

JDBC Exercises

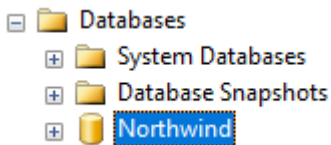
2017 © Edument AB

Exercise 1 – Getting to know JDBC terminology

1. First, we need to become more familiar with **JDBC**, read the following resources:
 - a. https://en.wikipedia.org/wiki/Java_Database_Connectivity
 - b. Try to figure out to what databases JDBC can connect to
2. Another term that you will see from time to time is **ODBC**, do read up about it and see if you can compare the differences between **JDBC** and **ODBC**.
3. Visit <https://www.connectionstrings.com/> and read more about **connection strings**. Do explore the various connection string options that exists for SQL Server.
4. Visit the page “Microsoft JDBC Driver for SQL Server” and get familiar with it, it can be handy to have later.
<https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server>

Exercise 2 - Setup the database

1. In this exercise, we will be using a database named **Northwind** that you can download from here:
<https://www.microsoft.com/en-us/download/details.aspx?id=23654>
2. Run the installer and then the database files will be located in
C:\SQL Server 2000 Sample Databases
3. Double-click on the file **instnwnd.sql** so that it opens in SQL Management Studio (SSMS) and **connect** to your local SQL Server Express database at **.\sqlexpress**

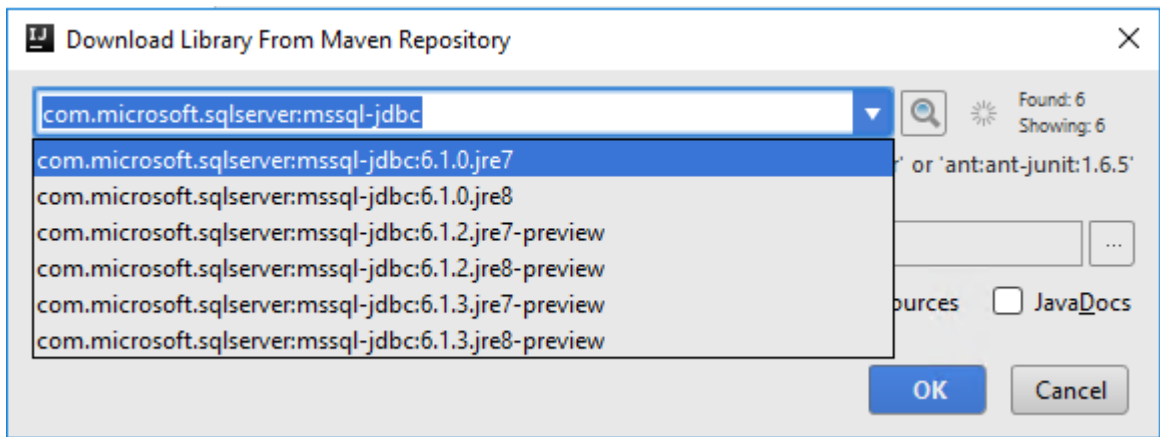
Name	Date modified	Type	Size
instnwnd	2/4/2004 8:45 AM	Microsoft SQL Ser...	2,066 KB
instpubs	3/23/2004 10:50 AM	Microsoft SQL Ser...	126 KB
NORTHWND	12/13/2004 4:14 PM	SQL Server Databa...	1,024 KB
NORTHWND	12/13/2004 4:14 PM	SQL Server Databa...	2,688 KB
PUBS	12/13/2004 4:14 PM	SQL Server Databa...	1,280 KB
PUBS_LOG	12/13/2004 4:14 PM	SQL Server Databa...	768 KB
ReadMe_SQL2000SampleDbScripts	3/23/2004 10:50 AM	HTM File	61 KB
4. Press **Execute** to run the script, if you see any “Could not find stored procedure 'sp_dboption'.” Errors, then just ignore those.
5. Refresh the database list and you should now see a new database named **Northwind** here:
6. Navigate and look at the content of the tables in the database to familiarise yourself with the tables and content.

Exercise 3 – Connecting to the database

In this exercise, we will learn how to connect to our database, open and then close the connection just to make sure we can talk to the database.

This exercise assumes you have a SQL Server Express located at `.\sqlexpress` and you have installed the **Northwind** database.

