

# Introduction to The Oracle Database Software

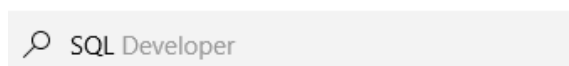
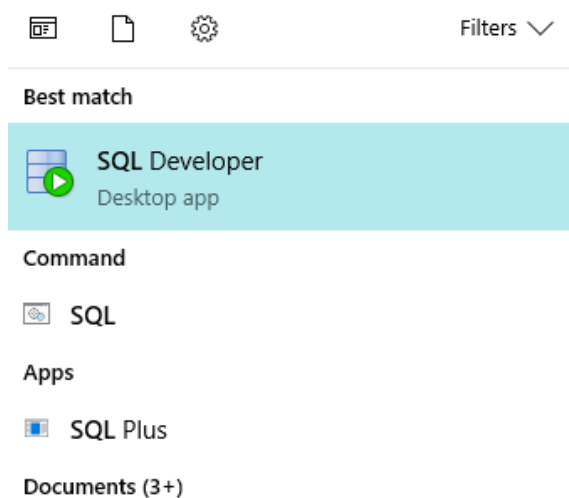
---

## Aims and Objectives

In this lab the students learnt the following things:

- How to use the SQL developer
- How to create a new connection in the SQL developer
- How to create tables in the SQL developer

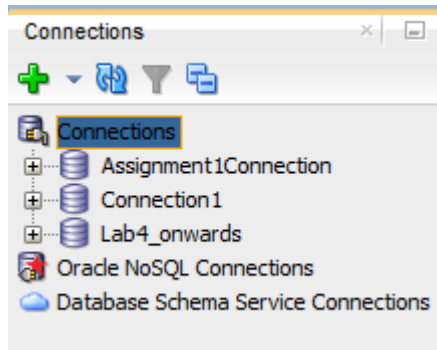
In the Windows search bar type SQL Developer, open the SQL developer by clicking on it.



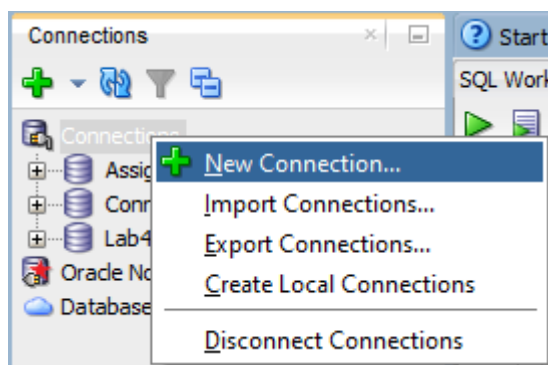
After the installation of the Oracle SQL developer, when you open it the very first time, the SQL developer asks you for the java path.

## Creating a new connection in database

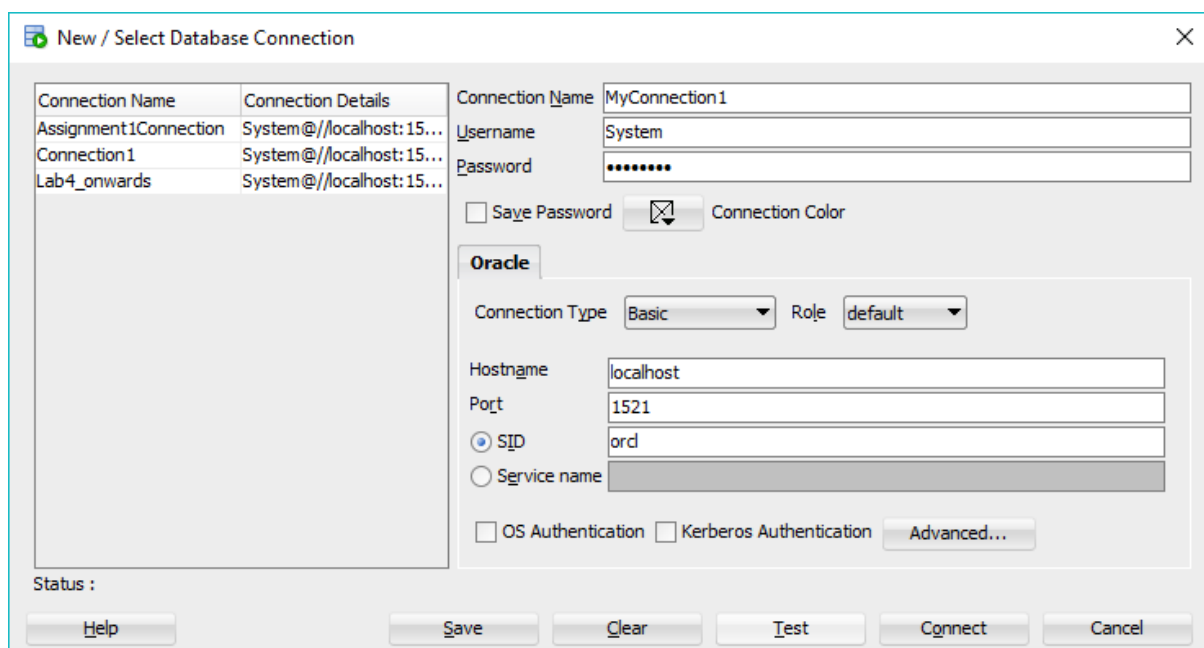
When we install the Oracle the very first time, no connection has been established. So that we go to the “Connections” in the left navigation bar. Right click on it, and select the option “New Connection”.



