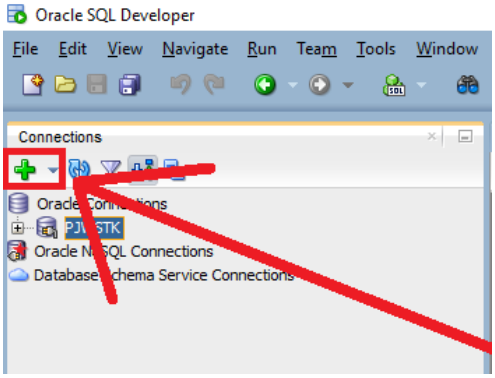
WSI SQL Tutorial

In today's lesson, we'll learn basic SQL commands. In this file you will find SQL query templates, which I recommend to test on a sample database. These templates contain elements such as "column" and "table" that need to be renamed to the names of columns and tables that exist in the database.

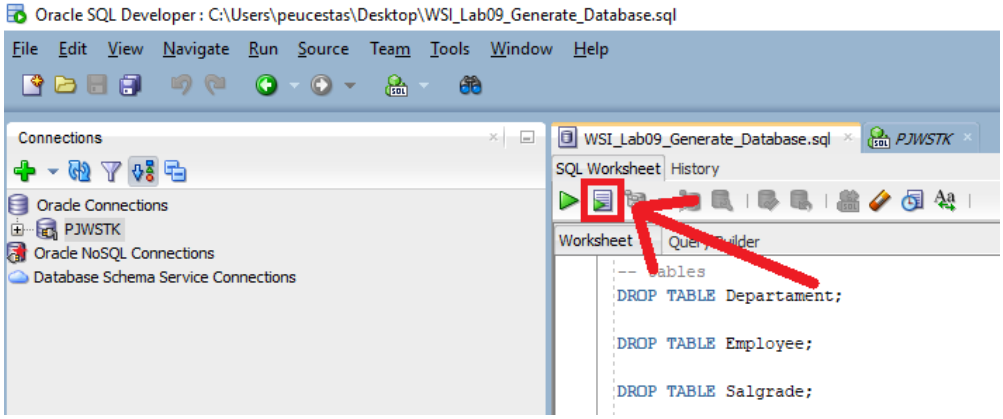
Connecting to the database

During today's lesson, we will be using SqlDeveloper, which can be found on disk G or R, in the "Program Files" folder. You have to give it some time to load, because it works rather slowly. Once it loads, in order to use it, we need to establish a connection to the database. To do this, click on the green plus in the upper left corner. In the window that appears, you need to fill in fields as it is shown **here (connecting to the Oracle database).**



Generating a sample database

After establishing the connection, we can generate a sample database: **A file to generate the database**.

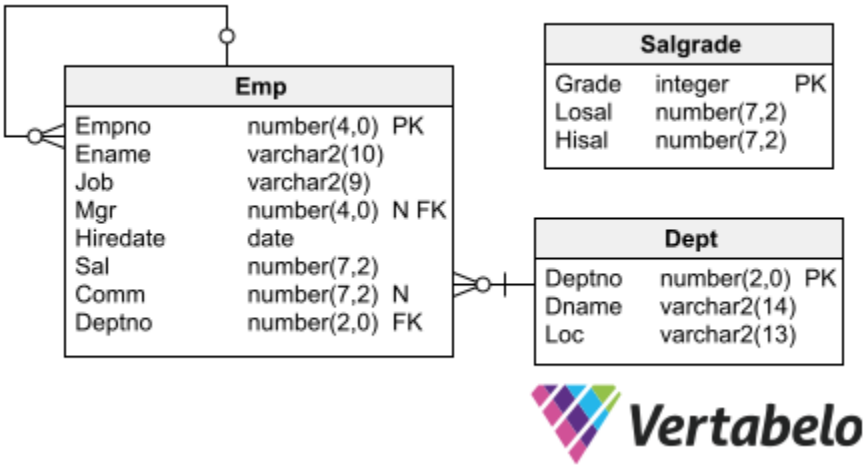


The "demobld.sql" file can be opened in Files -> Open and then click the "run script" button to generate a database.

If the tables didn't appear right away, you'll need to reset the connection.