1. Start a new IntelliJ command-line project
2. Add the following library from Maven:
`com.microsoft.sqlserver:mssql-jdbc`



Choose version **6.1.0.jre8** or higher.

3. First, we need to provide a connection string:

```
String connstr =  
"jdbc:sqlserver://localhost;instanceName=sqlexpress;databasename=Northwind;"
```

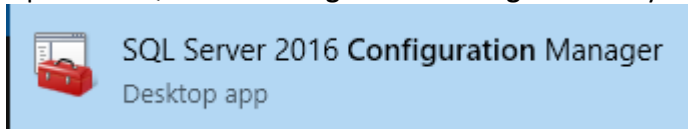
4. Then we add the following code where we open a connection and close it, just to make sure it works! Make sure you understand the code, google if you are unsure!

```
Connection dbconn = null;
try {
    dbconn = DriverManager.getConnection(connstr);

} catch (SQLException e) {
    e.printStackTrace();
}
finally {
    if(dbconn!=null)
        dbconn.close();
}

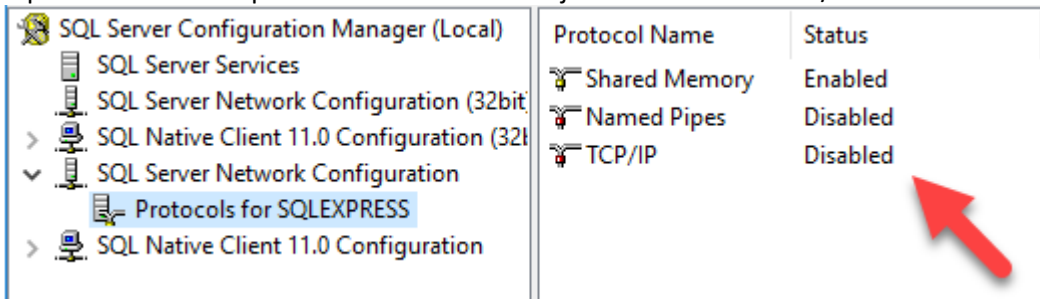
System.out.println("Done");
```

5. Run the program and you will probably get an **exception**. Explore the reason in the exception.
6. The problem is that SQL Server Express by default don't allow connections via TCP/IP. To enable this, we need to enable it.
7. Open the **SQL Server Configuration Manager tool** on your computer:



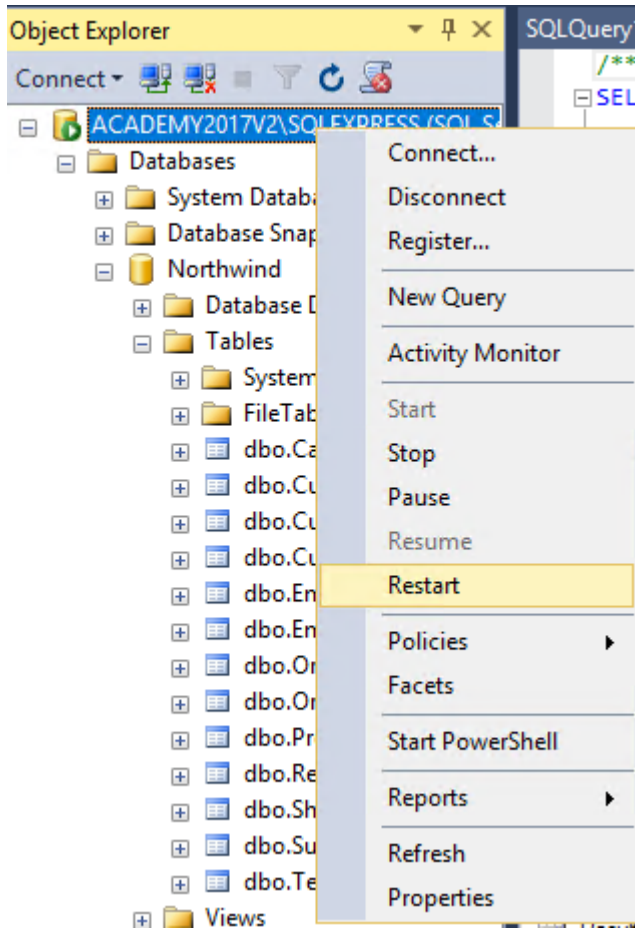
If you can't find it, then visit: <https://stackoverflow.com/questions/9844771> or search on your computer for **SQLServerManager13.msc** if you are running Sql Server 2016 express.

