School of Computing Technologies

ISYS1101/ 1102 Database Applications

Week 1: Tute/Lab – Database Concepts Revision and Preparatory Work

Semester 2 2022

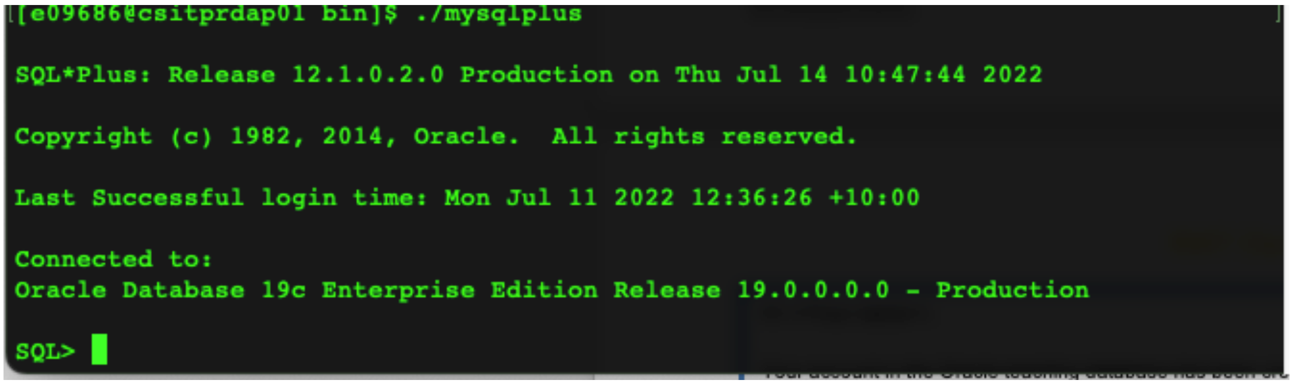
1 Objective

The objective of this tute/lab session is to revise what's learned in your first database course, specifically:

- Make yourself familiarised with RMIT University's Oracle database setup;
- Familiarise with Oracle SQL Developer;
- Learn how to create tables in Oracle;
- Learn how to do simple CRUD (create, read, update, and delete) data in tables;
- Learn about integrity constraints;
- Revise SQL programming that you had previously learned in (Practical) Database Concepts or other similar pre-requisite courses.

1.1 Preparation Tasks

For the tasks in this week as well as the future lab work and assignment 1, you are required to have Oracle SQL Developer application installed in your computer. Oracle SQL Developer is an Interactive Development Environment that allows you to interact with an Oracle Database Server installed locally or remotely. In this course, you will be using RMIT University's Oracle Server, as such you are not required to install a database server.



```
[e09686@csitprdap01 bin]$ ./mysqlplus

SQL*Plus: Release 12.1.0.2.0 Production on Thu Jul 14 10:47:44 2022

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Mon Jul 11 2022 12:36:26 +10:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL>
```

There are two ways to connect to RMIT's SQL Server.

1. Using Oracle SQL Developer, installed in your computer. We primarily use this method in this course
2. Using sqlplus Unix tool, available on school's core teaching servers (titan.csit.rmit.edu.au, Jupiter.csit.rmit.edu.au, and Saturn.csit.rmit.edu.au). There is a small guide on how to use sqlplus on core teaching servers, however, no further support provided.

1.1.1 Installing Oracle SQL Developer

Oracle SQL Developer is available as a free-to-install, downloadable from Oracle Technology Network.

For more information on Oracle SQL Developer (installation guide, features and how-to-use guide, etc) is available at: https://docs.oracle.com/en/database/oracle/sql-developer/22.2/index.html

It is downloadable from: https://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html (Links to an external site.)It is released under Oracle Technology Network License Agreement (https://www.oracle.com/downloads/licenses/sqldev-license.html (Links to an external site.)). If you have limited bandwidth to download installation files from Oracle Technology Network, copies are locally available.
Modules → Course Welcome and Orientation → Course Resources

Installing on a Windows Laptop
Installation on a 64-bit Windows laptop is very straight-forward. Please follow the instructions given at: https://docs.oracle.com/en/database/oracle/sql-developer/22.2/rptig/installing-sql-developer.html

Installing on a Mac
Installation on a Mac depends on whether you have Java 11 JDK is available on your Mac or not. If you do not have JDK installed, you will need it installed prior to install Oracle SQL Developer. The .dmg file for Java 11 JDK is also available at:
Modules → Course Welcome and Orientation → Course Resources

Then, you can follow installation instructions at https://docs.oracle.com/en/database/oracle/sql-developer/22.2/rptig/installing-sql-developer.html

Installing on a Linux
The major Linux distributions, such as RedHat, Ubuntu, Suse, and Oracle Linux are supported. Follow the instructions on https://docs.oracle.com/en/database/oracle/sql-developer/22.2/rptig/installing-sql-developer.html

1.1.2 Setting up Oracle Account on RMIT Oracle Server

As an enrolled student in the ISYS1101/1102 Database Applications course, you have been provided access to Oracle 12c database server at the RMIT University. Oracle is a classic client-server application where you can use a multitude of different client applications to access the database server.
By now you should have received an email from ITS services regarding with your Oracle access details. E.g.:

Hi <<Your name>>,

Your account in the Oracle teaching database has been created.
Your login credentials are as follows:

Please note the old service name GENERAL / GENERAL.CS.RMIT.EDU.AU is now updated to CSAMPR1.ITS.RMIT.EDU.AU

Connection Name	CSAMPR1 (this is only a descriptive label)
Username	<<Your Student Number>>
Hostname Name	talsprddb01.int.its.rmit.edu.au
Service Name	CSAMPR1.ITS.RMIT.EDU.AU

You can set your password at https://access.csit.rmit.edu.au
This site can only be accessed within RMIT Network.

If you need to access this site via non-RMIT network, please access it via a web browser through MyDesktop.

Please consult your instructor on how to access this database for your class.

If your instructor has identified issue with your account or access, please contact the IT Service and Support Centre.

MyCommunity Link for accessing Oracle database using sqlplus and SQLDeveloper : How to acces Oracle Database

You have the access to two resources in RMIT's Oracle server.

1. Sample training databases: movies, library, flight, sales, research etc.
2. Your own database work space

You will use your own database work space for this week's Tute/ Lab activities.

In this exercise we use Oracle SQL Developer client to access your Oracle account.

[**FIRST TIME ONLY**] Goto https://access.csit.rmit.edu.au and reset your Oracle password.

DO NOT use your RMIT password as your Oracle password. In future labs, you may be required to use this password with PHP code in plain text, and is not secure there.

