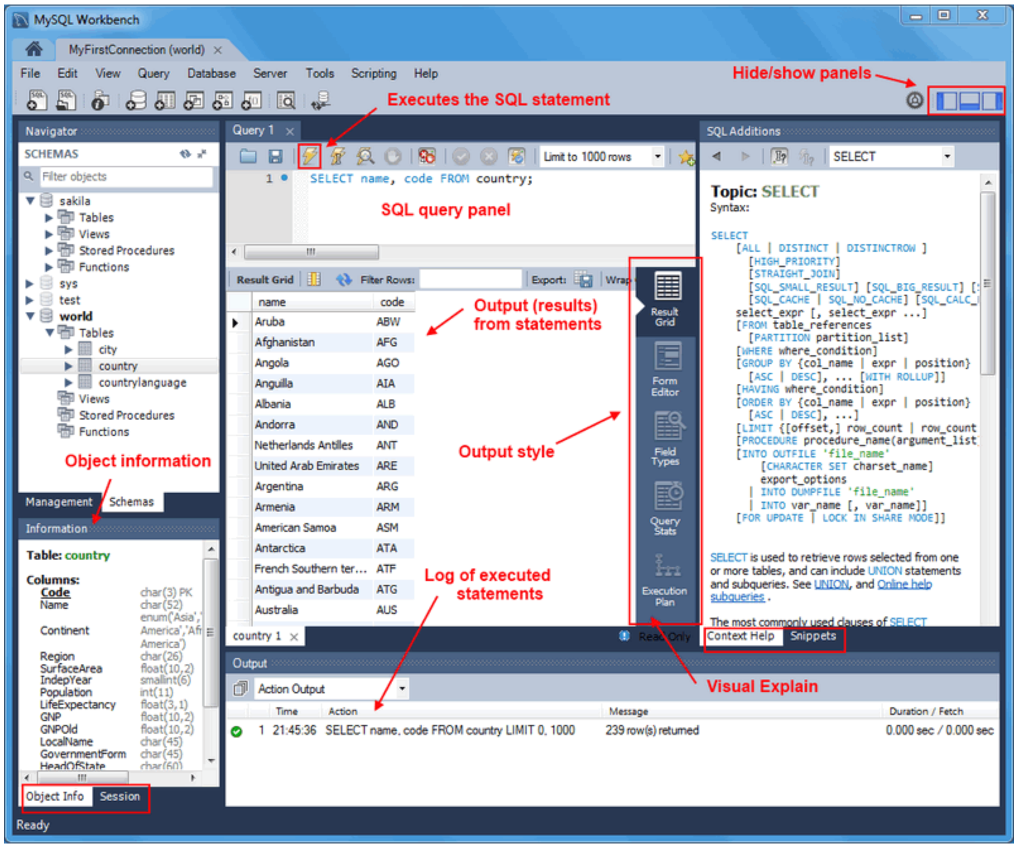


Fig. 3: Visual editor in MySQL Workbench.



```
SELECT name, code FROM country;
```

name	code
Aruba	ABW
Afghanistan	AFG
Angola	AGO
Anguilla	AIA
Albania	ALB
Andorra	AND
Netherlands Antilles	ANT
United Arab Emirates	ARE
Argentina	ARG
Armenia	ARM
American Samoa	ASM
Antarctica	ATA
French Southern ter...	ATF
Antigua and Barbuda	ATG
Australia	AUS

Table: country

Columns:

Code	char(3) PK
Name	char(52)
Continent	enum('Asia',' America','Afr America')
Region	char(26)
SurfaceArea	float(10,2)
IndepYear	smallint(6)
Population	int(11)
LifeExpectancy	float(3,1)
GNP	float(10,2)
GNPOld	float(10,2)
LocalName	char(45)
GovernmentForm	char(45)
HeadOfState	char(60)

Topic: SELECT
Syntax:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count
    [PROCEDURE procedure_name(argument_list
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT is used to retrieve rows selected from one or more tables, and can include UNION statements and subqueries. See UNION, and Online help subqueries .