8. Explore the various options in the tool and our job now is enable TCP/IP here:

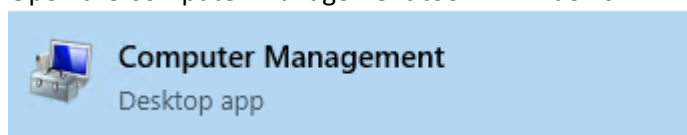


Double-click on it and enable TCP/IP.

9. We now need to **restart** SQL-Server by right-click on the server in the **SQL Server Management Studio** here:

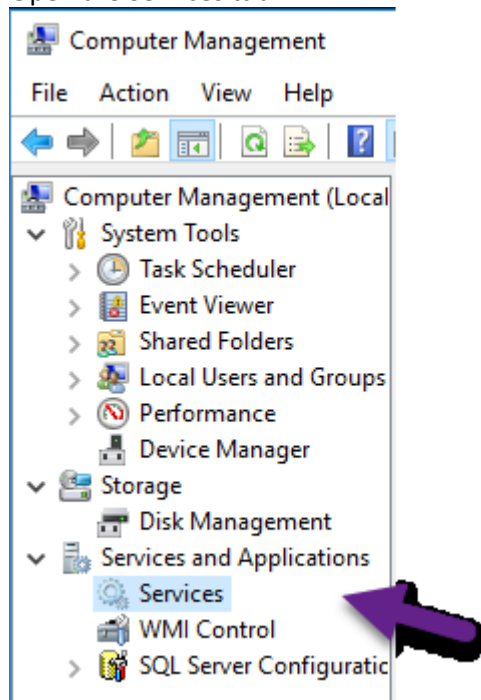


10. Run the program again, and you still see that the program crashes.
11. The second problem we need to fix is to start the SQL Server Browser Windows service.
- First google and read what **SQL Server Browser service** does or visit:
[https://technet.microsoft.com/en-us/library/ms181087\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms181087(v=sql.105).aspx)
 - If you don't know what **Windows Services** are, then google and read about it or visit:
 - [https://msdn.microsoft.com/en-us/library/d56de412\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/d56de412(v=vs.110).aspx)
 - https://en.wikipedia.org/wiki/Windows_service
 - <http://www.howtogeek.com/school/using-windows-admin-tools-like-a-pro/lesson8/>
 - Open the Computer Management tool in Windows:



(Or open **Control Panel\System and Security\Administrative Tools** in the Control Panel)

- d. Open the Services tab:



- e. Double-click on the SQL Server Browser service:

Software Protection	Enables the ...	Running	Author...
Spot Verifier	Verifies pote...		Manu...
SQL Server (SQLEXPRESS)	Provides sto...	Running	Author...
SQL Server Agent (SQLEXPRESS)	Executes jo...		Disabl...
SQL Server Browser	Provides SQ...		Disabl...
SQL Server CEIP service (SQL Server)	CEIP service...	Running	Author...
SQL Server VSS Writer	Provides th...	Running	Author...
SSDP Discovery	Discovers n...	Running	Manu...
State Repository Service	Provides re...	Running	Manu...

- f. Then set the **Startup type** to **Automatic (Delayed Start)** and then press **Apply**

Path to executable:

"C:\Program Files (x86)\Microsoft SQL Server\90\Shared\sqlbrowser.exe"

Startup type:

Automatic (Delayed Start)

- g. Google and learn about the various **Startup type** options
h. Then press the **Start** button to start the service, make sure it starts!

Service status: Stopped

Start

Stop

Pause

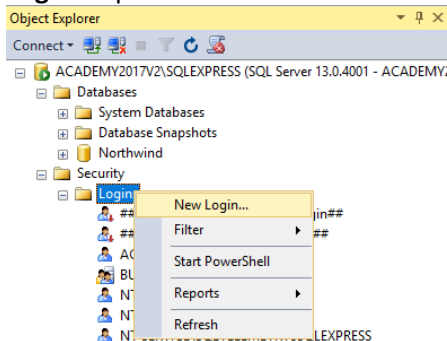
Resume

12. Run the application again and you now get this error:

```
com.microsoft.sqlserver.jdbc.SQLServerException: Login failed for user ".  
ClientConnectionId:f05267c3-48fd-4db7-a8af-2edecb57605
```