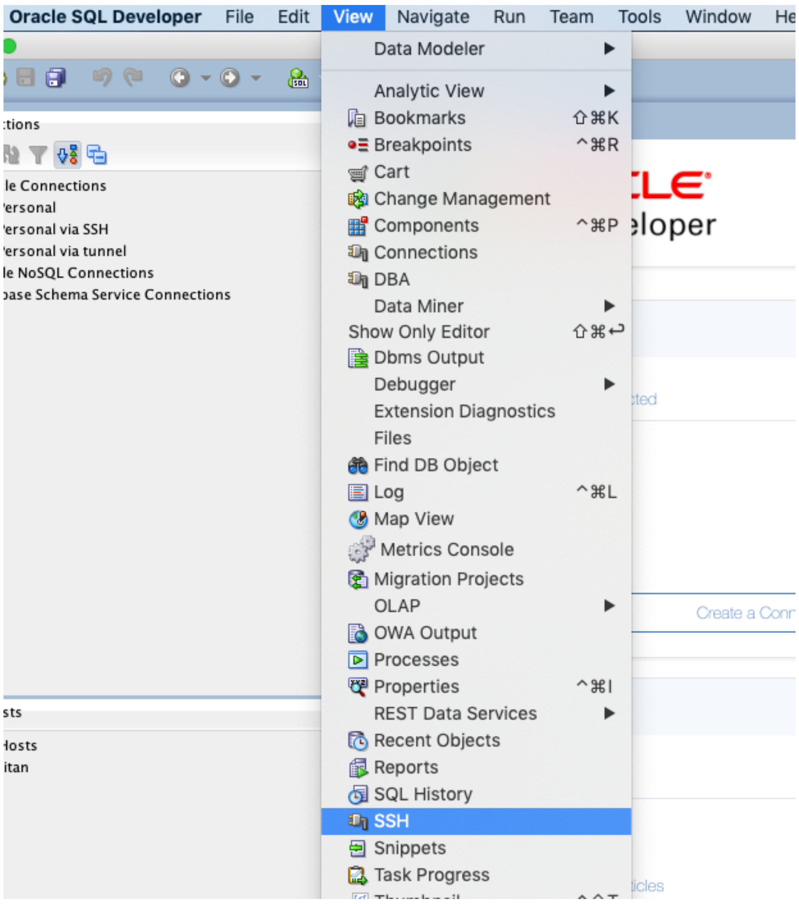
The following activity must be completed on your laptop. If you do not have a laptop in-class, you may be able to use RMIT's virtual Desktop Environment (mydesktop.rmit.edu.au) on desktop computer in the lab. However, it is expected that you use your own Laptop in the remaining lab sessions.

Your Lab assistant will demonstrate you how to access MyDesktop, choose Applications folder and start up the SQL Developer application.

Step 1: **Setting up SSH on SQL Developer**
You **cannot** directly connect to Oracle which is accessible from port 1521 of `talsprddb01.int.its.rmit.edu.au` server from your home computer. You will be required

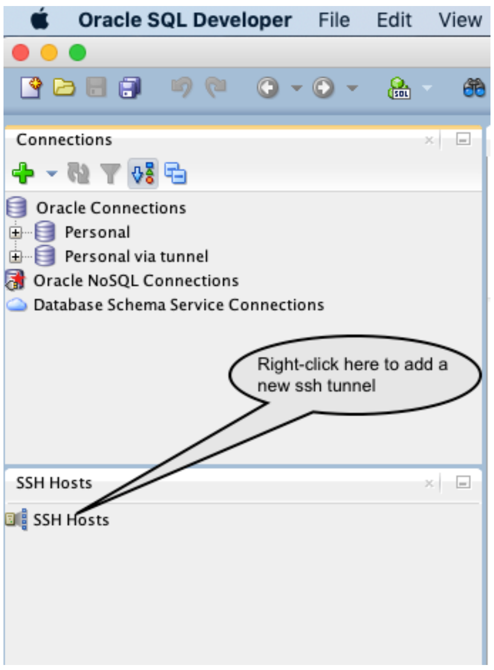
to establish a connection via one of public core teaching servers (such as titan.csit.rmit.edu.au). In the networking world, this method is called “ssh tunneling”

Open SQL Developer.
By default, SSH Hosts panel is not open on SQL Developer. From the main menu, choose View → SSH

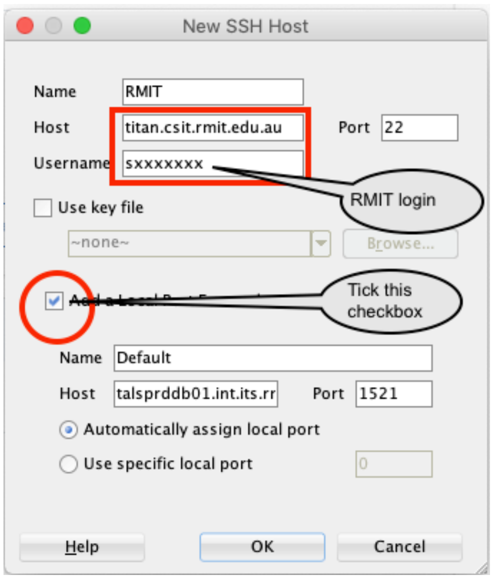


This will add SSH Hosts pane, as follows.

Add a new SSH tunnel.
Right-click on the “SSH Hosts”

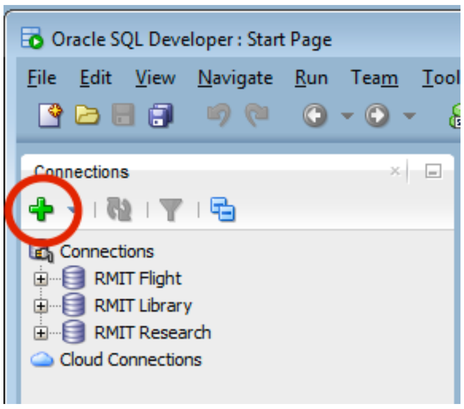


A dialog box will appear. Enter on of the core teaching servers details (Saturn, Jupiter or Titan) as follows.