*(As, I had worked before on Oracle that's why other connections are showing here. On the first-time installation, no connection would be showing here.)*

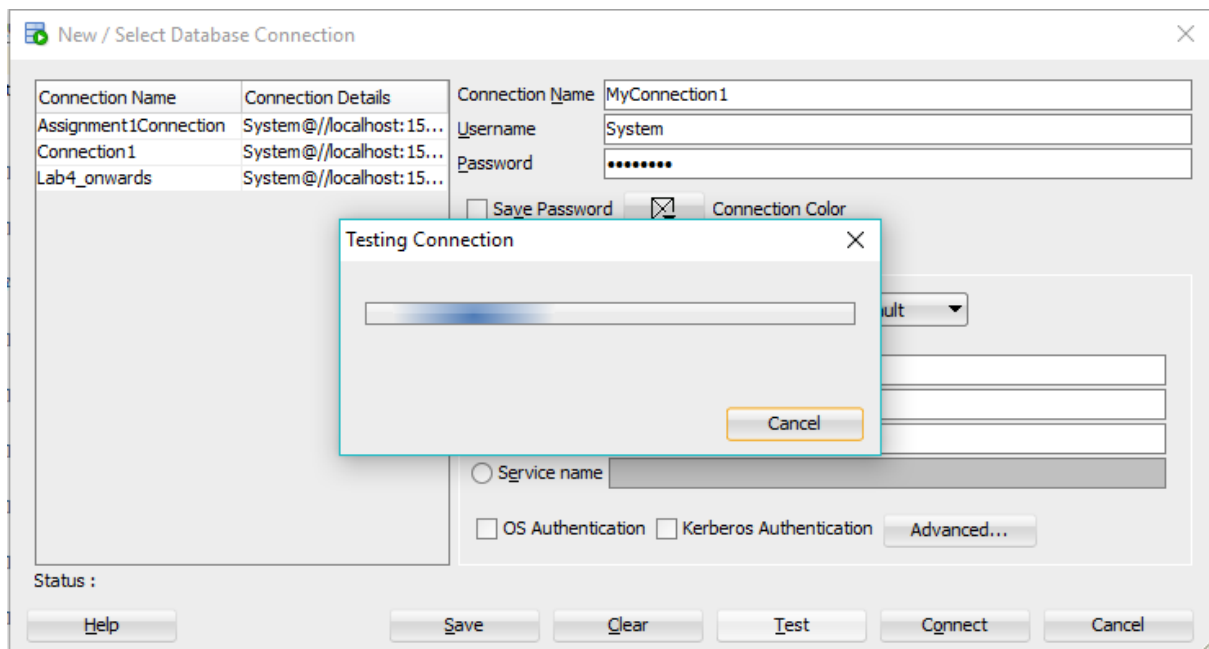


After we select the “New Connection” option a screen appears, in which we have to fill the connection details.



Here, you have to give a name to your connection. I gave the name to my connection as 'MyConnection1'. For now, as we have to log in to the database as the System administrator, so that in the "Username" column we enter 'System' and as we had entered the default administrator password as 'Oracle\_1' so that we enter this password in the "Password" column. In the end we change the SID name to the one that we had kept at the time of Oracle installation. As, at the time of my Oracle installation, I had kept it 'orcl' so in the "SID" column I entered the same name.

After filling the connection details, you have to click "Test" button. This checks that if the formation of this new connection would be successful or not. If 'Success' appears in front of the "Status:" title, it means that the connection would be established successfully, otherwise if an error appears here, it means that there has been some mistake or other error while creating the connection and we should try establishing the connection again with our database.



The screenshot shows the 'New / Select Database Connection' dialog box. On the left, a table lists existing connections:

Connection Name	Connection Details
Assignment1Connection	System@//localhost:15...
Connection1	System@//localhost:15...
Lab4_onwards	System@//localhost:15...

The 'Status' is 'Success'. The 'Test' button is highlighted in yellow.

On the right, the 'Oracle' tab is selected. The 'Connection Name' is 'MyConnection1'. The 'Username' is 'System' and the 'Password' is masked with dots. The 'Save Password' checkbox is unchecked. The 'Connection Color' is set to a default color. The 'Connection Type' is 'Basic' and the 'Role' is 'default'. The 'Hostname' is 'localhost', the 'Port' is '1521', and the 'SID' is 'ord'. The 'Service name' is empty. The 'OS Authentication' and 'Kerberos Authentication' checkboxes are unchecked. The 'Advanced...' button is visible.

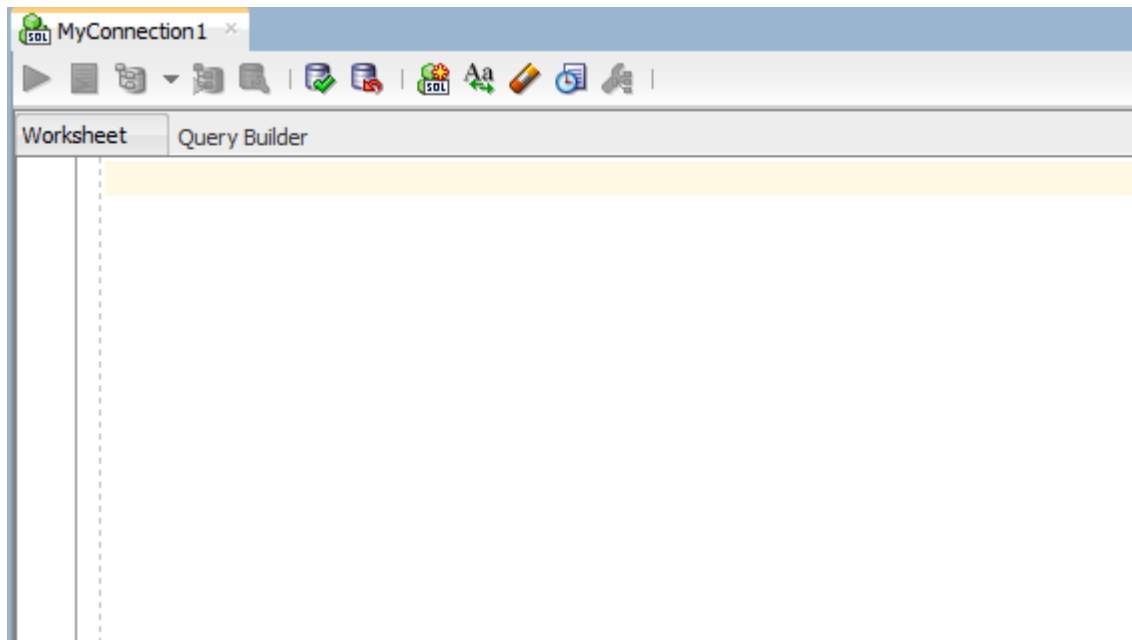
At the bottom, there are buttons for 'Help', 'Save', 'Clear', 'Test', 'Connect', and 'Cancel'.