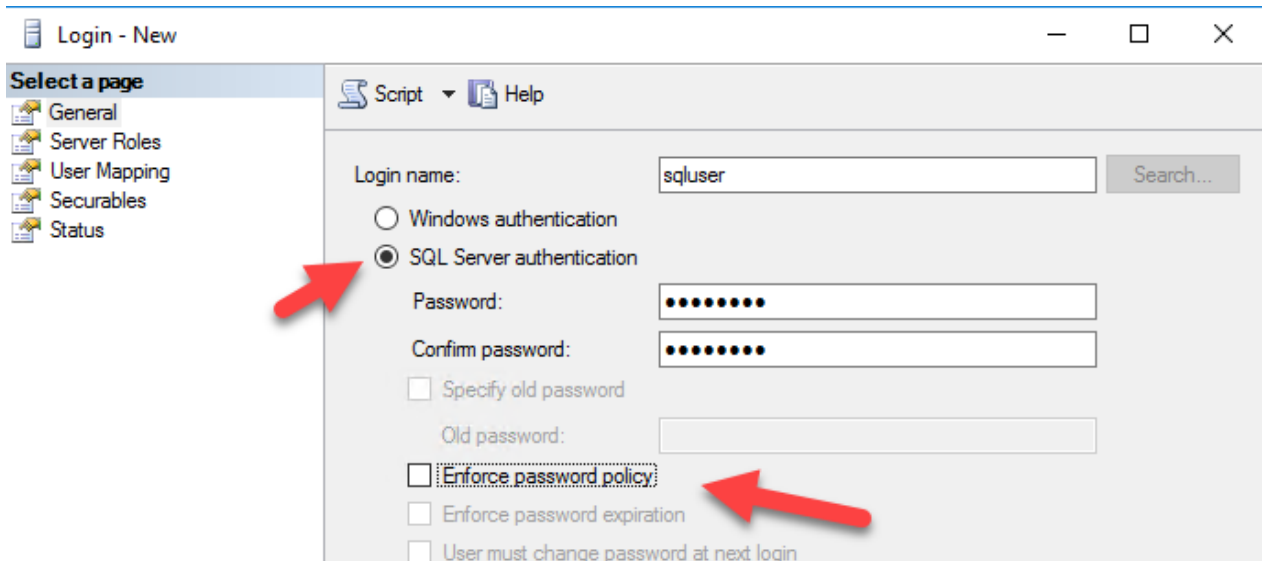
13. We could use **Integrated security** to solve this but let's instead create a local user in our database so that we learn how that works.

Open SQL Management Studio and open the **Security -> Login** folder and right click on the **New Login...** option.