Working with a sample database

Our sample database, which we just generated in SqlDeveloper, consists of 3 entities: Emp, Dept and Salgrade. Below you can see it's ERD.



The tables are filled with the following data:

Data in the Emp.

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980-12-17	800	(null)	20
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
4	7566	JONES	MANAGER	7839	1981-04-02	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850	(null)	30
7	7782	CLARK	MANAGER	7839	1981-06-09	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	1981-11-17	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
11	7876	ADAMS	CLERK	7788	1983-01-12	1100	(null)	20
12	7900	JAMES	CLERK	7698	1981-12-03	950	(null)	30
13	7902	FORD	ANALYST	7566	1981-12-03	3000	(null)	20
14	7934	MILLER	CLERK	7782	1982-01-13	1300	(null)	10

Data in the Dept.

	DEPTNO	DNAME	LOC
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON

Data in the Salgrade.

	GRADE	LOSAL	HISAL
1	1	700	1200
2	2	1201	1400
3	3	1401	2000
4	4	2001	3000
5	5	3001	9999

Legend for sample database

Emp (Employee) Table

Empno - employee id
Ename - employee's name
Job - employee's position
Mgr - boss id
Hiredate - date of employment
Sal - salary (monthly)
Comm - commission (annual)
Deptno - id of the department given employee is working in

Dept (Department) Table

Deptno - department id
Dname - the name of the department
Loc - location (city) of the department

Salgrade Table

Grade - grade number
Losal - minimum wage in a grade
Hisal - maximum pay in a grade

0. SQL query writing convention

- SQL commands are written with capital letters such as: **FROM**, **SELECT**, **WHERE**.
- The names of columns and tables are written in lowercase letters, such as: emp, dept.

1. Reading data (SELECT FROM)

To read the data from the database, we need to define a column and table from which we want to read the values. To define a column, use **SELECT** command, and to define a table, use **FROM** command.

1.1 Reading values from one column in a given table

```
SELECT column
FROM table;
```

1.2 Reading values from multiple columns in a given table

```
SELECT column_1, column_2, column_3
FROM table;
```

1.3 Reading values from all columns in a given table

```
SELECT *
FROM table;
```

2. Custom column names (SELECT AS)

To give a custom name to the displayed column, we can use the **SELECT AS** command. If the name consists of several words, it is necessary to enclose it in quotation marks.

2.1 Labeling a column with a single word

```
SELECT column AS name
FROM table;
```

2.2 Labeling a column in a few words

```
SELECT column AS "column name"
FROM table;
```

3. Display data from different columns in one column.

By reading data from multiple columns, we can combine them so that they are displayed in a common column. We can use **||** operator for this .

3.1 Display data from 2 columns in a common column

```
SELECT column_1 || column_2
FROM table;
```

3.2 Display data from 2 columns in a common column with additional text

```
SELECT column_1 || ' text between data' || column_2
FROM table;
```

4. Reading values without repetition (SELECT DISTINCT)

In a given table, one value can occur multiple times (unless it is the primary key). If we want to display the values without repeating them, we can use **SELECT DISTINCT** command.

4.1 Display data from the selected column without repeating it

```
SELECT DISTINCT column
FROM table;
```

4.2 Display different combinations of values from 2 columns, without repeating them

```
SELECT DISTINCT column_1, column_2
FROM table;
```

5. Sort the displayed data (ORDER BY)

To determine the order of the displayed records, we can use the **ORDER BY** command. In addition, we can sort descending or ascending by adding **DESC** or **ASC**.

5.1 View sorted values from a column

```
SELECT column
FROM table
ORDER BY column;
```

5.2 Display all table values sorted by a given column

```
SELECT *
FROM table
ORDER BY column;
```

5.3 Display values from a column sorted first by one column and then by another column, without displaying the other column

```
SELECT column_1
FROM table
ORDER BY column_1, column_2;
```

5.4 Display table values sorted by a column in ascending order

```
SELECT *
FROM table
ORDER BY column ASC;
```

5.5 Display values sorted by a given column in descending order

```
SELECT *
FROM table
ORDER BY column DESC;
```

5.6 Display values sorted first by one column in descending order, then by the second column in ascending order

```
SELECT *
FROM table
ORDER BY column_1 DESC, column_2 ASC;
```