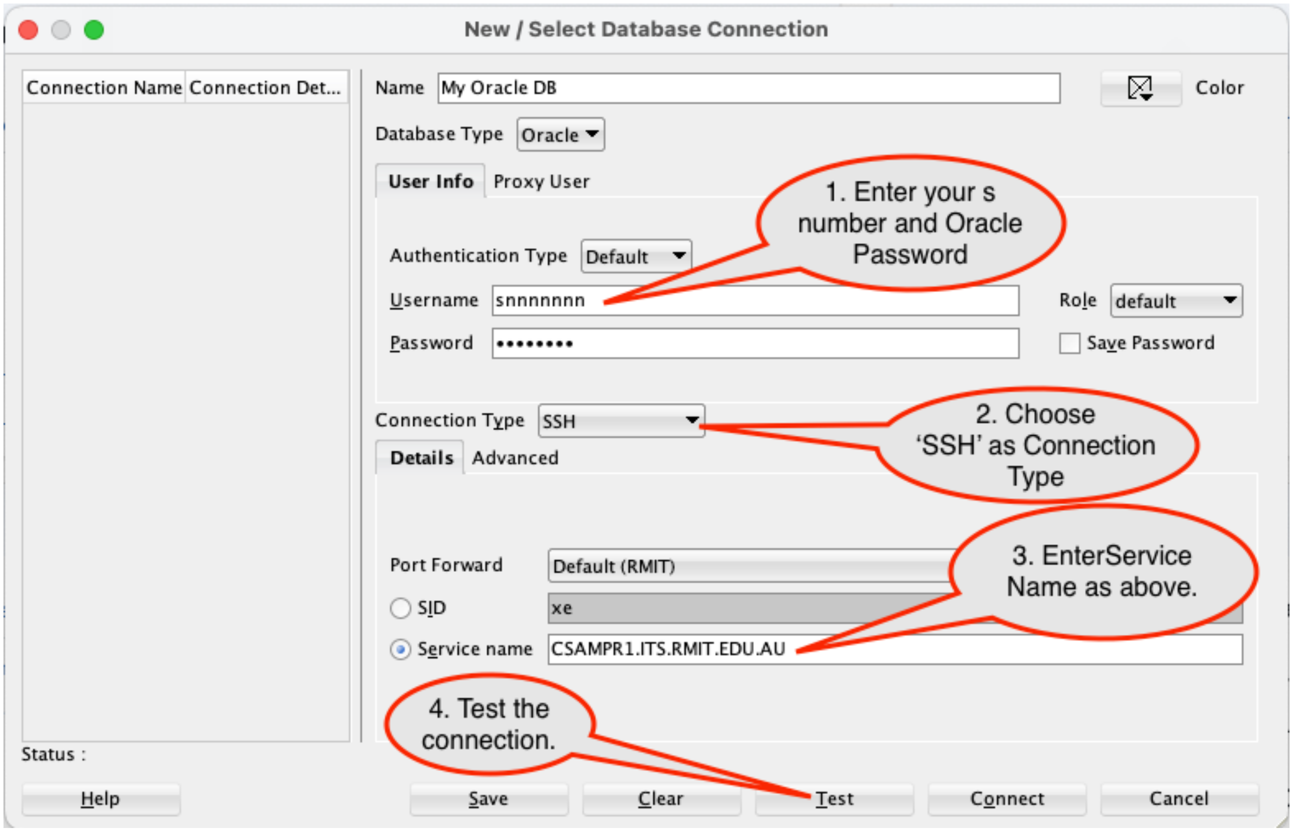
Name: RMIT
Host: titan.csit.rmit.edu.au
Port: 22
Username: sxxxxxxx
Use key file: ☐
~none~
☑ Add a Local Port Forward
Name: Default
Host: talsprddb01.int.its.rr
Port: 1521
◉ Automatically assign local port
○ Use specific local port: 0

Step 2. Add a new connection.
Click on the + symbol on the left-hand pane.



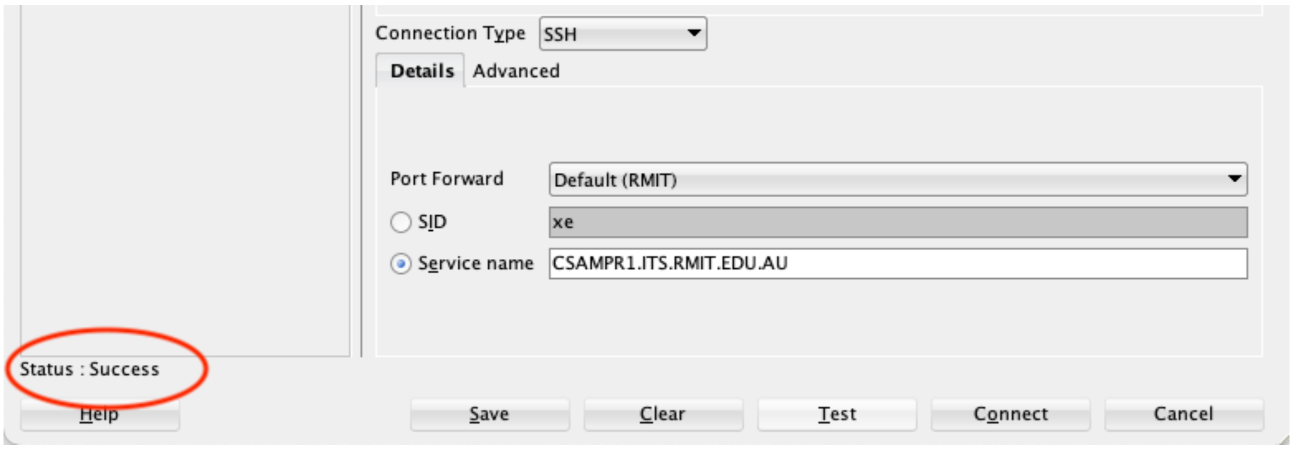
Step 3: On the Connection dialog box, enter connection details and login details. Use the details you received from ITS:

Connection Name	My Oracle DB (you may call it whatever you wish)
Username	Snnnnnnn
Password	Your Oracle Password (NOT your regular RMIT single sign-on password)
Hostname Name	talsprddb01.int.its.rmit.edu.au
Port	1521
Service Name	CSAMPR1.ITS.RMIT.EDU.AU


New / Select Database Connection
Connection Name
Connection Det...
Color
User Info
Proxy User
1. Enter your s number and Oracle Password
Details
Advanced
2. Choose 'SSH' as Connection Type
3. EnterService Name as above.
4. Test the connection.
Status :
Help
Save
Clear
Test
Connect
Cancel

Name: My Oracle DB
Database Type: Oracle
Authentication Type: Default
Username: snnnnnnn
Role: default
Password: ••••••••
Save Password: ☐
Connection Type: SSH
Port Forward: Default (RMIT)
SID: xe
Service name: CSAMPR1.ITS.RMIT.EDU.AU

Step 4: Test the database connection.


Connection Type: SSH

Port Forward: Default (RMIT)

SID: xe

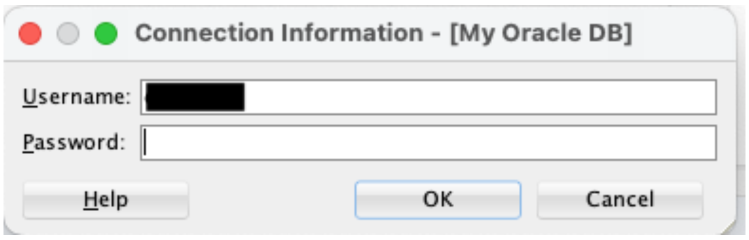
Service name: CSAMPR1.ITS.RMIT.EDU.AU

Status : Success

If the test failed, double-check the details you entered in step 3.

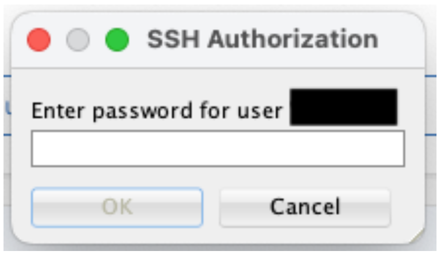
Step 5: Save your connection details and then press 'Connect'.

It will ask to enter your Oracle password once again and the RMIT password to authenticate ssh connection.