The most commonly used clauses of SELECT

```
     Time      Action                                        Message               Duration / Fetch
✓ 1  21:45:36  SELECT name, code FROM country LIMIT 0, 1000  239 row(s) returned   0.000 sec / 0.000 sec
```

Fig. 4: SQL Query Tab.

Let's take a look at the "SQL Query" tab (see **Fig. 4**), which you will need to execute queries. The most important elemens in a query tab are the query panel and output panel. You can save snippets of SQL code by clicking on "⭐" from the SQL query toolbar.

4.4 Logical database design model: Entity-Relationship (ER) diagram

Entity-relationship (ER) diagrams can be used to design relational databases. They are "logical" database design models.

To illustrate how ER diagrams are created to design relational databases, consider the following case study of "MyFlix" (see full tutorial[2] for more information), which is a business that rents out movies to people. A database management system is created to store and manage movie records.

[2] The "MyFlix" example case study of relational database management system is part of the following tutorial on https://www.guru99.com/er-modeling.html

Entities are real-world things and can be conceptual (e.g. sales orders). In this example, the entities to be included are:

- Members with member information
- Movies with movie information
- Categories with information on movie categories
- Movie Rentals with info on rented movies to members
- Payments with payment information

Relationship information between the entities is:

- A member can rent more than one movie in the same period
- A movie can be rented by more than one member in a given period
- A movie can only belong to one category
- Categories can have more than one movie
- A member can only have one account
- A member can make a number of payments

To create an entity-relationship diagram in MySQL Workbench, click on "**+**" next to "*Model*":

There are two important icons you need to work with to create the ER diagram:

- The Table icon "▦" creates entities and attributes associated with the entities
- The Relationship icons allow to create relationships between entities. There are 1:1, 1:n, n:m relationships possible.

To create the table "*members*", click on the "*Table*" icon and drag the icon into the "*Tools*" panel. Double click on the Table and the "*Properties*" window appears. Change the name of the entity to "*members*" and add the following attributes as columns:

- Membership number "*membership_number*" (Primary key, INT)
- Full names "*name*" (VARCHAR(45))
- Gender "*gender*" (NCHAR(1))
- Date of birth "*birth*" (Date)
- Physical address "*physic_ad*" (VARCHAR(45))
- Postal address "*post_ad*" (VARCHAR(45)))

Make sure you select the right data type for each of the attributes. Select the correct primary key for this entity.

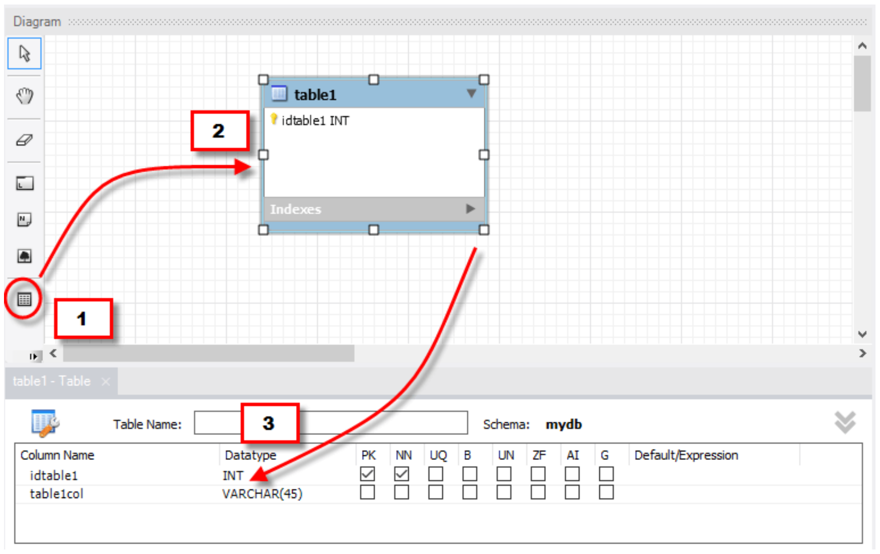


Fig. 5: Screenshot components ER diagram.

