

One of Many Books in the Need to Know Series from BrainMass

# EVERYTHING YOU **NEED** TO KNOW ABOUT

# SQL Database OBJECTS

Ideal for  
Beginner  
SQL  
Students

By Narasimha Rao



[www.brainmass.com](http://www.brainmass.com)

# **Everything You Need to Know About SQL Database Objects**

By Narasimha Rao

© BrainMass Inc. 2011

# What Is This Book?

---

Everything You Need to Know publications are the best way to get a quick but detailed overview of a specific topic. Within the pages of this book, you'll find exactly what you need in order to understand the key concepts of this topic. You'll find yourself completely prepared for the next stage of your learning as it relates to this topic. Within every book, we include a collection of the most important terms and their definitions so that whenever you're in need of a refresher, you can easily refer back and remind yourself of what you're dealing with.

This book covers all basic **Structured Query Language (SQL)** statements including **Data Definition Language, Data Manipulation Language, and Transaction Control statements**. This book deals mainly with one of the database objects: Tables. It gives a brief introduction to each category of **SQL** statements, followed by a set of solved sample questions. A number of practice questions are provided at the end of each topic, with their solutions included at the end of the book. The topics covered in this book include understanding naming rules for table and column names, explanation of different operators and data types apart from explanation of different categories of **SQL** statements with examples.

This book is suitable for students who are in their first year of undergraduate or graduate studies and are learning **SQL** for the first time. The software used in this book is Oracle Enterprise Edition.

# Table of Contents

---

<b>Introduction.....</b>	<b>5</b>
<b>Everything You Need to Know .....</b>	<b>6</b>
1. Database Objects .....	6
2. Naming Rules for Table and Column Names .....	6
3. Data Types.....	7
4. Structured Query Language .....	8
5. Data Definition Language Statements (DDL).....	8
6. Data Manipulation Language Statements (DML).....	12
7. Transaction Control Language Statements (TCL) .....	20
8. Operators .....	23
9. Common Pitfalls.....	26
10. Solved Examples .....	28
11. Practice Questions .....	35
12. Solutions to Practice Questions.....	36
<b>Glossary .....</b>	<b>37</b>
<b>References.....</b>	<b>38</b>
<b>About the Author .....</b>	<b>39</b>

# Introduction

---

After reading this book, you should be able to understand different database objects, understand naming rules for tables and column names, state the different data types, understand the significance of **Structured Query Language (SQL)**, understand the various types of **SQL** statements, describe and work with different **Data Definition Language (DDL)** statements, describe and work with different **Data Manipulation Language (DML)** statements, describe and work with different **Transaction Control** statements and explain how constraints are created at the time of table creation.

Though all the queries mentioned in this book have been tested using the Oracle 11g; these would also work with previous versions of Oracle as most of the queries are basic in nature. Most of the queries should work as such with other database packages such as SQL Server and MySQL; minor modifications may be needed in some queries. Note that **Structured Query Language (SQL)** is a standard that is implemented by all database packages, so the syntax of all the SQL queries is same in all database packages, with the manufacturers of these packages adding some additional features and options to basic SQL. The free Enterprise Edition of Oracle software is available at the following URL:

<http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html>

# Everything You Need to Know

---

## 1. Database Objects

---

There are several database objects including:

- Table
- View
- Sequence
- Synonym

Since this book is introductory in nature, we shall deal only with tables in this book.

## 2. Naming Rules for Table and Column Names

---

These rules are valid for most of the database packages implementing SQL:

- a) All names must begin with an alphabet.
- b) The length of the names varies between 1 and 30 characters.
- c) The names can contain only the following characters: A-Z,a-z,0-9,\_,,\$ and #.
- d) Two different database objects cannot share the same name.
- e) The name should not be a registered keyword.
- f) Names are not case sensitive, so there is no difference between lower and upper case.



## 3. Data Types

---

There are several different data types used in SQL. The four most important and common data types are:

- a) Char
- b) Varchar2
- c) Number
- d) Date

There are several other data types, but are not necessary for learning basic SQL commands.