6. Display data with specific values (WHERE)

6.1 Displaying data where agiven tribute has a specific value or name

```
SELECT *
FROM table
WHERE column = 27;
```

```
SELECT *
FROM table
WHERE column = 'TEXT';
```

6.2 Displaying data where an attribute does not have a specific value or name

```
SELECT *
FROM table
WHERE NOT column = 27;
```

```
SELECT *
FROM table
WHERE NOT column = 'TEXT';
```

6.3 Displaying data where a given attribute has a value greater than 0

```
SELECT *
FROM table
WHERE column > 0;
```

6.4 Display data where an attribute has a greater value than an attribute from another column

```
SELECT *
FROM table
WHERE column_1 > column_2;
```

7. Displaying data about values about which we have incomplete information (% , _ , LIKE)

Sometimes we want to find values in the database with a certain number of characters or containing specific letters, then we can use **%**, **_** and **LIKE** operators.

_ - one unknown character
% - many number of unknown characters

```
SELECT *
FROM table
WHERE column LIKE 'TE%';
```

OR

```
SELECT *
FROM table
WHERE column LIKE 'TE__';
```

8. Logical functions (AND, OR)

8.1 Displaying data where a certain attribute takes one of the given values.

```
SELECT *
FROM table
WHERE column_1 = 'TEXT1' OR column_1 = 'TEXT2';
```

8.2 Displaying data where given attributes take the given values

```
SELECT *
FROM table
WHERE column_1 = 'TEXT1' AND column_2 = 'TEXT2';
```

8.3 Data display conditioned by a multi-factor logical function

```
SELECT *
FROM table
WHERE ( column_1 > 1 AND column_1 < 10 ) OR column_2 = 'TEXT';
```

9. Displaying data from the given range (WHERE BETWEEN)

9.1 Displaying data where the value of a given column is in a numeric range

```
SELECT *
FROM table
WHERE column_1 > 10 AND column_1 < 20;
```

OR

```
SELECT *
FROM table
WHERE column BETWEEN 10 AND 20;
```

9.2 Displaying data where the value of a given column is in a numeric range

```
SELECT *
FROM table
WHERE column_1 < 10 OR column_1 > 20;
```

OR

```
SELECT *
FROM table
WHERE NOT column BETWEEN 10 AND 20;
```

10. Displaying data from the given pool (WHERE IN)

```
SELECT *
FROM table
WHERE column_1 = 'TEKST1' OR column_1 = 'TEXT2' OR column_1 = 'TEKST3';
```

OR

```
SELECT *
FROM table
WHERE column_1 IN ( 'TEXT1', 'TEXT2', 'TEXT3' );
```

11. Display the length of a given value (WHERE LENGTH)

11.1 Displaying the Length of a Value

```
SELECT *
FROM table
WHERE LENGTH ( column ) = 5;
```

OR

```
SELECT *
FROM table
WHERE column LIKE '_____';
```

12. Operations on values with nullable property.

12.1 Displaying values that are NULL

```
SELECT *
FROM table
WHERE column IS NULL;
```

12.2 Displaying values that are not NULL

```
SELECT *
FROM table
WHERE column IS NOT NULL;
```

13. Calculations on retrieved data

On the read data we can make various calculations, then we can display them modified with a given numerical operation. The data in the database remains unchanged, we simply define how it is displayed.

13.1 Display data modified with a sample arithmetical operation

```
SELECT ( column + 25 ) * 2
FROM table;
```

```
SELECT column_1 - column_2
FROM table;
```

14. Calculations on data including Null values.

When we want to perform arithmetical operations on values with the nullable property, we encounter a problem, because performing arithmetic operations on NULL values also gives us NULL (eg. NULL + 2 = NULL). To avoid this, we need to use the **NVL()** function.

NVL(column , 0) – the first argument is the column that we want to use for calculations, the second argument is the value that we want to replace NULL with.

```
SELECT  column_1 + NVL( column_2 , 0 )
FROM table;
```

15. Date operations

```
SELECT *
FROM table
WHERE column >= '01/01/80'  AND column <= '31/12/80';
```

```
SELECT *
FROM table
WHERE EXTRACT (YEAR FROM column) = 1980;
```