Create another entity "*movierentals*" and add the following attributes:

- Reference number "*ref_id*"
- Transaction date "*trans_date*"
- Return date "*return_date*"

Now, create a relationship between the "*members*" and "*movierentals*" entities by clicking on the relationship icon and then clicking on the "*memb_id*" and "*ref_id*" attributes of each entity. The relationships are created and immediately, foreign keys appear in the entity. Q: which direction of the relationship do you think is most appropriate?

Go to the MyServer tab and open the "MyFlix.mwb" model in MySQL Workbench. Double click on the ER diagram. The ER diagram is shown in **Fig. 6**.

The ER diagram is a "logical" model of our database and, of course, there is no physical database created yet. To go from the ER model to a physical implementation of the database, you can use forward engineering. Forward engineering is a process to translate logical models into physical models automatically. To automatically generate the SQL scripts to go from logical database model to the physical database model, visit the following website[3]. In contrast, reverse engineering refers to the process of reconstructing a logical data model from a physical data model. More information on reverse engineering an existing database to extract the model can be found on the following website[4]. (This is just additional information, and we encourage you to try this at home).

[3] https://www.guru99.com/how-to-create-a-database.html
[4] https://www.techotopia.com/index.php/Using_MySQL_Workbench_to_Create_a_Database_Model

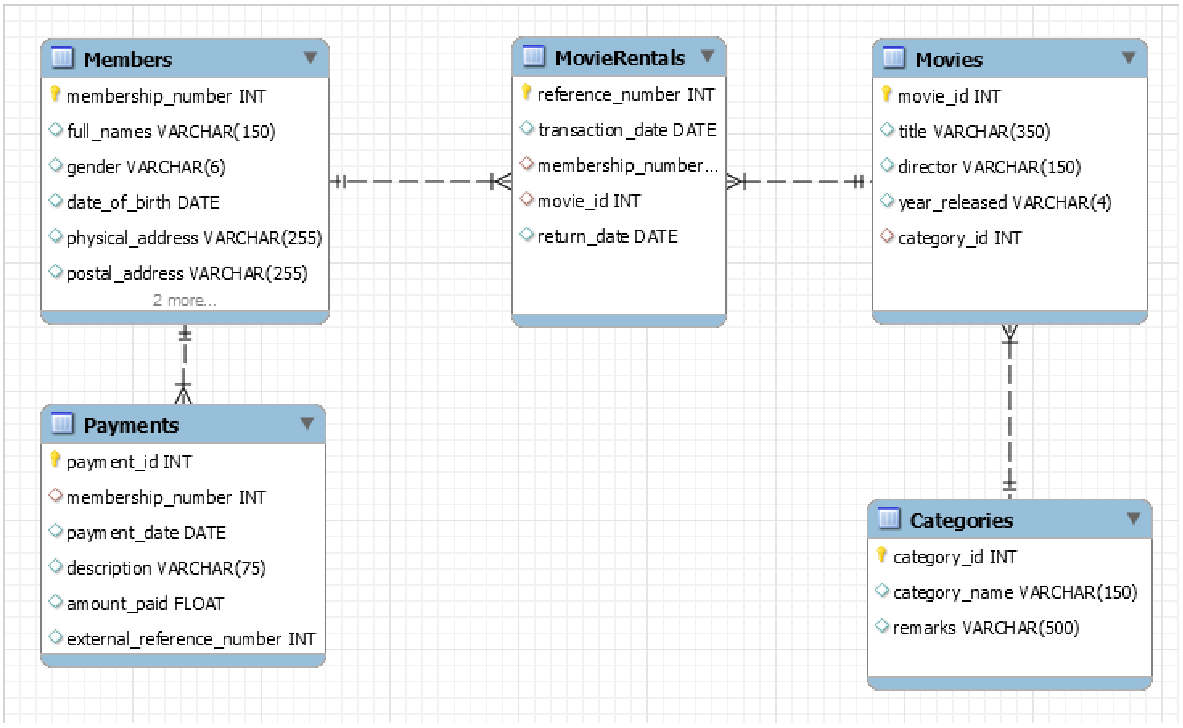


Fig. 6: "MyFlix" Entity-Relationship diagram.

4.5 Databases (MySQL Schemas)

Logical database models (ER diagrams) can be translated into physical ones by forward engineering. MySQL Workbench is able to generate SQL scripts from the ER diagram that can be executed to create a physical database (the actual tables with data in it). Note that in MySQL a "database" is synonym for "schema".