After the success status we have to click the “Connect” button, so that we might be connected permanently to our database by this connection and do our work like creating tables or etc. in the database.

This screenshot is identical to the one above, showing the 'New / Select Database Connection' dialog box with the 'Status' as 'Success' and the 'Test' button highlighted.

Without creating a connection with the database, we can’t do anything like enter or retrieve data from our database.

When a successful connection has been established with our database, Oracle opens us a worksheet in which we can write the SQL queries and do some work in our database.



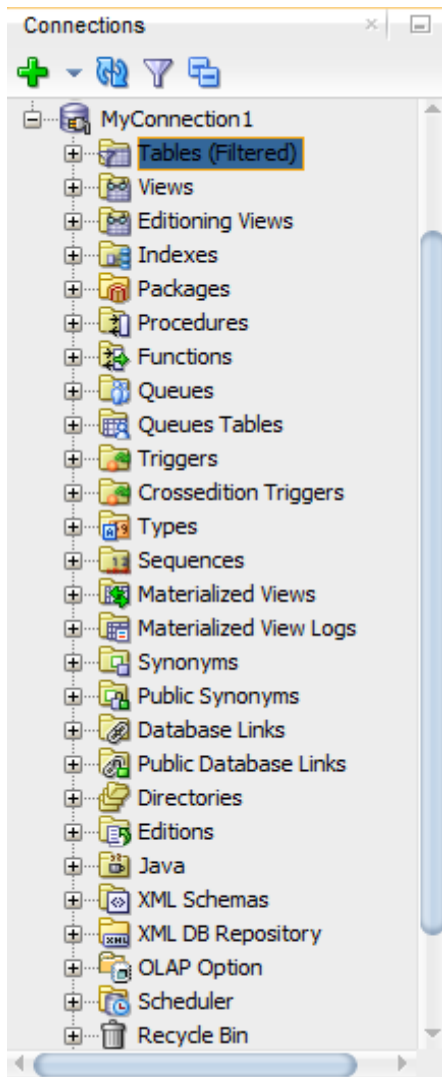
## Creating a table in database

We can create tables in our database by two methods.

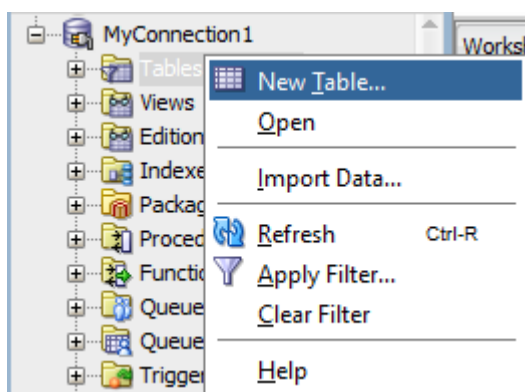
1. Through the Oracle GUI
2. Through the SQL query.

### Creating table by using GUI

For this, we expand our connection menu and select the “Tables” option.



After selecting the “Tables” option, right click it and select “New Table” option to create a new table.



Now a new panel appears, in which we have to fill the details to create a table. By default, whenever a table is created via GUI, the name “TABLE1” and a column with name “COLUMN1”, value “VARCHAR2” and size “20” is already been assigned to it. So, we change the name of our table to ‘STUDENT\_T’, changed its column name to ‘STUDENT\_NAME’ by simply double clicking on the column name and finally we added

another column, by the + sign made at the top right corner of our panel, and name it as 'STUDENT\_ID'. (We can also change the data type and the size of our column depending upon our need.)

Schema: SYSTEM

Name: STUDENT\_T

Advanced

Table DDL

Columns: name

PK	Name	Data Type	Size	Not Null	Default	Comment
	STUDENT_NAME	VARCHAR2	20	<input type="checkbox"/>		

Add Column (ALT-1)

Help OK Cancel

Create Table

Schema: SYSTEM

☐ Advanced

Name: STUDENT\_T

Table

DDL

Columns:

PK	Name	Data Type	Size	Not Null	Default	Comment
	STUDENT_NAME	VARCHAR2	20	<input type="checkbox"/>		
	STUDENT_ID	VARCHAR2	20	<input type="checkbox"/>		