Connection Information - [My Oracle DB]

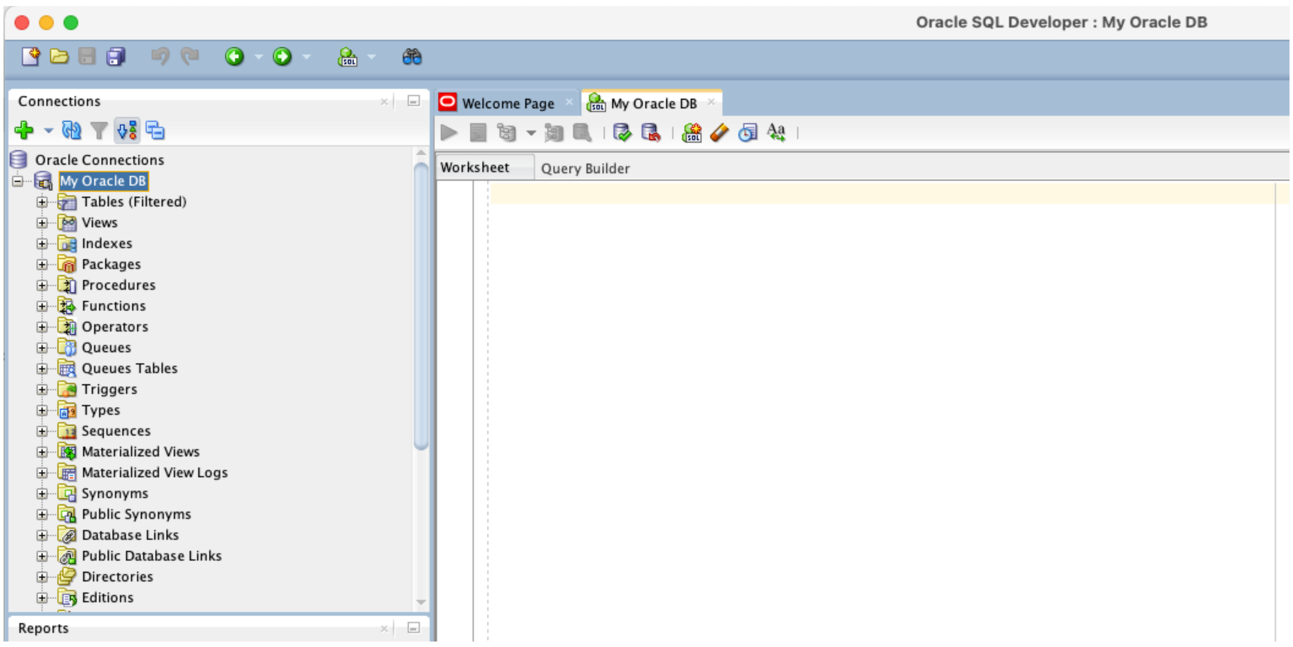
Username: [redacted]

Password:


SSH Authorization

Enter password for user [redacted]

If you are successful, you should have presented a blank window where you can enter and run SQL commands.



Activity 1: Creating Database Tables in Oracle

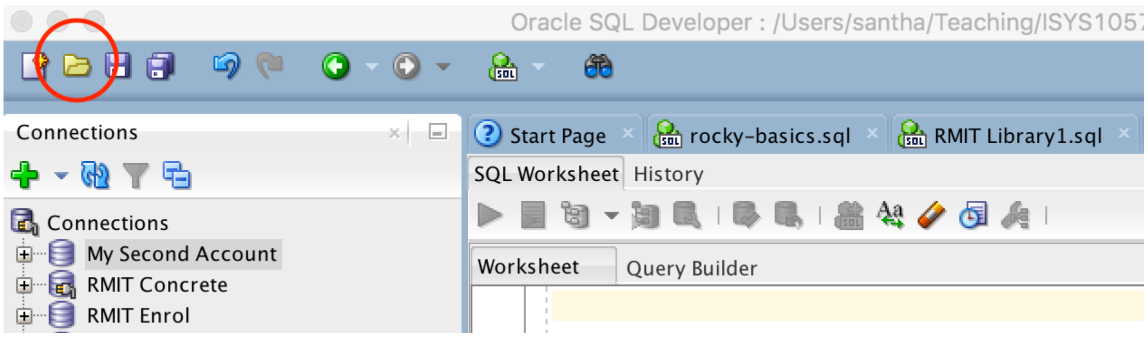
Log in to your Canvas shell, go into "Course Resources" → "Sample Databases" → "Movies Database" area. There are a bunch of SQL script files in that folder.
Download them into a folder in your computer (or a folder on MyDesktop).

	Your Lab assistant will demonstrate you how to copy files between your desktop computer and MyDesktop.

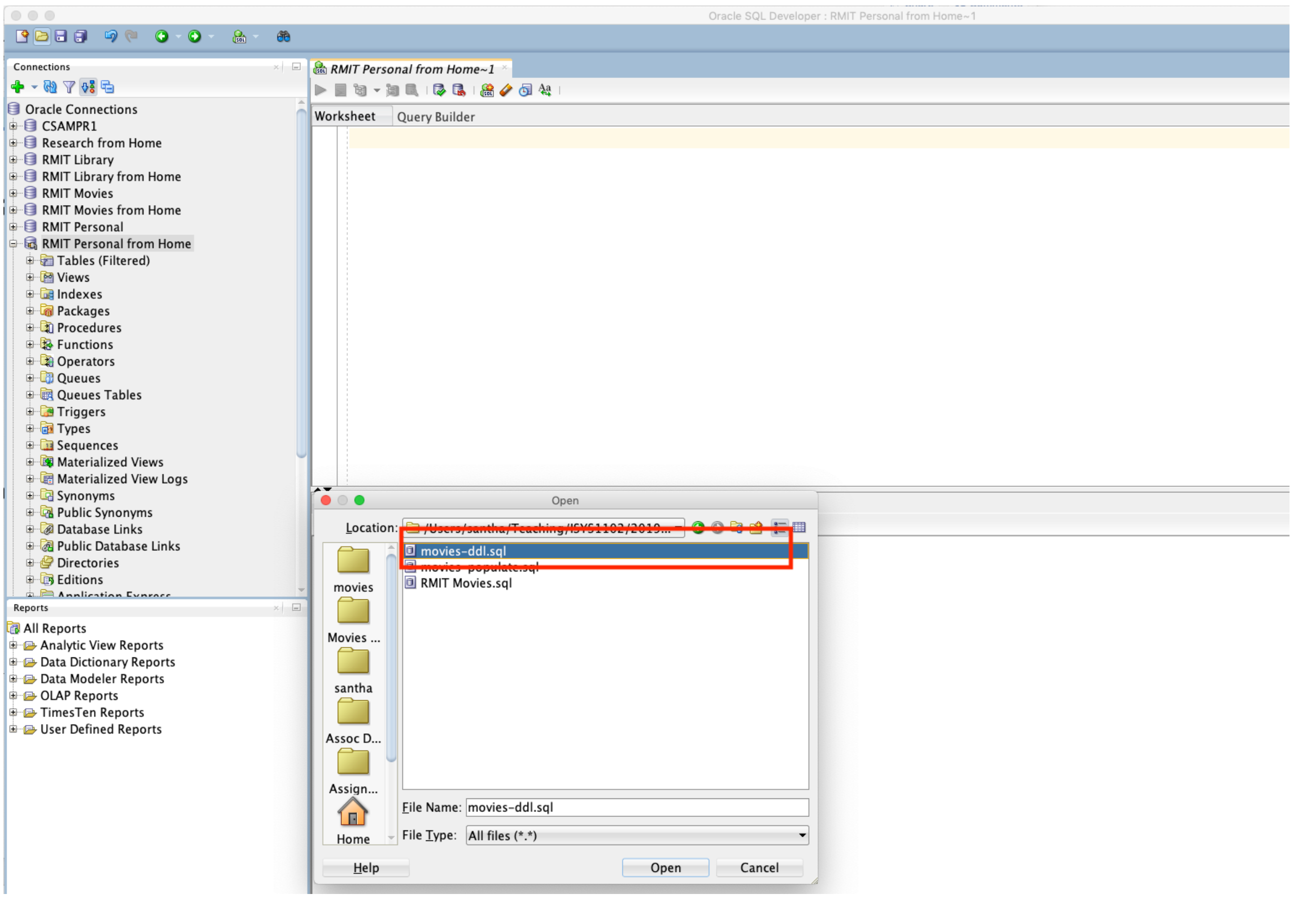
In this activity, you familiarize yourself with running a SQL DDL script in Oracle SQL Developer to create following four tables.