The "CREATE DATABASE" command can be used to create databases, while "CREATE TABLE" can be used to create tables in a database. Note again that MySQL Workbench is not case sensitive.

4.6 SQL scripts: create MySQL schema and insert data

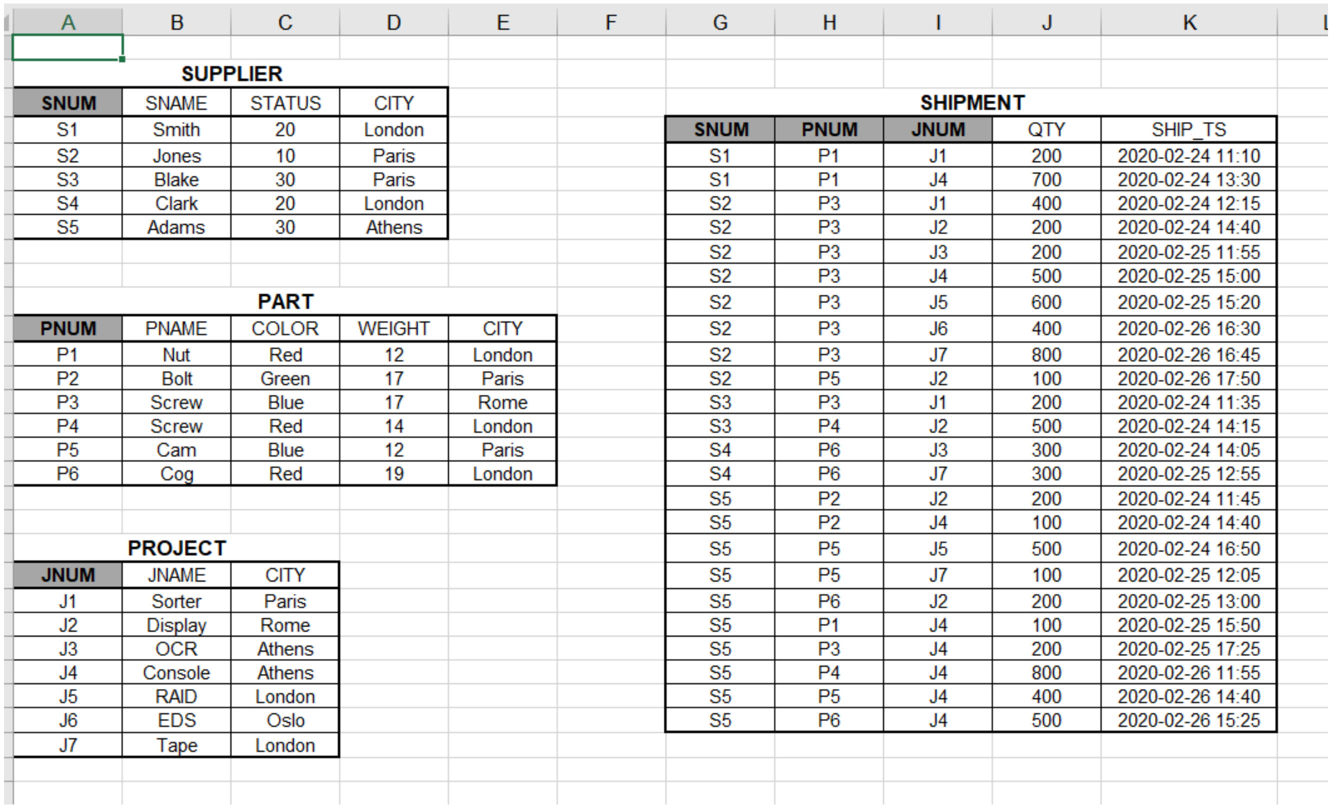
We will create a database called **SPJ (Supplier Parts Projects)**, which is a simple example database made available by Date (2004) to illustrate relational databases (more info on the database). The database contains four different tables:

- Table "**supplier**": information about suppliers. The snum identifies the supplier, while the other attributes each hold one piece of information about the supplier.
- Table "**part**": information about the parts. The pnum identifies the parts, while the other attributes each hold one piece of information about the part.