↑

↑

↓

↓

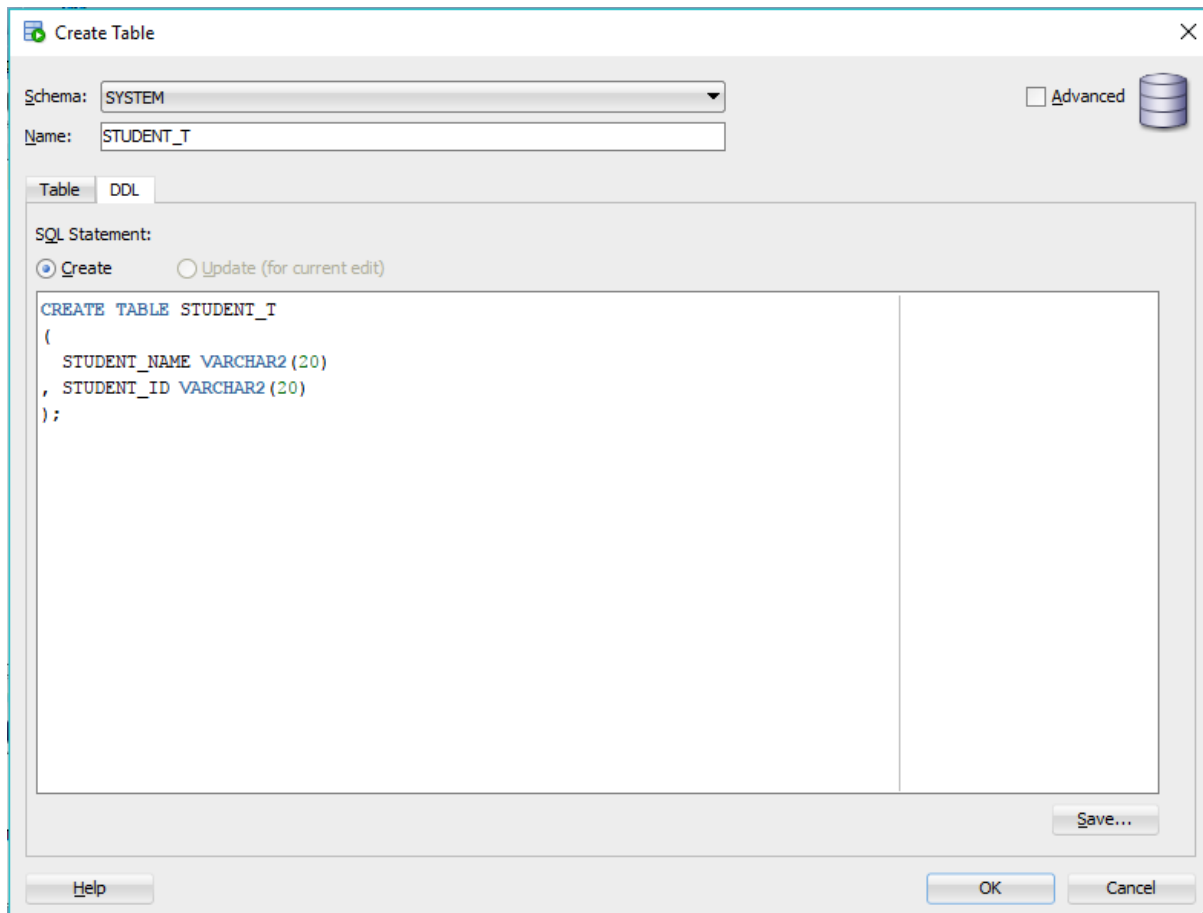
Help

OK

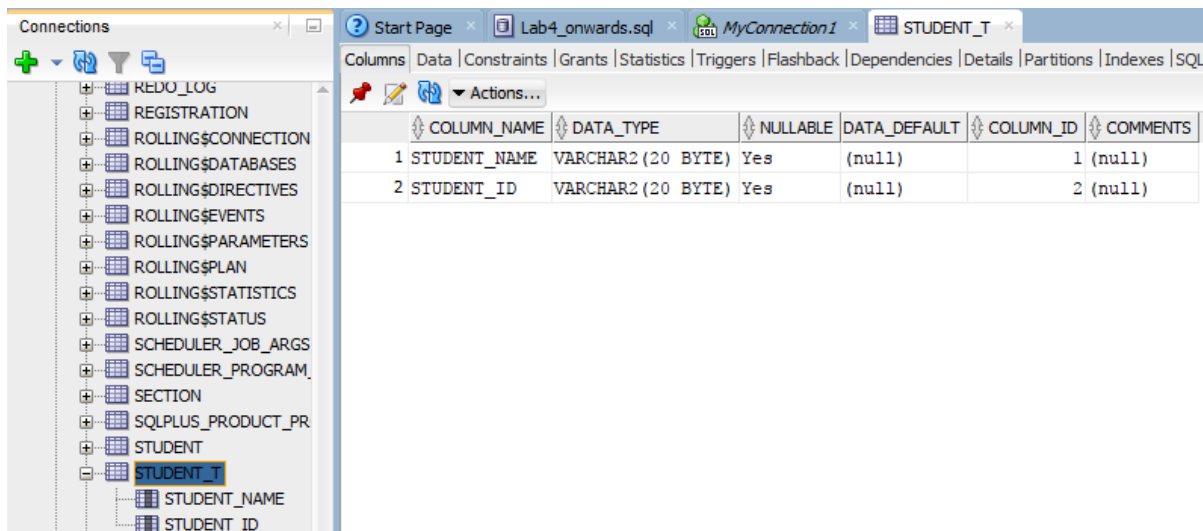
Cancel

We can also see the DDL created and working at the back end when we are creating our table. Now, we are in the “Table” tab for checking the DDL we can navigate to the “DDL” tab, and see the SQL query for our table created at the back end.