“Char” data type is used for representing fixed length character data. This data type is normally used when a character field needs a fixed number of columns, such as gender which is represented as ‘m’ for male and ‘f’ for female. In such a case, the “char” data type can be used.

“Varchar2” or “Varchar” is used for representing variable length character data. For example, in the case of name of a student, the name field could contain a name such as ‘John’ which requires only 4 characters, or a 20-character long name. In such a case, it would result in wastage of memory/disk (rather persistent) space if we used a fixed length data type to reserve space. In such cases, using the variable length character data types is a better option.

The “Number” data types used to describe both integers and floating point numbers. For example if you wish to define a field called age as a 3-digit number, then you can indicate its data type as number (3). Similarly if you want to define a field called salary as a floating point number with 4-digits in integer portion and 2 digits in decimal precision, then you can indicate its data type as number (6, 2).

The “Date” data type is used to define a field that stores dates. When we input date values into a field, we should use the format: “DD-MMM-YYYY”, for example “21-JAN-1997” is in this format. However, this format can be changed by using the date functions.

## 4. Structured Query Language

---

**Structured Query Language** (SQL) is the standard language for relational databases accepted by both ANSI (American National Standards Institute) and ISO (International Standards Organization).

## 5. Data Definition Language Statements (DDL)

---

**Data Definition Language** statements set up, change and remove structures from tables. The five key **data definition language** statements are:

1. Create Table

2. Alter Table

3. Drop Table

4. Rename

5. Truncate Table



## 5.1 Create Table

---

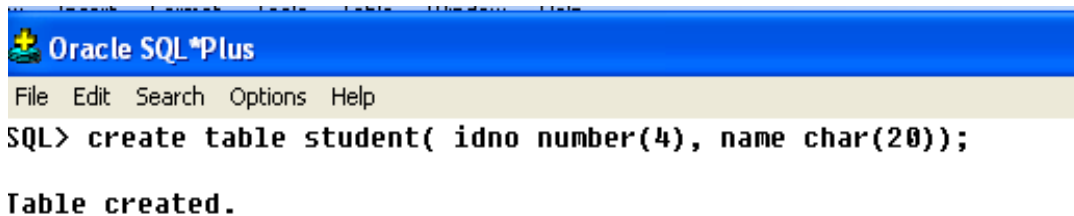
This command is used to create a table.

Syntax:

Create table <table name> (<column name> <datatype>(<width>),...);

Example:

SQL> create table student( idno number(4), name char(20));

A screenshot of the Oracle SQL\*Plus command window. The title bar is blue with the Oracle SQL\*Plus logo and text. Below the title bar is a menu bar with 'File', 'Edit', 'Search', 'Options', and 'Help'. The main window area is white and contains the text: 'SQL> create table student( idno number(4), name char(20));' followed by a blank line and then 'Table created.'

Note:

Desc or Describe is a **SQL\*** Plus command that is used to display the structure of a table. In SQL Server, you will use sp\_help instead of desc to display the structure of a table.

## 5.2 Alter Table

---

This statement is used to alter the structure of an existing table. There are two options:

- a) Add
- b) Modify

Syntax:

Alter table <table-name> <add/modify>(<column-name> <data type>(<width>),....);

Example:

SQL> alter table student add(gender char(2));



```
SQL> alter table student add(gender char(2));
```

**Table altered.**

The add option is used along with alter table statement to add a new column to the table. In the above example we are adding a new column called as gender to an existing table.

The modify option is used to modify an existing column of an already existing table. For example if we want to modify the width of "name" field in the above table from 20 characters to 25 characters, then we have to issue the following command:

```
SQL> alter table student modify (name char(25));
```



```
SQL> alter table student modify (name char(25));
```

**Table altered.**

```
SQL> desc student
```

Name	Null?	Type
IDNO		NUMBER(4)
NAME		CHAR(25)
GENDER		CHAR(2)

This would modify the "name" field width from 20 characters to 25 characters.

Using modify clause we can:

- Increase or decrease the width of an existing column.
- Change the data type of an existing column.

However, there are some restrictions while changing the data type and decreasing the width of a column. However there are no restrictions while increasing the width of a data type. These restrictions will be dealt in detail at a later stage.