- Table “**project**”: information about the projects. The jnum identifies the projects, while the other attributes each hold one piece of information about the project for which the part is used.
- Table “**shipment**”: information about the shipments. The snum, pnum and jnum identify each shipment, while the remaining attribute indicates how many parts were shipped. It is assumed that only one shipment exists for each supplier/part/project pairing (which is not very realistic in real-world scenarios).

Go to “File” → “Open SQL Script” and open both the SQL scripts “create_spj_tables” and “insert_spj_data”.

To create the tables, first go to the SQL script “create_spj_tables” and click on the “ ” icon to run the script. The tables are now created. Next, run the SQL script “insert_spj_data” to insert data into the tables. The following tables will be created (see SPJ_DB.xlsx):



SUPPLIER

SNUM	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

PART

PNUM	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

PROJECT

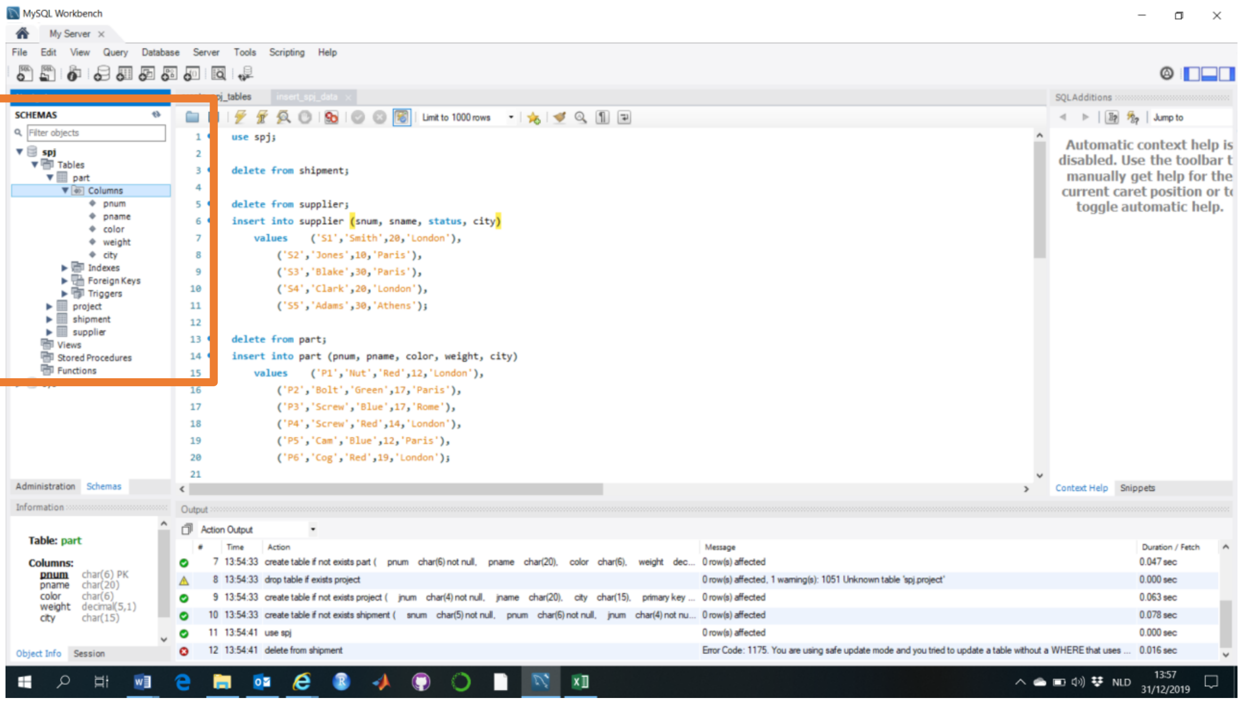
JNUM	JNAME	CITY
J1	Sorter	Paris
J2	Display	Rome
J3	OCR	Athens
J4	Console	Athens
J5	RAID	London
J6	EDS	Oslo
J7	Tape	London