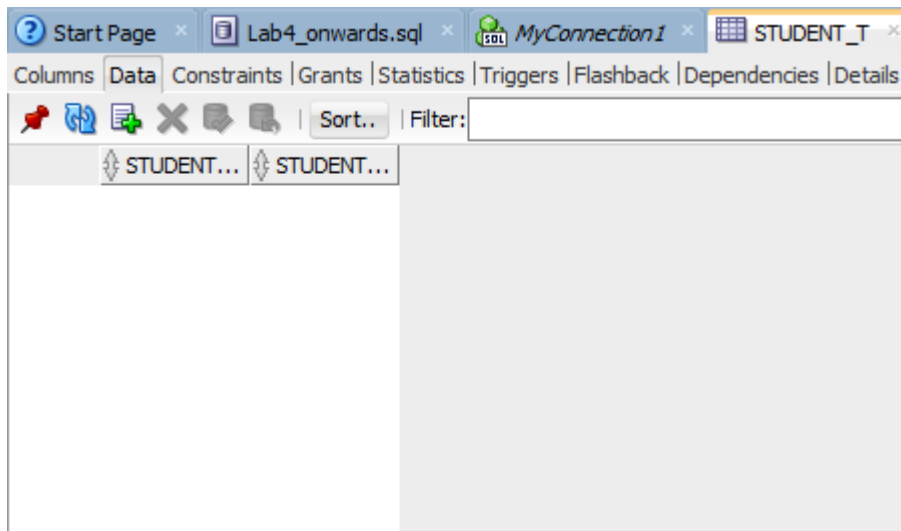



And now when we press “OK”, the table is created in our database. We can see and access this table easily by simply navigating our “Tables” option in the left navigation bar.




Here we can see that a table has been successfully created with the details we have entered while creating the table via GUI.

Now, if we navigate to the “Data” tab in the worksheet area, we can see that there is no data in our table i.e. our table is empty.



For entering a row of data in the table, press  and enter data in the table.

	STUDENT_NAME	STUDENT_ID
+1	Hania Arif	15-SE-03

After entering the data in the table, we have to store the data which we can do simply by clicking . This is for committing changes i.e. saving changes in our table.

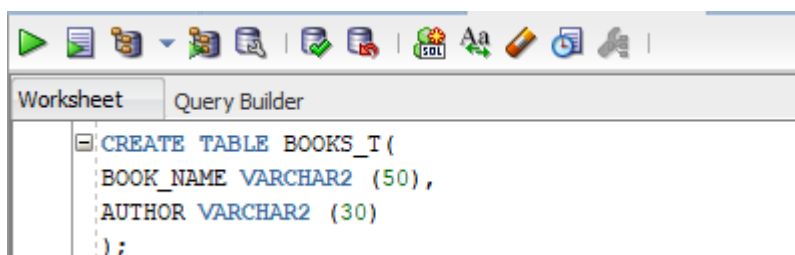
## Creating table by SQL query


The general syntax for creating a table by SQL query is:

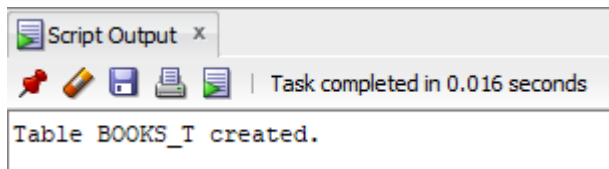
```
CREATE TABLE table_name (
column_name1 data_type,
column_name2 data_type,
column_name3 data_type, ...);
```

For creating a table by SQL query, we have to write the SQL code in the workspace of our connection. I had created a table named 'BOOKS\_T' with two columns of 'BOOK\_NAME' and 'AUTHOR' with the size and value of 'VARCHAR2 (50)' and 'VARCHAR2(30)' respectively.

I have created the BOOKS\_T table on the same syntax, in the worksheet area.

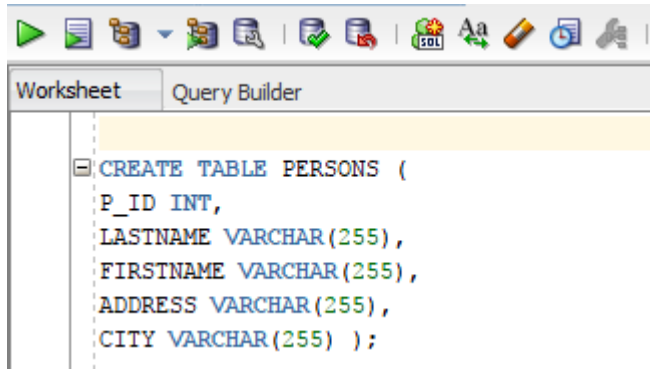


Now, if I run this query by the  button, the table will be created in the database and the script output would be shown as follows:

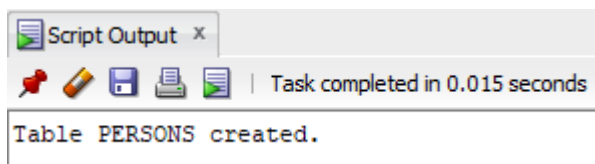


## CREATE TABLE example

In this example we created a table named as 'PERSONS' in our database with the following query:



And when we run this query the script output shows that the PERSONS table has successfully been created.



## Oracle database software tools

Oracle has the many software tools. Some of them are:

- SQL \* PLUS
- Oracle Forms
- Report Writer

### SQL \* PLUS

This tool is made up of two parts:

- Interactive SQL
- PL/SQL

#### Interactive SQL

Interactive SQL is designed to create, access and manipulate data structures like tables and indexes in the database.

#### PL/SQL

It is used to develop programs for different applications.