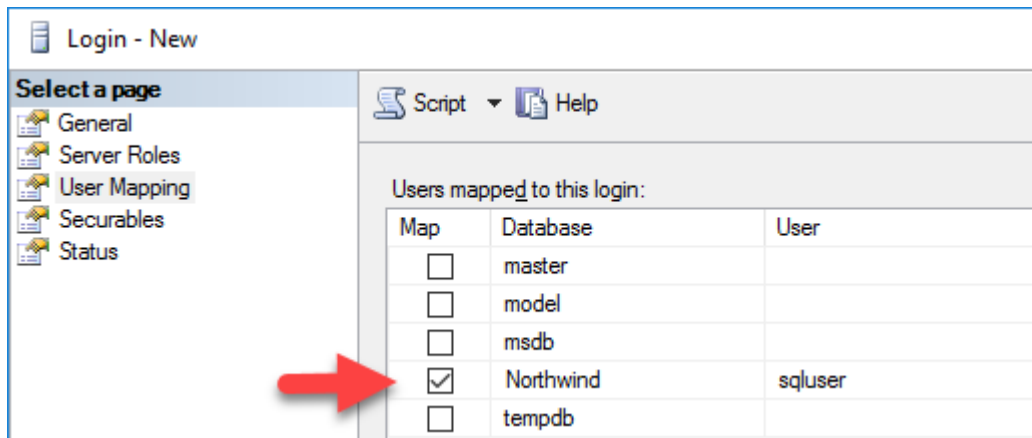


14. Select SQL Server authentication and enter Login name **sqluser** and a password of your choice, like **password**.

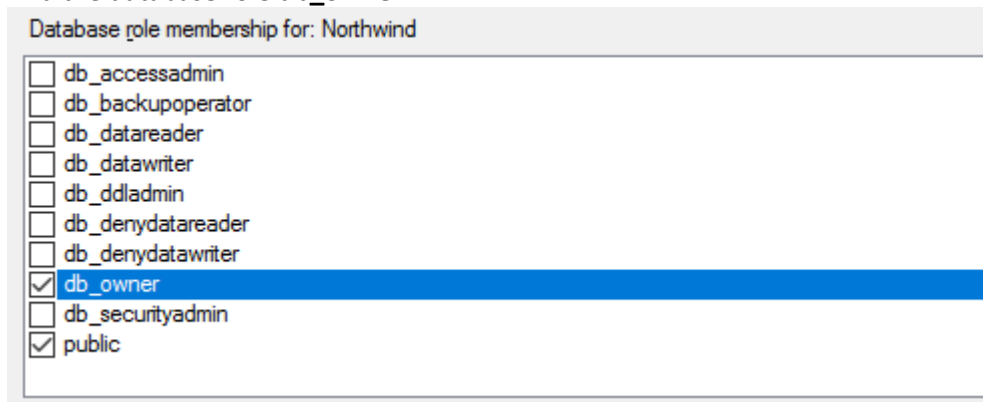
Also uncheck the **Enforce password policy**



15. Then press the User Mapping page in the left column and give this user access to the **Northwind** database:

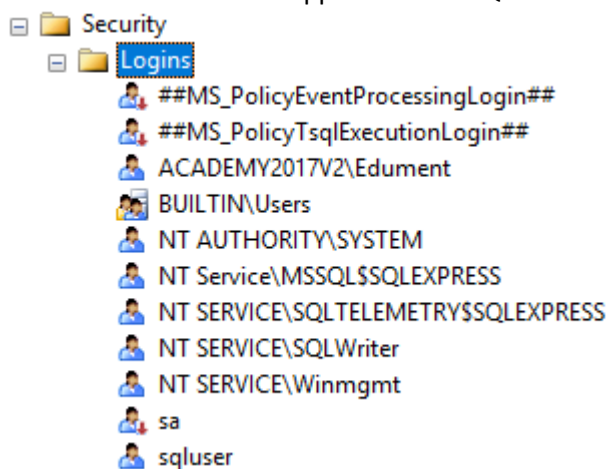


And the database role **db_owner**:



Press OK to create the new user.

16. Make sure the new user appears in the SQL Server logins list:

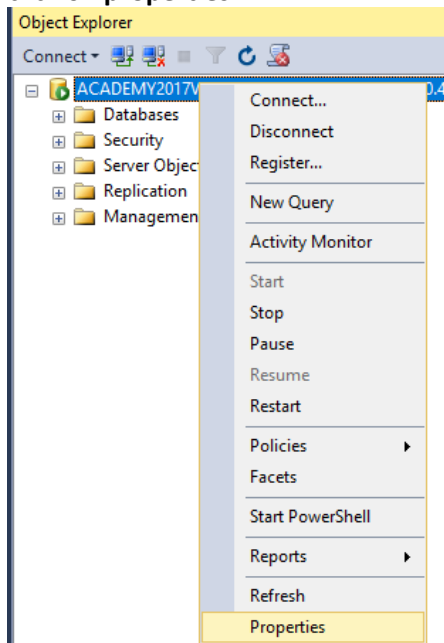


17. We now need to add the following to our connection string:

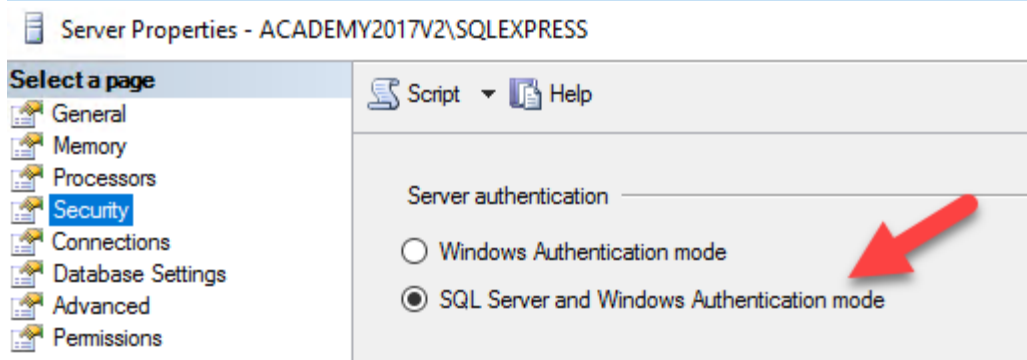
```
String connstr =  
"jdbc:sqlserver://localhost;instanceName=sqlexpress;databasename=Northwind;  
user=sqluser;password=password";
```