SHIPMENT

SNUM	PNUM	JNUM	QTY	SHIP_TS
S1	P1	J1	200	2020-02-24 11:10
S1	P1	J4	700	2020-02-24 13:30
S2	P3	J1	400	2020-02-24 12:15
S2	P3	J2	200	2020-02-24 14:40
S2	P3	J3	200	2020-02-25 11:55
S2	P3	J4	500	2020-02-25 15:00
S2	P3	J5	600	2020-02-25 15:20
S2	P3	J6	400	2020-02-26 16:30
S2	P3	J7	800	2020-02-26 16:45
S2	P5	J2	100	2020-02-26 17:50
S3	P3	J1	200	2020-02-24 11:35
S3	P4	J2	500	2020-02-24 14:15
S4	P6	J3	300	2020-02-24 14:05
S4	P6	J7	300	2020-02-25 12:55
S5	P2	J2	200	2020-02-24 11:45
S5	P2	J4	100	2020-02-24 14:40
S5	P5	J5	500	2020-02-24 16:50
S5	P5	J7	100	2020-02-25 12:05
S5	P6	J2	200	2020-02-25 13:00
S5	P1	J4	100	2020-02-25 15:50
S5	P3	J4	200	2020-02-25 17:25
S5	P4	J4	800	2020-02-26 11:55
S5	P5	J4	400	2020-02-26 14:40
S5	P6	J4	500	2020-02-26 15:25

Fig. 7: SPJ_DB.xlsx screenshot.

Click on “Reconnect to DBMS” icon ( ) to refresh. Now, on the left side of the window, you see the created database called “spj” (under “SCHEMAS”). You can navigate through the Tables, Views, Stored Procedures and Functions of this database.



```
use spj;

delete from shipment;

delete from supplier;
insert into supplier (snum, sname, status, city)
    values    ('S1','Smith',20,'London'),
        ('S2','Jones',10,'Paris'),
        ('S3','Blake',30,'Paris'),
        ('S4','Clark',20,'London'),
        ('S5','Adams',30,'Athens');

delete from part;
insert into part (pnum, pname, color, weight, city)
    values    ('P1','Nut','Red',12,'London'),
        ('P2','Bolt','Green',17,'Paris'),
        ('P3','Screw','Blue',17,'Rome'),
        ('P4','Screw','Red',14,'London'),
        ('P5','Cam','Blue',12,'Paris'),
        ('P6','Cog','Red',19,'London');
```

```
Table: part

Columns:
  pnum     char(6) PK
  pname    char(20)
  color    char(6)
  weight   decimal(5,1)
  city     char(15)
```

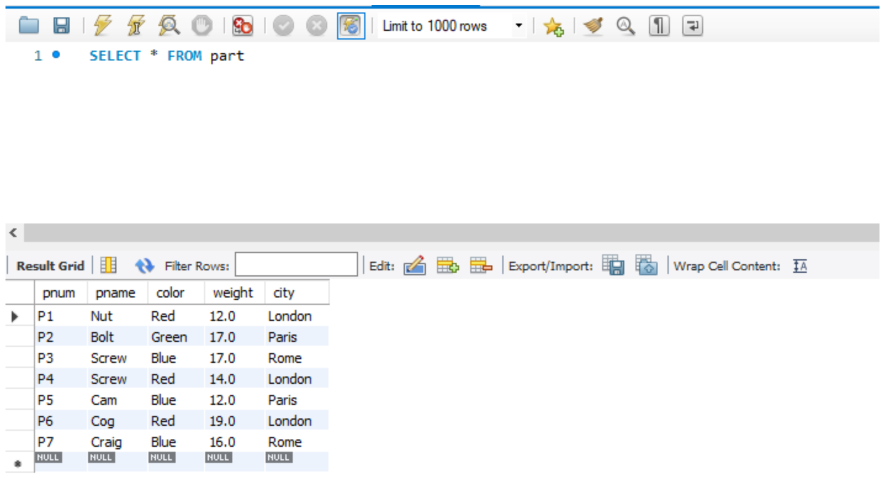
#	Time	Action	Message	Duration / Fetch
7	13:54:33	create table if not exists part (pnum char(6) not null, pname char(20), color char(6), weight dec...	0 row(s) affected	0.047 sec
8	13:54:33	drop table if exists project	0 row(s) affected, 1 warning(s): 1051 Unknown table 'spj.project'	0.000 sec
9	13:54:33	create table if not exists project (jnum char(4) not null, jname char(20), city char(15), primary key ...	0 row(s) affected	0.063 sec
10	13:54:33	create table if not exists shipment (snum char(5) not null, pnum char(6) not null, jnum char(4) not nu...	0 row(s) affected	0.078 sec
11	13:54:41	use spj	0 row(s) affected	0.000 sec
12	13:54:41	delete from shipment	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses ...	0.016 sec