## Oracle Forms

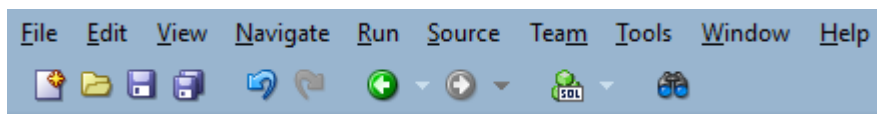
This tool allows us to create a data entry screen along with suitable menu objects. Thus, it is used to handle data gathering and data validation in commercial applications.

## Report Writer

It handles the reporting section of the commercial applications. It allows the programmers to prepare innovative reports by using the data from the oracle structures like tables and views, etc.

## SQL Developer User Interface

The Oracle 12c – Release 2 has the following user interface:



We learnt the use short keys to access menus and menu items. Following are the results of using some of the sort keys in Oracle.








Short Key	Menu to be accessed	Results when short key used
Alt+F	File	A screenshot of the 'File' menu in Oracle SQL Developer. The menu is open, showing various options. The 'File' menu bar is highlighted at the top. The options listed are: New... (Ctrl-N), Open... (Ctrl-O), Reopen, Close (Ctrl-F4), Close All (Ctrl+Shift-F4), Save (Ctrl-S), Save As..., Save All, Rename..., Compare With, Replace With, Page Setup..., Print... (Ctrl-P), Print Preview..., Print Area, and Exit (Alt-F4). The 'New...' option is currently selected and highlighted in blue.

Alt+E	Edit	
Alt+H	Help	
Alt+S	Source	

We can also use F10 to display the File menu but in the SQL worksheet F10 is for 'Explain Plan'. Alt+F4 is used to close any window and any of its dependent windows.

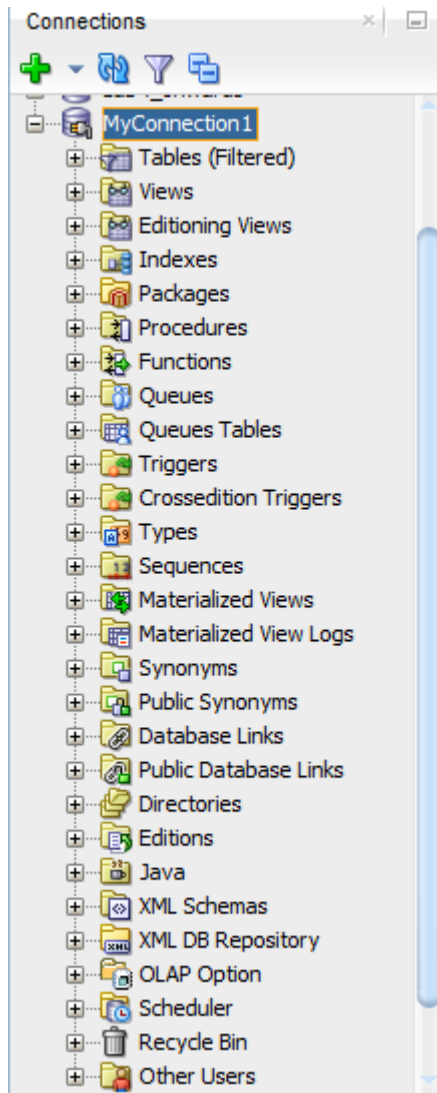
After the use of short keys, we also learnt the propose of the different buttons in the main menu bar. Here are the uses of those buttons:

Button Name and Icon	Purpose
----------------------	---------

 New (Ctrl-N)	Create a new database object
 Open... (Ctrl-O)	Open a file
 Save (Ctrl-S)	Save changes to currently selected object
 Save All	Save changes to all open objects
 Back to lab_report.sql (Alt-Left)	Move back to the pane that you have recently visited. (Here it is showing 'Back to lab_report.sql' as I had last visited lab_report.sql)
 Forward (Alt-Right)	Move to the pane after the current one in the list of visited pane.
 SQL Worksheet (Alt-F10)	It opens the SQL worksheet. If the user does not uses the dropdown arrow, it asks the user to establish connection with one of his connections so that it might open the worksheet of that specific connection.

## Metadata

There is a metadata tree in the connections pane. The metadata tree of 'MyConnection1' - the connection that we created earlier- is as follows:



The metadata tree displays all the objects, accessible to all connections. We can select any sub-tree and expand the object that we want to use or see.

## Creating SCHEMA in Oracle

### Lab Task

Create all the tables given below with columns specified: (Coulmn\_ID should be Int type, Name is VarChar)

STUDENT (StudentID, StudentName)

<u>StudentID</u>	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

QUALIFIED (FacultyID, CourseID, DateQualified)

<u>FacultyID</u>	<u>CourseID</u>	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
...		

FACULTY (FacultyID, FacultyName)

<u>FacultyID</u>	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

SECTION (SectionNo, Semester, CourseID)

<u>SectionNo</u>	<u>Semester</u>	<u>CourseID</u>
2712	I-2008	ISM 3113
2713	I-2008	ISM 3113
2714	I-2008	ISM 4212
2715	I-2008	ISM 4930
...		

COURSE (CourseID, CourseName)

<u>CourseID</u>	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
...	

REGISTRATION (StudentID, SectionNo, Semester)

<u>StudentID</u>	<u>SectionNo</u>	<u>Semester</u>
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008
...		