- Movie
- Director
- Star
- Movstar
- Member
- Borrow

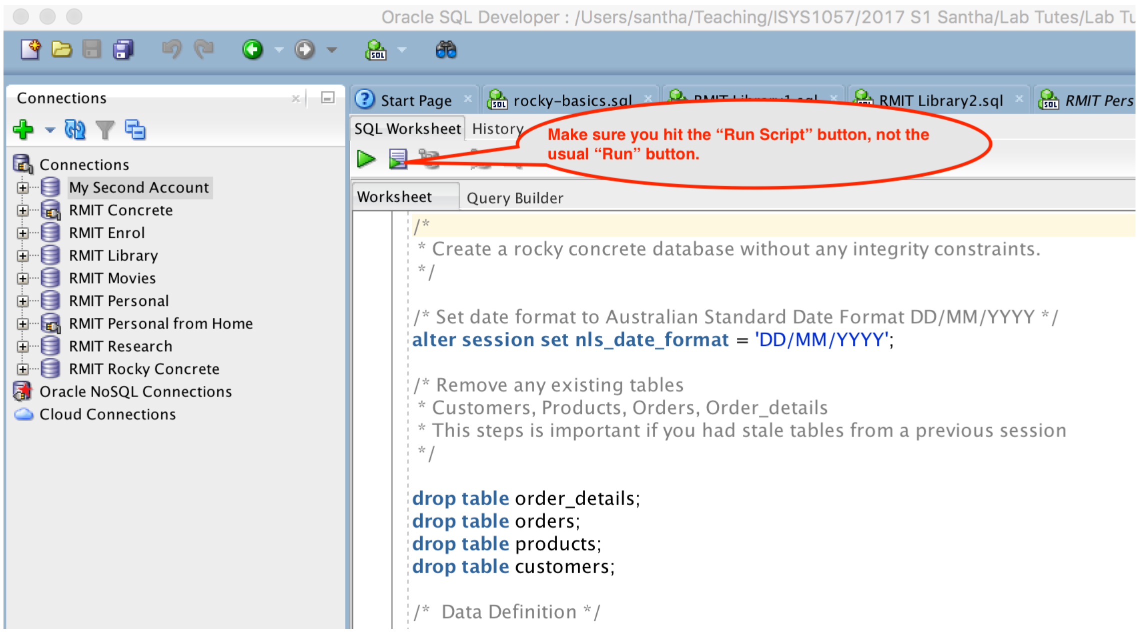
On Oracle SQL Developer, Click on "Open Script" button as shown on the following screenshot, or go to File → Open.



On Open Script dialog window, choose the “movies-ddl.sql” file.



Once the script appears on your SQL Editor, hit the “Run Script” button on the top. Do not use “Run” button this time.



```
/*
 * Create a rocky concrete database without any integrity constraints.
 */

/* Set date format to Australian Standard Date Format DD/MM/YYYY */
alter session set nls_date_format = 'DD/MM/YYYY';

/* Remove any existing tables
 * Customers, Products, Orders, Order_details
 * This steps is important if you had stale tables from a previous session
 */

drop table order_details;
drop table orders;
drop table products;
drop table customers;

/*  Data Definition */
```

1. To list all the tables in the your database, run the following SQL command.

```
SELECT * FROM tab;
```

2. To view full listing of Movie table in, run the following SQL command.

```
SELECT * FROM movie;
```

Since you haven't populated the tables yet, you won't see any rows.

3. To view Movie table schema, run the following SQL command.

```
describe movie;
```

This will show the list of attributes in the table with data types and (if any) constraints.

```
Name     Null?    Type
-------- -------- ----------
MVNUMB   NOT NULL NUMBER(38)
MVTITLE           CHAR(30)
YRMDE             NUMBER(38)
MVTYPE            CHAR(6)
CRIT              NUMBER(38)
MPAA              CHAR(2)
NOMS              NUMBER(38)
```

Activity 2: Populating Database Tables in Oracle

To populate tables in the Movies database, follow these steps.

On Oracle SQL Developer, click on "Open Script" button as shown above, or go to File → Open.

On Open Script dialog window, choose the "movies-populate.sql" file. Then, run the script in tha same as above.

1. To view full listing of Movie table in, run the following SQL command.

```
SELECT * FROM movie;
```

Now, you should be able to see the rows you just entered.

2. Before you continue onto next step, it is a good idea to commit the transactions you have successfully completed.

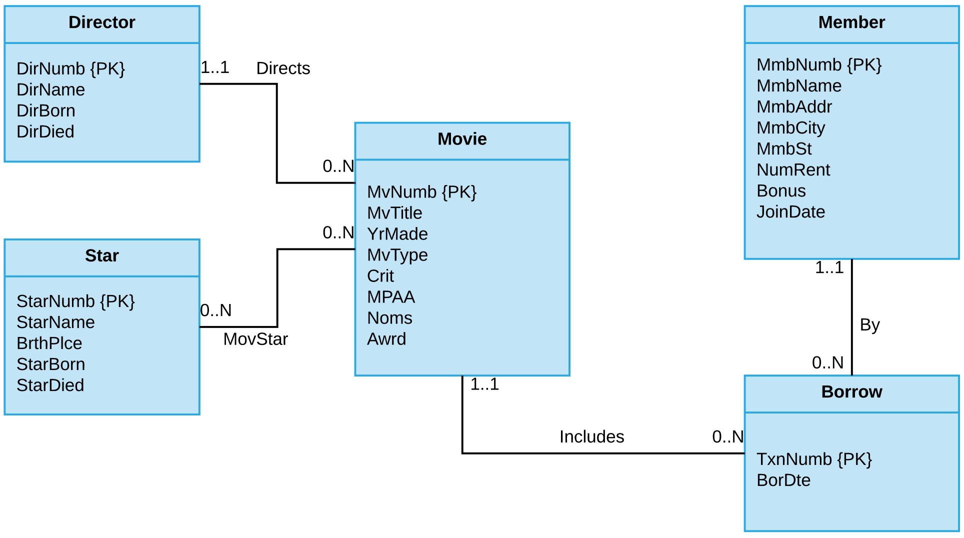
```
commit;
```

2 SQL Questions

The following SQL exercises were extracted from ISYS3412 Practical Database Concepts course. If you are very comfortable with SQL programming, including advanced concepts such as complex joins, nested queries, etc, you may skip this section. However, if your SQL programming is a bit rusty or you wish to improve your SQL programming skills, complete these exercises.

The movies database is built to store information about movies, directors, and stars (i.e. actors and actresses) in a video store. In addition, it stores information on members (who got active memberships to borrow videos from the video store) and their borrowing transactions.

A sketchy ER diagram for the movies database is given below.