## 5.3 Drop Table

---

Drop Table statement is used to delete the columns of a table. This command physically deletes the table from the disk and must be used with caution. **Once a table is dropped, it cannot be retrieved back.**

Syntax:

Drop table <table name>

Example:

SQL> drop table student;

## 5.4 Rename

---

The rename statement is used to rename a table. Once the rename statement is issued, the table can be accessed only using the new name.

Syntax:

Rename <old-table-name> to <new-table-name>;

Example:

SQL> rename student to stud;

## 5.5 Truncate Table

---

Truncate table is used to delete all the records of a table, but the structure of the table remains intact.

Syntax:

Truncate table <table-name>

Example:

SQL>Truncate table student;

## 6. Data Manipulation Language Statements (DML)

---

**Data Manipulation Language** statements retrieve data from the database, enter new rows, change existing rows, and remove unwanted rows from tables in the database. The four **data manipulation language** statements are:

1. Insert

2. Update

3. Delete

4. Select

## 6.1 Insert

---

The Insert statement is used to insert a single record into the database. It can also be used to insert multiple records at a time.

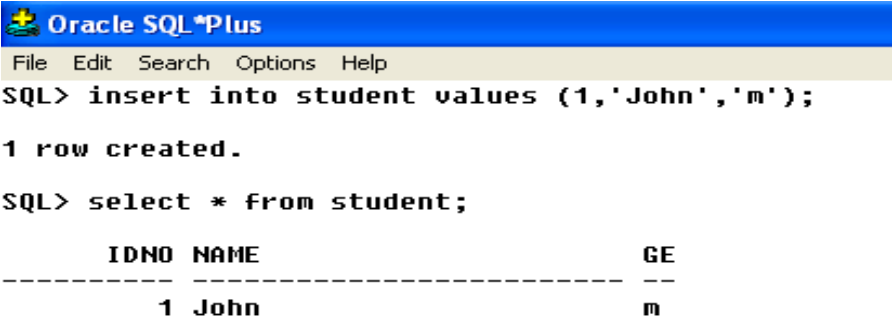
Syntax:

Insert into <table-name> values (value1, value2,...);

Note: *While inserting character and date values, they must be enclosed in single quotes.* The numeric values do not have to be inserted in single quotes.

Example:

SQL> insert into student values (1,'John','m');



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> insert into student values (1,'John','m');

1 row created.

SQL> select * from student;

  IDNO NAME          GE
-----
    1 John           m
```

Insert command with substitution variable (&)

*The substitution variable is available only in Oracle.*

In case we have to insert 10 records, inserting them with 10 insert commands would be cumbersome. Instead, we can issue the following command:

Example:

SQL> insert into student values (&idno, '&name', '&gender');

Now it would prompt you to enter the values. After you have entered the values, you would get a confirmation message saying 1 record created.

Now issue the following command to issue the next record:

SQL> /

The forward slash executes the last executed command, which in this case is the insert command. So you can enter new values.

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> insert into student values (&idno, '&name', '&gender');
Enter value for idno: 2
Enter value for name: Jack
Enter value for gender: m
old 1: insert into student values (&idno, '&name', '&gender')
new 1: insert into student values (2, 'Jack', 'm')

1 row created.

SQL> /
Enter value for idno: 3
Enter value for name: Mary
Enter value for gender: f
old 1: insert into student values (&idno, '&name', '&gender')
new 1: insert into student values (3, 'Mary', 'f')

1 row created.

SQL> select * from student;

      IDNO NAME                GE
-----
      1 John
      2 Jack
      3 Maru

```

Multiple records can also be inserted to the database using the insert all statement. An example of this is given below. If you want to insert 3 records at the same time into the student table, then issue the following insert all statement:

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> insert all into student values(4,'Alex','m')
2 into student values (5,'Maria','f')
3 into student values (6,'Smith','m')
4 select * from dual;

3 rows created.