## Creating STUDENT table

```
CREATE TABLE STUDENT (
    STUDENTID INT NOT NULL PRIMARY KEY,
    STUDENTNAME VARCHAR(50)
);
```

## Inserting data in STUDENT table

```
INSERT INTO STUDENT (STUDENTID, STUDENTNAME) VALUES (38214, 'Letersky');
INSERT INTO STUDENT (STUDENTID, STUDENTNAME) VALUES (54907, 'Altveter');
INSERT INTO STUDENT (STUDENTID, STUDENTNAME) VALUES (66324, 'Aiken');
INSERT INTO STUDENT (STUDENTID, STUDENTNAME) VALUES (70542, 'Marra');
```

## Script Output



Table STUDENT created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

## Creating FACULTY table

```
CREATE TABLE FACULTY(  
    FACULTYID INT NOT NULL PRIMARY KEY,  
    FACULTYNAME VARCHAR (10)  
);
```

## Inserting data in FACULTY table

```
INSERT INTO FACULTY (FACULTYID, FACULTYNAME) VALUES (2143, 'Birkin');  
INSERT INTO FACULTY (FACULTYID, FACULTYNAME) VALUES (4756, 'Berndt');  
INSERT INTO FACULTY (FACULTYID, FACULTYNAME) VALUES (3467, 'Collins');
```

## Script Output

Table FACULTY created.

1 row inserted.

1 row inserted.

1 row inserted.

## Creating COURSE table

```
DROP TABLE COURSE;  
CREATE TABLE COURSE(  
    COURSEID VARCHAR(10) NOT NULL PRIMARY KEY,  
    COURSENAME VARCHAR (50)  
);
```

## Inserting data in COURSE table

```
INSERT INTO COURSE (COURSEID, COURSENAME) VALUES ('ISM 3113', 'Syst Analysis');  
INSERT INTO COURSE (COURSEID, COURSENAME) VALUES ('ISM 3114', 'Syst Design');  
INSERT INTO COURSE (COURSEID, COURSENAME) VALUES ('ISM 4930', 'Database');  
INSERT INTO COURSE (COURSEID, COURSENAME) VALUES ('ISM 4212', 'Networking');
```

## Script Output

Table COURSE created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

## Creating QUALIFIED table

```
CREATE TABLE QUALIFIED(  
    FACULTYID INT,  
    COURSEID VARCHAR (10),  
    DATEQUALIFIED VARCHAR(10),  
    CONSTRAINT FK1 FOREIGN KEY (FACULTYID) REFERENCES FACULTY(FACULTYID),  
    CONSTRAINT FK2 FOREIGN KEY (COURSEID) REFERENCES COURSE(COURSEID)  
);
```

## Inserting data in QUALIFIED table

```
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (2143,'ISM 3113','9/1988');  
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (2143,'ISM 3113','9/1988');  
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (3467,'ISM 4212','9/1995');  
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (3467,'ISM 4930','9/1996');  
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (4756,'ISM 3113','9/1991');  
INSERT INTO QUALIFIED (FACULTYID, COURSEID, DATEQUALIFIED) VALUES (4756,'ISM 3113','9/1991');
```

## Script Output

Table QUALIFIED created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

## Creating SECTION table

```
CREATE TABLE SECTION(  
    SECTIONNO INT NOT NULL PRIMARY KEY,  
    SEMESTER VARCHAR (10),  
    COURSEID VARCHAR(10),  
    CONSTRAINT FK_SEC FOREIGN KEY (COURSEID) REFERENCES COURSE(COURSEID)  
);
```

## Inserting data in SECTION table

```
INSERT INTO SECTION (SECTIONNO, SEMESTER, COURSEID) VALUES (2143, 'I-2008', 'ISM 3113');
INSERT INTO SECTION (SECTIONNO, SEMESTER, COURSEID) VALUES (2144, 'I-2008', 'ISM 3113');
INSERT INTO SECTION (SECTIONNO, SEMESTER, COURSEID) VALUES (3467, 'I-2008', 'ISM 4212');
INSERT INTO SECTION (SECTIONNO, SEMESTER, COURSEID) VALUES (3468, 'I-2008', 'ISM 4930');
```

## Script Output

Table SECTION created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

## Creating REGISTRATION table

```
CREATE TABLE REGISTRATION(
  STUDENTID INT,
  SECTIONNO INT,
  SEMESTER VARCHAR(10),
  CONSTRAINT FK_REG1 FOREIGN KEY (STUDENTID) REFERENCES STUDENT(STUDENTID),
  CONSTRAINT FK_REG2 FOREIGN KEY (SECTIONNO) REFERENCES SECTION(SECTIONNO)
);
```

## Inserting data in REGISTRATION table

```
INSERT INTO REGISTRATION (STUDENTID, SECTIONNO, SEMESTER) VALUES (38214, 2143, 'I-2008');
INSERT INTO REGISTRATION (STUDENTID, SECTIONNO, SEMESTER) VALUES (70542, 2144, 'I-2008');
INSERT INTO REGISTRATION (STUDENTID, SECTIONNO, SEMESTER) VALUES (66324, 3467, 'I-2008');
INSERT INTO REGISTRATION (STUDENTID, SECTIONNO, SEMESTER) VALUES (54907, 3468, 'I-2008');
```

## Script Output

Table REGISTRATION created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Here foreign key and primary key are also included in the SQL query as we have learnt them in the next lab. For now, if we don't want to insert the primary and foreign keys, we can also run the rest of the query without the primary and foreign keys, the code will run in the same way and simple tables without primary or foreign keys would be created. We can also insert data in the tables by using the GUI at this level, as done in the earlier examples. As, we had learnt the row insertion method in the next lab so that this method is used here.

# Introduction to SQL and DDL, Creating Tables and Implementing Collation in Oracle

---

## Aims and Objectives

In this lab the students learnt the following things:

- What is SQL and how does it work?
- What are the datatypes in SQL?
- What is DDL?
- Different SQL constraints
- What is collation?