2.1.1 Queries using a single table

The following is the script of an imaginary scene associated with the video shop. You are required to provide the database assistance to Penny, the checkout clerk, by giving him the SQL command for each of the query required in the discussion. All of these queries can be answered with using a single table. As such, joining two or more tables is not an skill you develop in this exercise.

Jerry, a client (a member of the video rental shop) comes to the shop and is greeted by Penny.

Penny:
Good afternoon, I am Penny. May I help you, Sir?
Jerry:
Yes please. What are the movies available in your collection?
Penny:
Just a second, I'll check it in the database.

This question can be interpreted as "list all movies in the *Movies* table. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks Penny. But that's too much information. I am overwhelmed with information overload. Can you show me the movie titles, nothing else.
Penny:
Of course. That is easy.

In database terms, this question can be interpreted as "display the titles of all movies in the *Movies* table." Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
That's nice. Very impressive movies collection. By the way, how many movies are there in your collection?
Penny:
Let me count them on the screen, ….. , oh that's too long, (and while scrolling down), I lost the count. I am thinking of a better way to this.

In database terms, this question can be interpreted as "count all the movies in the *Movies* table." Write a SQL statement to assist Penny to obtain required data from the database.

Penny:
There are 24 movies in our collection. Are you interested in borrowing a few for tonight?
Jerry:
I am interested in Horror movies. So, can you help me find some?
Penny:
Of course. Let me find some horror movies.

In our database schema, there is an attribute called "`MvType`" which stores movie type. What is required in database terms is to filter the Movies table on this attribute. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks you. That's nice. Out of these horror movies, are there any movies nominated for academy awards?
Penny:
I recall there were some very highly regarded horror movies. Let me find out.

In our database schema, there is also an attribute called "`noms`" which stores the number of nominations each of the film received. What is required in database terms is to filter the Movies table on this attribute in addition to the filtering on `MvType`. "`noms`" must be greater than 0. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks you. I'll borrow them. I might take a few comedies for my kids. Can you help find some.
Penny:
I am here to help. Let me find out.

This is a tricky one. You have to interpret the client's requirements carefully. He wants to borrow horror movies with nominations and comedy movies. However, when this generic requirement converted to database query terms, the results set should meet one of the two criteria: (1) good horror movies with nominations; or (2) comedy movies. The first condition itself consists with two components. So, the `WHERE` clause should contain three conditions, which are carefully ordered (using parenthesis if required). Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:

While we are at it, I must admit that my kids are also very picky about movies. Have these comedies received any award nominations?
Penny:
Why not? Let's try that.

You have to interpret this requirement differently. He wants to borrow horror movies or comedy movies, with those with award nominations. We must re-arrange the conditions by changing where the parenthesis are used. Here we first retrieve movies of either type and then see if they have received any nominations. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks. Having being nominated to an award is one thing, and receiving an award is definitely quite an achievement. What are the movies in your collection, which have received more than three Academy Awards?
Penny:
That's easy to find.

This is in fact an easy question to interpret. What we have to look for is values greater than 3 in `awrd` attribute. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
I vaguely remember there was an enthralling movie about the Nuremberg Trials. But, I cannot recall the exact title. Do you know what is it?
Penny:
Cannot remember from the top of my head. But, we can check the database.

This is a classic partial match query. The client knows a word (or a phrase) and requires to retrieve all matching rows. (If the result set is still too broad (lengthy) we may require to narrow down our partial match or add more conditions). The client knows that the word "Nuremberg" in the movie title, so we must use wildcard characters appropriately placed in the comparison string to be included in the `LIKE` condition within `WHERE` clause.
Write a SQL statement to assist Penny to obtain required data from the database.

Extension to this is that if you do not know the case of the data stored in the database (upper-case, lower-case, mixed-case). For example, Nuremberg can be stored as either Nuremberg or NUREMBERG or nuremberg.
In this case, your comparison string should be able to pick any of these three.

Write a SQL statement to assist Penny to obtain required data from the database, assuming Penny doesn't know in which case the data are stored.
Hint: use `OR` operator.

Jerry:
Thanks Penny. You are very helpful. I'll take these videos today. See you again next week.
Penny: You are mostly welcome.

2.1.2 Advanced Activity

After the client, Jerry, left, Penny was called by Anna, their store manager.

Anna:
Hi Penny, I heard that you are a database expert.
Penny:
I wouldn't call myself a database expert. I can write simple SQL statements, though.
Anna:
That's good. I have few questions about our clients.
I am concerned about how active our clients are. On average, how many rentals in total they have done?
Penny:
Let's see if we can obtain these data.

This requires the use of aggregate functions. We have already used one aggregate function – `COUNT` – to count rows in a result set. In this query, we have to average out a column that contain the total number of rentals each member has made. `NumRent` attribute holds these data. You are required to calculate the average of the numbers down that column.
Write a SQL statement to assist Penny to obtain required data from the database.

Anna:
Thanks Penny. That's promising. We may start a promotional campaign to encourage our clients to rent more videos. I was thinking of having a free video day – clients can rent a video for free on their membership anniversary day. Can you prepare a report containing member names and their join dates?
Penny:
Easy. Here it is.

The Member table has `joindate` attribute. So, this is, in fact, an easy task.
Write a SQL statement to assist Penny to obtain required data from the database.

Anna:
Sorry Penny, but, I do not like the date format you have in your report. Can you produce these dates in standard Australian date format?
Penny:
Yes, it is possible. Here the dates are listed in DD/MM/YYYY format.

Different database systems manage dates differently. So, Penny must know about date functions, date formatting, and internal date storage types for the database he uses.

In SQLite, dates are stored as either text, integer or real data. For more information on handling dates in SQLite, please refer to Module 6 additional slides (named "Dates in SQLite and Oracle").

In Oracle, we can format the output whatever the style you wish, by changing the session settings. After changing these settings, it displays the dates in the desired format until you close the current session. The session variable `NLS_DATE_FORMAT` can be used to set the date format of the current session.
Similarly, we can use `NLS_TIMESTAMP_FORMAT` to change timestamp format.

Write a SQL statement to assist Penny to change the date format displayed on the current session.

Anna:
Thanks, I like the current look of it. Can you please add another column to display how long they are members with us.
Penny:

I think I know how to do this. Do you want to display it in days or years?
Anna:
Just show them both.

We can do simple arithmetic calculations (such as adding or subtracting dates) with `DATE` type data. However, the results may require some form of post-processing (such as changing the display format, rounding up or rounding down, etc).

Write a SQL statement to assist Penny to obtain required data from the database.

Anna: That's great. But, isn't it nice if you sorted these results. Say, sort on the alphabetical order of names.
Penny:
No problem.

This is an easy task. You simply add an `ORDER BY` clause. You will be surprised to know that even experienced database programmers incorrectly add a `SORT BY` clause!

Write a SQL statement to assist Penny to obtain required data from the database.

Anna:
Mmm, I think it is better to sort by the tenure (i.e. number of days since they joined) in the descending order. Can you fix it?
Penny:
That's easy.
This is an easy task. You simply add an `ORDER BY` clause. But, remember to add `DESC` at the end.
Write a SQL statement to assist Penny to obtain required data from the database.