(use the password you used when creating the user)

18. Run the application and you still see the application crash! This is because we also need to enable **SQL Server and Windows Authentication** mode by right click on the SQL Server here and click on **properties**:



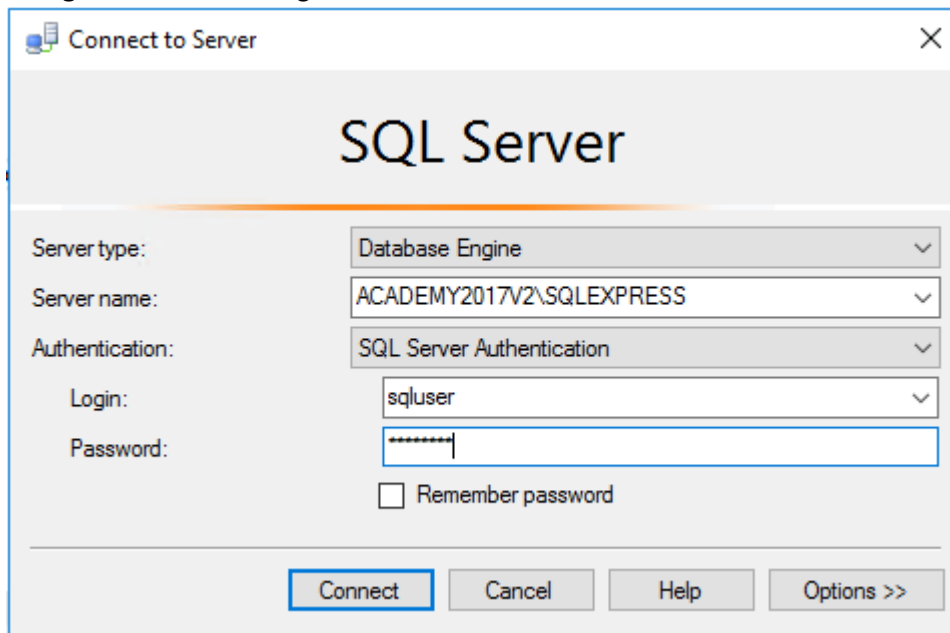
And then choose under Security the **SQL Server And Windows Authentication mode**:



Right click on the database and restart it again!

19. Now run the application again and now it should work!

A good way to diagnose user or connection problems is to connect to the database using **SQL Management Studio** using **SQL Server Authentication** mode:



If that works you know that the user account is good and the security is correct.

Exercise 4 – Hello world!

In this exercise, we will send a simple SELECT query to the database and back, just to verify that we can some data back from the database.

A good tutorial is to look at is <https://docs.oracle.com/javase/tutorial/jdbc/> , do check it out and use it as a reference during these exercises.

1. Use the code from the previous exercise.
2. Query using SQL Management Studio how many customers there are in the **Customers** table in the **Northwind** database.
3. Write the code that counts the number of customers in the **Customers table** and prints out the result to the screen. Compare the result with the previous step. You will find example code on how to do this here: <https://docs.oracle.com/javase/tutorial/jdbc/>

4. In the previous step you probably used the `rs.getInt(1);` method to get the value from the `ResultSet`.

Now let's modify the SQL query so that you can use this code to get the result: (hint use aliases)

```
int count = rs3.getInt("NumberOfCustomers");
```

Exercise 5 – Slightly more advanced select.

1. We need to print out the details for **EmployeeID=9** in the Employee table:

Modify the code from the previous exercise so that it prints out the following fields:

	EmployeeID	LastName	FirstName	Title	Country
1	9	Dodsworth	Anne	Sales Representative	UK

It could print out something like:

```
ID 9
LastName Dodsworth
FirstName Anne
Title Sales Representative
Country UK
Done
```

Exercise 6 – Advanced select with join

We need to print out all the territories that **Anne Dodsworth** (ID=9) is responsible for.