## SQL Datatypes

SQL language has the many different datatypes which we can select for any column as per our requirement. Following are the datatypes used in SQL:

- CHAR
- VARCHAR / VARCHAR2
- NUMBER
- DATE
- LONG
- RAW

### CHAR (size)

It is used to store character string values and it can store up to 255 characters.

### VARCHAR(size) / VARCHAR2(size)

It is used to store alpha-numeric data of variable length. It can store up to 2000 characters.

### NUMBER (P, S)

It is used to store fixed or floating-point numbers. The precision (P) determines the number of places to the right of the decimal. It can store up to 38 digits of precision.

### DATE

It is used to represent and store date and time.

### LONG

It is used to store variable length string characteristics. It can store up to 2GB characters.

### RAW

It is used to store binary data such as image or picture. It can store data of maximum length of 255 bytes and it can store up to 2GB.

## Types of SQL statements

There are 5 types of SQL statements. They are:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Retrieval Language (DRL)
4. Transactional Control Language (TCL)
5. Data Control Language (DCL)

## Data Definition Language (DDL)

This language is used to create and destroy databases and database objects. Generally, 4 DDL commands used are:

- CREATE
- ALTER
- DROP
- RENAME

### CREATE

#### CREATE TABLE

##### *General syntax*

CREATE TABLE table\_name (

Field1 datatype(size),

Field2 datatype(size),

...

);

##### *Example*

##### **SQL query**

```
CREATE TABLE STUDENTS_T (  
    SNO NUMBER(3),  
    SNAME CHAR(10),  
    CLASSNAME CHAR(5)  
);
```

##### **Script output**

Table STUDENTS\_T created.

#### CREATE TABLE ... AS SELECT ...

##### *General syntax*

CREATE TABLE new\_table\_name(

field1, field2, ...)

AS (

SELECT (field1, field2, ...)

FROM old\_table\_name;

### *Example*

#### **SQL query**

```
CREATE TABLE stud_t (rno,sname)
AS SELECT sno,sname FROM STUDENTS_T;
```

#### **Script output**

```
Table STUD_T created.
```

## **Task**

Create the table “Students” with columns:

- ✓ **RegNo** (field type: varchar) & make it primary key
- ✓ **First\_Name** (field type: varchar)
- ✓ **Last\_Name** (field type: varchar)
- ✓ **Father\_Name** (field type: varchar)
- ✓ **Address** (field type: varchar)
- ✓ **CGPA** (field type: float)

#### **SQL query**

```
CREATE TABLE STUDENTS (
  REGNO VARCHAR(20) PRIMARY KEY,
  FIRST_NAME VARCHAR(20),
  LAST_NAME VARCHAR(20),
  FATHER_NAME VARCHAR(20),
  ADDRESS VARCHAR(20),
  CGPA FLOAT
);
```

#### **Script output**

```
Table STUDENTS created.
```

## **SQL Constraints**

Following are the generally used constraints in SQL:

- NOT NULL
- UNIQUE
- ALTER TABLE
- PRIMARY KEY
- FOREIGN KEY

- CHECK
- DEFAULT

## NOT NULL

It enforces a column, not to accept any NULL values.

### Example

#### SQL query

```
CREATE TABLE Persons (
  P_Id int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
);
```

#### Script output

Table PERSONS created.

## UNIQUE

By this keyword we can uniquely identify each record in the database table. A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it. We can have many UNIQUE constraints in a table but there could be only one PRIMARY KEY in a table at a time.

### UNIQUE Constraint on CREATE TABLE

#### Example

Here we have made a single column as UNIQUE

#### SQL query

```
CREATE TABLE Persons (
  P_Id int NOT NULL UNIQUE,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255)
);
```

#### Script output

Table PERSONS created.

#### Example

Here we have made multiple columns as UNIQUE, at a time

#### SQL query

```
CREATE TABLE Persons (  
    P_Id int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255),  
    CONSTRAINT uc_PersonID UNIQUE (P_Id, LastName)  
);
```

### Script output

Table PERSONS created.

### UNIQUE Constraint on ALTER TABLE

ALTER TABLE is used if we want to make some changes in our tables which have been already present in the database.

#### *Example*

Here we suppose that the PERSONS table has already been created in the database without any UNIQUE constraint as:

```
CREATE TABLE Persons (  
    P_Id int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

Now we want to make the P\_ID column as UNIQUE so that we will use the following SQL query:

### SQL query

```
ALTER TABLE PERSONS  
ADD UNIQUE (P_ID);
```

*(The ADD command is used to add something in the table)*

### Script output

Table PERSONS altered.

#### *Example*

Here suppose that the PERSONS table has already been created in the database without any UNIQUE constraint as:



```
CREATE TABLE Persons (  
    P_Id int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

Now we want to make the P\_ID and last name column as UNIQUE, at the same time, so that we will use the following SQL query:

### SQL query

```
ALTER TABLE PERSONS  
ADD CONSTRAINT uc_PersonID UNIQUE (P_ID, LastName);
```

*(The ADD command is used to add something in the table, the CONSTRAINT command after ADD tells that we are going to add some constraint to the table)*

### Script output

Table PERSONS altered.

### DROP a UNIQUE Constraint

DROP is used if we want to delete something from our table.

### Example

Here we suppose that the PERSONS table has already been created in the database with the P\_ID as UNIQUE as:

```
CREATE TABLE Persons (  
    P_Id int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255),  
    CONSTRAINT uc_PersonID UNIQUE (P_Id, LastName)  
);
```

Here we are deleting the UNIQUE constraint from the PERSON table

### SQL query

```
ALTER TABLE PERSONS  
DROP CONSTRAINT uc_PersonID;
```

### Script output

Table PERSONS altered.