Anna:
That's not bad. However, it is nice if you can change column headers to make them clear and meaningful. I am going to present this report to my colleagues.
Penny:
No worries. Do you like to call the columns, say Member Name, Date Joined and Number of days since joined?
Anna:
Yes, that will do.
It is always a good practice to rename column names with more readable, meaningful column headers. Use `AS` keyword to substitute column headers.
Write a SQL statement to assist Penny to obtain required data from the database.

Anna:
Thanks. Have a nice day.
Penny:
You are welcome.

2.1.3 More difficult Queries that require more than one table

Jerry, a client (a member of the video rental shop) comes to the shop and is greeted by Penny.

Penny:
Good afternoon, I am Penny. May I help you, Sir?
Jerry:
Yes, please. The other day you showed me the full list of movie titles, but didn't show me the directors who directed them. Can you show me?
Penny:
Just a second, I'll check it in the database.

This question requires data from two tables: Movie and Director. Therefore, we must join them using the common attribute. DirNumb exists in Movie table as a foreign key, referencing Dirnumb in Director table. So, we can use that pair to join two tables. After the join, we filter movies on Director name (DirName). Use the simple JOIN .. ON within FROM clause to achieve this result.

Write a SQL statement to assist Penny to obtain required data from the database.

Furthermore, since both joining attributes have the exact same name (DirNumb) we can use NATURAL JOIN in this particular exercise.
Try the above query again, using NATURAL JOIN.

Jerry:
What an impressive collection! By the way, what are the movies directed by Woody Allen in your collection?
Penny:
Just a second, I'll check it in the database.

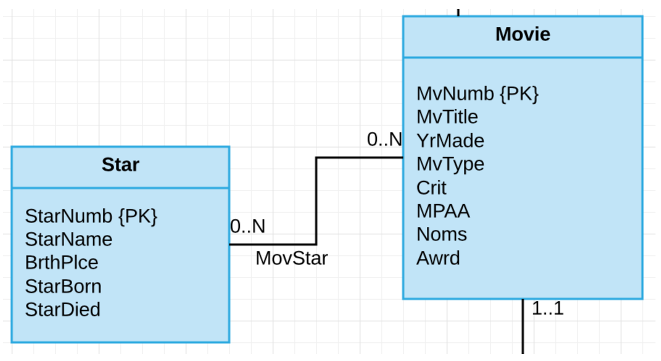
This is a simple extension to the previous query. After doing the join (as in previous one), we filter on the DirName.
Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Just for curiosity. What are the movies you have in the collection, in which the director is still alive? And who are the directors of those movies?
Penny:
That's easy, we have bio data of directors in our database.

That's an easy query. Just extend the previous query with extra filtering condition. If DirDied attribute is blank, that means they are still alive.
Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks, Penny. Great. I'd like to know more about *Manhattan* movie. Who are the lead actors in that movie?
Penny:
Of course. That is easy.

This requires data from Movie table and Star table. But, the difficulty is that we cannot directly join these two tables (no common attributes between them). The reason was that they came from a many-to-many relation in the underlying ER model, where were required to add extra relation in between. The MovStar table in our database represents this relation, which comprises of primary keys from two relations at each end of the relationship.



As a result, we are required to use three tables to retrieve required information. In this case, we join movie table with MovStar table using MvNumb attribute, and then join this result with Star table with StarNumb attribute.
Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thank you. I am a bit curious, what are the movies that directors themselves played a role?
Penny:
Interesting question! I'll try to find out.

Hint: Even if it is the same person, dirnumb in Director relation and starnumb in Star relation may be different. So, comparing DirNumb vs StarNumb is meaningless and could result in incorrect results. Best approach is to compare the names in Director against names in Star, for each movie.

This require joining of Movie table with Director on one side and Movie table with Star on the other side. Then, match DirName and StarName attributes.

Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thank you. Since you have information about every aspect of movies, you must have a database of directors, too. Can you show me the full list of directors, with all the movies they have directed.
Penny:
Yes, I have information about directors.

This is a situation where a simple join does not produce the required output. A simple join will show the list of directors with their movies, but, it will miss the directors who haven't directed a movie. So, you will not get the "full list" of directors. What you must use here is LEFT OUTER JOIN which displays all rows in the table in the left of the join. So, use Director in the left of the join, and Movie on the right side. Then, the query will show the desired result with NULL values to pad up Director rows with no associated movies.

Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
That's interesting. Can you do the same for actors, too? I mean full list of actors with the movie list for each of them.

Penny:
I am here to help. Let me find out.

This is somewhat similar to the previous one. The difference is that we need to join three tables. Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks. I know that Woody Allen was a famous actor in his days. Do you know his co-actors?
Penny:
That's easy to find.

That's not as easy as Penny thinks. There are a few ways we can do it, though. One approach is to do a self-join. This is somewhat similar to what we did in the lectures to find the other languages spoken by the same people who spoke "Swahili". First we join Star and MovieStar to obtain MvNumb's where Woody Allen played. Then, make another pair of these two tables to of find the other actors who played in the movies that we have obtained in the first part of the query. Please note that we do not require Movie table in this join, because we are not asked to show the MvTitles in the result.

Write a SQL statement to assist Penny to obtain required data from the database.

As mentioned above, self-join is not the only way to do this query. Another approach is to use sub-queries using IN operator within WHERE clause.
The logic is as follows:
Build a query to produce a list of MvNumb's in which Woody Allen starred.
Next, use the result from this (inner) query in IN clause in the outer query which generates the list of actors in those movies.
Re-write the above SQL statement using IN.

Another way to build this query is to use EXISTS. The logic is as follows.
Open Star and MovStar tables.
For each row in this join, pick the corresponding MvNumb
See if Woody Allen starred in that movie, if so the original related to an actor in a movie co-starred with Woody. Return that row.
You may require to do some post-processing to eliminate Woody Allen from the result.
Re-write the above SQL statement using EXISTS.

Jerry:
What is the most recent movie you have got in your collection? To extend my question, what is the oldest movie you have got?

Penny:
Why not try to find? Let's try that.

This is a tricky one. You might think this is a simple matter of using the aggregate function MAX() on YrMade column will produce the result.

```
SELECT mvtitle, MAX(yrmde)
      FROM movie;
```

But, you will get an error message – you cannot mix attribute values and aggregate functions in one SELECT clause, as shown above.

Hint: You will require a nested query where the inner query produces one value (as a result of the aggregate function). Use it in the WHERE clause in the outer query.

Jerry: My favourite director is Alfred Hitchcock. What are the movies directed by Hitchcock, but I haven't borrowed yet?
Penny: To answer this, I need to know your member number.
Jerry: Even if I go by the name Jerry, my real name is Thanh Tran. But I cannot recall my member number.

This is a classic example for the use of IN clause. We can break this problem in to two parts.
Part 1: Generate a list of MvNumb's for Hitchcock movies Jerry had borrowed.
Part 2: Identify the Hitchcock movies NOT IN the result set from part 1.

Write a SQL statement to assist Penny to obtain required data from the database.

This query can be written using NOT EXISTS. The logic is as follows:
For each of Hitchcock movies,
Check if that movie is borrowed by Jerry.
If not, display the movie title.

Re-write the above query using NOT EXISTS.

Jerry: Among the Hitchcock movies, which one received the most academy awards?
Penny: Had Hitchcock ever received Academy awards? We can find it out from our database.