Fig. 8: SCHEMAS screenshot.

Q: What are the primary keys of each of the created tables? What are the foreign keys? Make sure you know the difference between primary and foreign keys.

To start querying, go to "*File*" → "*New Query Tab*". An example query is shown in **Fig. 9**.

An example query to select all data from Table "*part*" is:



```
SELECT * FROM part
```

pnum	pname	color	weight	city
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Rome
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London
P7	Craig	Blue	16.0	Rome
NULL	NULL	NULL	NULL	NULL

Fig. 9: Example query to show Table "*part*".

5. Exercises

A) SPJ database

Use the SPJ database for this exercise. This database are already created using the SQL scripts. First, create a new SQL file to make the following SQL queries. Save it on your computer as "Exercise_A_SQL".

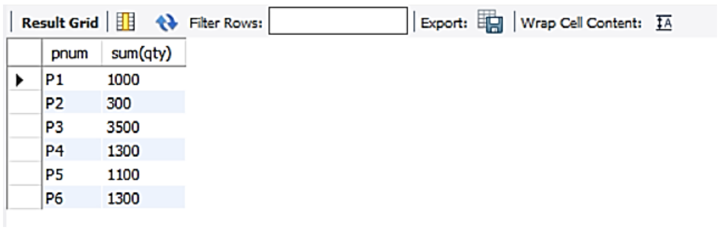
Exercise A.1: Insert the following two rows in the Table "*part*":

P7, Glass, Blue, 16, Rome

P8, Sensor, Yellow, 14, Paris

Example 1) *Show the total shipped quantity per part.*

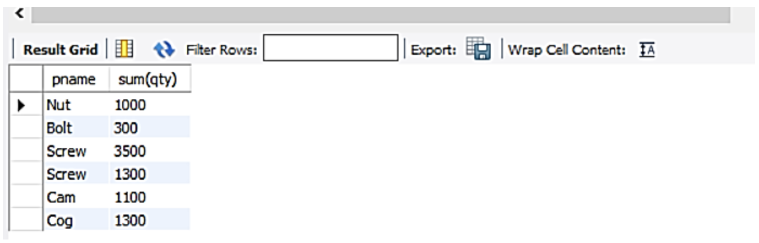
SELECT *pnum, sum(qty)* ***FROM*** *shipment* ***GROUP BY*** *pnum*



pnum	sum(qty)
P1	1000
P2	300
P3	3500
P4	1300
P5	1100
P6	1300

Example 2) *Show the name of parts in a first column and the total shipped quantity per part in a second column.*

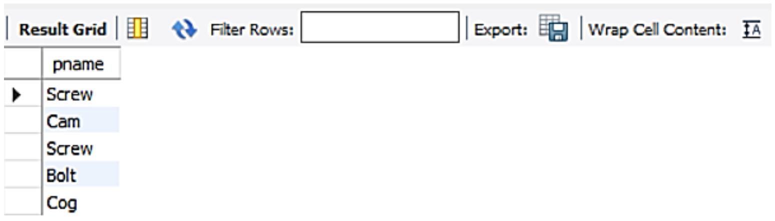
SELECT *pname, sum(qty)* ***FROM*** *part* ***AS*** *a* ***JOIN*** *shipment* ***AS*** *b* ***ON*** *a.pnum = b.pnum* ***GROUP BY*** *a.pnum* ***ORDER BY*** *a.pnum* ***ASC***



pname	sum(qty)
Nut	1000
Bolt	300
Screw	3500
Screw	1300
Cam	1100
Cog	1300

Example 3) *Show all parts used in project Display.*

SELECT *pname* ***FROM*** *(****SELECT*** *pname, jnum* ***FROM*** *shipment* ***AS*** *a* ***JOIN*** *part* ***AS*** *b* ***ON*** *a.pnum=b.pnum)* ***AS*** *c* ***JOIN*** *project* ***AS*** *d* ***ON*** *c.jnum=d.jnum* ***WHERE*** *jname='Display'*



pname
Screw
Cam
Screw
Bolt
Cog

Exercise A.2: Select the names and corresponding cities of all suppliers.