1. Create a SQL statement in SQL management studio that (using joins) results in the following output:

Results		Messages			
	EmployeeID	LastName	FirstName	TerritoryID	TerritoryDescription
1	9	Dodsworth	Anne	03049	Hollis
2	9	Dodsworth	Anne	03801	Portsmouth
3	9	Dodsworth	Anne	48075	Southfield
4	9	Dodsworth	Anne	48084	Troy
5	9	Dodsworth	Anne	48304	Bloomfield Hills
6	9	Dodsworth	Anne	55113	Roseville
7	9	Dodsworth	Anne	55439	Minneapolis

2. Use the code from the previous exercise and add the query from the previous step.
3. Modify the code so that it prints the details as presented above to the console, something like this:

```
ID, LastName, FirstName, TerritoryID, TerritoryDescription
9, Dodsworth, Anne, 3049, Hollis
9, Dodsworth, Anne, 3801, Portsmouth
9, Dodsworth, Anne, 48075, Southfield
9, Dodsworth, Anne, 48084, Troy
9, Dodsworth, Anne, 48304, Bloomfield Hills
9, Dodsworth, Anne, 55113, Roseville
9, Dodsworth, Anne, 55439, Minneapolis
Done
```

Exercise 7 – Exploring the ResultSet

Let's explore the ResultSet more in detail.

1. Create a new project and copy the code from the previous exercise and make sure it runs.
2. The **ResultSet** returned from **executeQuery()** contains a method named **.getMetaData();**, let's explore this method!
3. Read about this method here:
[http://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html#isSearchable\(int\)](http://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html#isSearchable(int))
(or Google for "jdbc ResultSetMetaData")
4. Write the necessary code that loops through all the found columns in the **ResultSetMetaData** and prints out the following:

```
Columns 5  
Column EmployeeID (int)  
Column LastName (nvarchar)  
Column FirstName (nvarchar)  
Column TerritoryID (nvarchar)  
Column TerritoryDescription (nchar)
```

As you see the **ResultSet** contains quite a lot of additional data!

Exercise 8 – Prepared statements

1. Write a method with this signature:

```
public static String GetProductName(int productId) {  
}
```

Requirements:

- Must use **prepared statements**
 - Return **empty string ""** if the product is not found
 - Returns the **ProductName** from the **Products** table.
2. Verify that it works with code like:

```
System.out.println(GetProductName(1));  
System.out.println(GetProductName(2));  
System.out.println(GetProductName(99999999)); //should return ""
```

Exercise 9 – Prepared statements

Introduction

In this exercise, we will learn how to access a database using the **JDBC library**. The goal is to complete a number of **unit-tests** written so that they all pass. You do this by completing the **empty Repository class** found in the starter project that is given to you.



The exercise

1. Open JDBC exercise **9 Starter-kit** project
2. When you run the project, you will see that a number of unit-tests are automatically executed located in the test class.

As you see there are a lot of unit tests failing and your task is now to make sure they **all pass**.

3. Important:
 - You should only implement the necessary code in the file NorthWindRepo class file and add the necessary JDBC code to it, to make sure all the test pass.
 - The class **NorthwindRepoTest** should **not be modified**.
 - The **CreateNewOrder ()** method is the most complicated so do that one last.

If you have time

1. Create the **UpdateOrder** and **DeleteOrder** methods and suitable tests
2. Add the **UnitPrice** field to the **OrderLine** class and update the necessary repository methods.
3. Read more about **Junit** tests and try to add a few more tests that might be missing against the existing code. For example, tests that:
 - a. Returns exception if the input parameters are invalid, so that we don't do any DB queries if the provided parameters are invalid. Like negative ID's.
 - b. What if you try to create a new empty order with no customerID?
4. Refactor the code in the repository to better use the **try-with-resources** pattern
<https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>
5. Experiment with **transactions** when updating or creating an order. So that when you query will rollback if there's an error in the query or code. We don't want partial updates to the database.
6. **SQL Injection attack!** Make sure you understand this security vulnerability

Further Reading:

- SQLException documentation:
<http://docs.oracle.com/javase/8/docs/api/java/sql/SQLException.html>
- Discussion of Java and repository pattern at Stackoverflow:
<http://stackoverflow.com/questions/31305199/repository-pattern-how-to-understand-it-and-how-does-it-work-with-complex-en>
- Repository Pattern for Dummies:
<http://blog.sapiensworks.com/post/2014/06/02/The-Repository-Pattern-For-Dummies.aspx>
- Using Transactions
<https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html>