This query can be written using an aggregate function MAX. However, we still need to use a nested query to retrieve the title of the corresponding movie.
Write a SQL statement to assist Penny to obtain required data from the database.

Jerry:
Thanks Penny. You are very helpful. I'll take these videos today. See you again next week.
Penny: You are mostly welcome.

2.1.4 Queries using Grouping, Grouped Aggregation, and Sets

Penny:

Good afternoon, I am Penny. May I help you, Sir?

Jerry:

Yes, please. The other day you showed me a lot of information about your movie collection. But, to be honest, some of the directors and their movies are mediocre and not worth my time watching them. I want to see those directors who were successful in winning the Oscars. I have heard some of them have won multiple awards. Can you show me the directors and a count of those awards? I might pick a movie today from one of those directors.

Penny:

Just a second, I'll check it in the database.

This question requires joining directors and movies tables and then group the resulting rows into groups (each group contains movies from a director). Then you can do a sum of nominations within each group.

Write a SQL statement to assist Penny to obtain the required data from the database.

Jerry:

Thank you, Penny. However, that report is so messy with lots of directors with no awards and highly awarded scattered among them. Surely, there must be a way to see those highly awarded directors at the top of your list.

Penny:

Of course. Let me sort them as you wish.

Hint: This query will require **ORDER BY** clause to sort the result, however, the sorting has to be done on an aggregate (i.e. result you got from the SUM function, not based on an existing attribute).

Write a SQL statement to assist Penny to obtain the required data from the database.

Jerry:

That's better. However, the other day, I recall you mentioned about the director Fredricco Fellini. However, he is not in your report today. What had happened? Was he purged from your collection?

Penny:

Sorry, sir. No, we didn't purge any data from our database since we opened the shop. The likely reason why he is not in my report is because I only showed the directors who got any movies in our collection. Perhaps Fredricco Fellini may not have any movies in our collection. Let me run it again with _all_ directors.

Hint: In this case, you may need to use LEFT OUTER JOIN with Director table on the left side of the join.

Assist Penny to modify the previous query so all directors are shown.

Penny:

How is this list? Now we have all directors on the list. You can see your favourite director Fredricco at the bottom of the list.

Jerry:

Sorry, Fredricco Fellini is not my favourite director! Just checking why he was not on your original list. By the way, now your report got that funny gobbledygook symbol NULL. I do not know what you computer geeks think about it but makes no sense to me. Sure, Fredricco did not receive any awards, but, rather than those NULLs you can say that he got zero awards.

Penny:

[whispers to herself] Oh great. Now I have to think of a way to getting rid of NULL values and replacing them with zeros. I am sure there must be a way, let me think.

Hint: You can improve the output by using two SQL queries combined using a SET operator. Write one query that produces results for directors who did have movies (like the first query) and then UNION it with another query that just show the directors who didn't have any movies, hence no nominations or awards. The IS NULL condition in WHERE clause of the second query will filter the rows that related to directors with no movies.

Write a SQL statement to assist Penny to obtain the required data from the database.

Penny:

Finally, I managed to get the report as you asked. However, it was not something I do regularly. But, you are a special customer.

Jerry:

You are a genius. Sorry to be a pain, but your directors list is too long. I do not care about those hundreds of directors who have never even received a nomination, let alone winning an award. Can you trim your list to only directors who had at least received a nomination for directing a great movie.

Penny:

No problem. I know how to do it. My teachers at uni had taught us how to do that kind of reports.

Hint: Now that you have to filter rows from your result set, you may think of using a WHERE clause like WHERE SUM(m.noms) >= 1 However, when you try something like:

```
SELECT d.dirname, SUM(m.awrd), SUM(m.noms)
    FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
    GROUP BY d.dirname
    WHERE sum(m.noms) >= 1;
```

You will get an error message stating that you cannot use a grouped aggregation in WHERE clause. In this case, you have to evaluate the filtering condition on each group, not on each row in your joined input. To evaluate a condition on the group (i.e. condition on GROUPed result), we use HAVING clause.

Write a SQL statement to assist Penny to obtain the required data from the database.

Jerry:

Fantastic. That's exactly how I wanted. Now, can I ask something a little more intriguing? Can I get a list of directors who got awards above the average?

Penny:

That's tricky, but can be done. What I would do is to first find the average number of awards for the directors in my collection, and then run through the list to choose the ones above that magic number.

Hint: For this one, we have to use a sub-query that produces the average number (of awards) across the board for all directors, then use it to compare with the GROUPed average for each director in the outer query.

Write a SQL statement to assist Penny to obtain the required data from the database.

At this point, the store manager, Anna, reached out to Penny for some help.

Anna:

I am very sorry to interrupt you while serving a customer. I won't take a lot of your time. While I was working on our special promotion, I suspect that some of our customers have created multiple accounts. They may have used different addresses to create these accounts. They know that we not only use their name but both name and address to cross-check them when they apply for new accounts. Penny, can you find out the details of such customers who used the same name but used different addresses to disguise themselves.

Penny:

No problem Anna. Let me find out. Excuse me, Jerry, I will be back in a moment.

Penny:

Just to make it clear that we are looking for customer records in our database who got the same name but different addresses.

Hint: To find the details of such members, we have to create a sub-query that produces a list that contains only the names of such members, and then the outer query will show full details of such members. The inner query must use grouped aggregation (i.e. grouped by name and identify the groups with more than one record) and a HAVING clause to find out any clients who got multiple records.

Write a SQL statement to assist Penny to obtain the required data from the database.

Note that our data set does not have such dodgy members. So, to test your query, make a minor change to one of the member details

```
UPDATE member
    SET mmbname = 'Stein, Billy'
    WHERE mmbname = 'Stein, Willy'
```

Now member records 11 and 13 belong to the same Billy Stein. Then, when you try your query, it should show details of Billy Stein.

Penny:

Sorry about that Jerry, how can I help you?

Jerry:

Sorry, I took a lot of time asking about directors. Now I recall why did I come to the video store in the first place. My kids are going to have a sleepover with their friends. Can I choose some comedies that are not rated R. I guess you know what I mean?

Penny:

Sure, I can help you out.

Hint: In your previous activities, you have used multiple conditions in the WHERE clause for this kind of query. In this activity, try it out with SET operations. Rephrase the query as follows:

```
Find all comedy movies
Except
the movies rated R.
```

Write a SQL statement to assist Penny to obtain the required data from the database.

Jerry:

Thank you. I will borrow a few of those comedies. While we are here, I wish to borrow a few for myself. Today, I would borrow Woody Allen movies. I know he had starred himself in some of them. Such a multi-talented fellow. I would like to borrow any of such movies that he both directed and starred himself.

Penny:

Of course. I can help you find any such movies in our collection.

Hint: If you try to write a single query to the above is not straightforward. Two separate queries can be built to find (1) Woody Allen- directed movies and (2) movies where Woody Allen appeared as a star. Finally, use the set operator INTERSECT to find a set of rows that are in both result sets. Write a SQL statement to assist Penny to obtain the required data from the database.

Jerry:

Thank you. Please add them to my cart. While we are on it, can you find the Woody Allen movies in which he didn't appear.

Penny:

Since we worked out the one he appeared just now, It is easy to find the others.

Hint: Just replace INTERSECT with EXCEPT.
Write a SQL statement to assist Penny to obtain the required data from the database.