Exercise A.3: Select shipments where the shipped quantity is at least 500. Save the output as a csv-file on your computer as "Output_exercise1.3.csv".

Exercise A.4: Show all projects that are based in London.

Exercise A.5: Give all shipments for project J7 where the quantity is at least 200.

Exercise A.6: Select parts with a name that starts with the letter "C", with a weight of at least 15 and with a city that is not Paris. (**Hint:** use a REGEXP[5] operator.)

Exercise A.7: Show the total weight of parts per city.

Exercise A.8: Select suppliers based in London and sort the data by increasing value of status. (**Hint:** use ORDER BY operator.)

Exercise A.9: Show the total quantities per shipment where you exclude the shipments with part "P1". Show only the shipments with a total quantity larger than 1000. (**Hint:** the WHERE operator cannot be used for aggregate functions).

Exercise A.10: Show the total shipped quantity of each part.

Exercise A.11: Give all shipments for project Tape with a minimum part quantity of 200. (**Hint:** use a JOIN operator.)

Exercise A.12: Give all part names shipped from London.

Exercise A.13: Give for each shipment the total weight.

Exercise A.14: Give for each shipment the total weight for projects in Paris.

Exercise A.15: Give the total weight of parts for each project. The first column is the project name, the second column is the total weight of parts.

Exercise A.16: Give all pairs of parts that are within the same city. (**Hint:** use a JOIN operator.) Make sure you do not show pairs of a part with itself.

[5] A regex or regular expression is a sequence of characters that define a search pattern. For more information, see previous lectures on regexes.

B) MyFlix database

For this exercise, you use the "MyFlix" database. This database was already created in earlier sections with the "MyFlixDB.sql" script. Create a new SQL file to make the following SQL queries. Save it on your computer as "Exercise_B_SQL".

Exercise B.1: Select the names, gender and physical address of all members in the database

Exercise B.2: Return a list of all members showing the membership ID, full names and year of birth. (**Hint**: use LEFT() function).

Exercise B.3: Return a list of movies from the database where you show the movie title and year of release in one field. Show the year in brackets. In a second column, show the director of that movie. (**Hint:** look at Concat() function.)

Exercise B.4: Show a list of members of the MyFlix database who are referred to "MyFlix" movie rental company and show in a second column the name of the person that referred them. Change column names respectively to "member" and "referred_by" using aliases.

EXTRA exercises (you can send your solutions to yanou.ramon@uantwerpen.be):

C) Courses database

Create an ER diagram for a "Courses" database[6] with following criteria:

- There are courses, teachers, and students
- Each course has an ID, a title, credits, description
- Courses have multiple sections that have a timing and they have one teacher
- Students' course schedules must be tracked and also their transcripts with grades should be recorded
- The database must track which classes the professor has taught
- Database should work over multiple semesters

D) Medical database[7]

Design your own logical database model (ER diagram) which contains information on patients visiting a medical doctor. The requirements are:

- Patients make visits to Doctors located in Medical Centers
- Patients and Medical Centers have Addresses that are stored in an Addresses Table for easy validation.
- Medical Procedures such as X-Rays are carried out on Patients
- Each Procedure has a Cost and the total Cost of Procedures for each Visit is derived and stored
- During a Visit, the Doctor might prescribe Medication for the Patient. Each Medication has a cost and the total Cost is derived and stored.

[6] This exercise is part of http://www-inst.eecs.berkeley.edu/~cs186/fa05/lecs/8ER-6up.pdf
[7] This exercise can be found on the following website: http://www.databaseanswers.org/data_models/patients_visits_to_the_doctor/index.htm