SQL> select * from student;

      IDNO NAME                GE
-----
      1 John
      2 Jack
      3 Mary
      4 Alex
      5 Maria
      6 Smith

```

*Please note that the select \* from dual statement is needed at the end of every insert all statement.*

## 6.2 Update

---

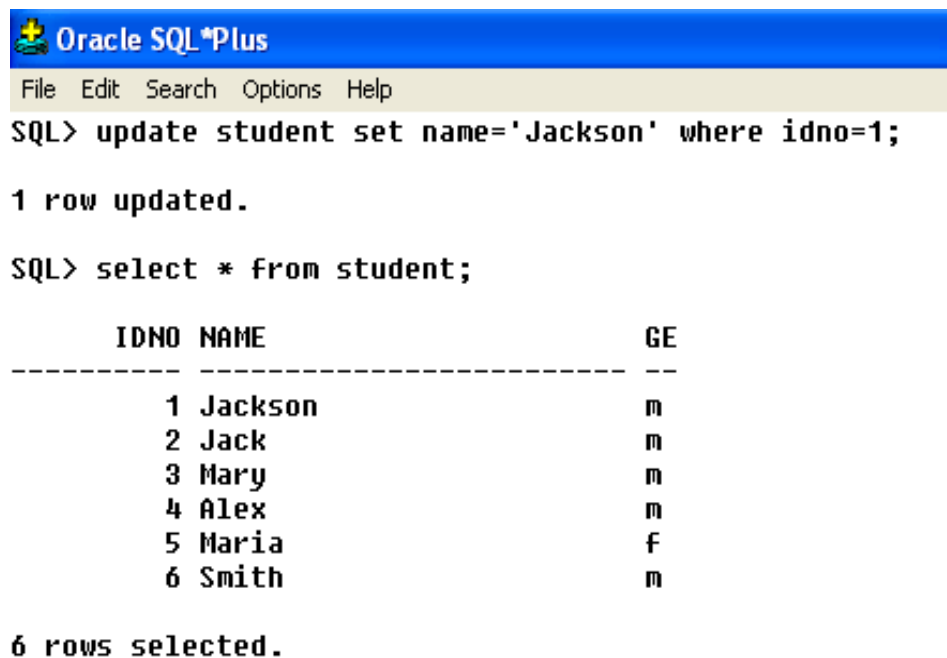
The update statement is used to update the values of a particular record or a group of records.

Syntax:

Update <table-name> set < column-name>=<value> where <condition>;

Example:

SQL>update student set name='Jackson' where idno=1;



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> update student set name='Jackson' where idno=1;

1 row updated.

SQL> select * from student;

  IDNO NAME                GE
-----
    1 Jackson              m
    2 Jack                  m
    3 Mary                  m
    4 Alex                   m
    5 Maria                  f
    6 Smith                  m

6 rows selected.
```

*Note: If the update command is given without "where" clause, then all the records in the table are updated.*

## 6.3 Delete

---

The delete statement is used to delete a single record or a group of records from the table.

Syntax:

Delete from <table-name> where<condition>

Example:

SQL>delete from student where idno=1;

*Note: If the delete command is given without "where" clause, then all the records in the table are deleted.*

## 6.4 Select

---

Select statement can be used for selection (selecting set of rows), projection (selecting set of columns) as well as joins (data from multiple tables).

In its basic form, a SELECT statement must include the following:

- a) A SELECT clause, which specifies the columns to be displayed
- b) A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

Syntax:

SELECT \*|{[DISTINCT] column|expression [alias],...} FROM table;



In the syntax:

SELECT	is a list of one or more columns
*	selects all columns
DISTINCT	suppresses duplicates
column expression	selects the named column or the expression
alias	gives the selected columns different headings
FROM table	specifies the table containing the columns

Basic Select statement can be used in different situations for

- a) Selecting all columns
- b) Selecting specific columns
- c) Selecting records meeting specific criteria using the WHERE clause
- d) Ordering the selected records using the "ORDER BY" clause (Examples are provided in the solved examples section of this book.)

*We can display all columns of data in a table by using the select statement with an asterisk (\*) sign.*

For example, the second SQL statement below would display all the columns of the student table.

Example:

```
SQL> select idno, name from student;
```

```
SQL> select * from student;
```

## 6.5 Removing Duplicates Using Select Statement

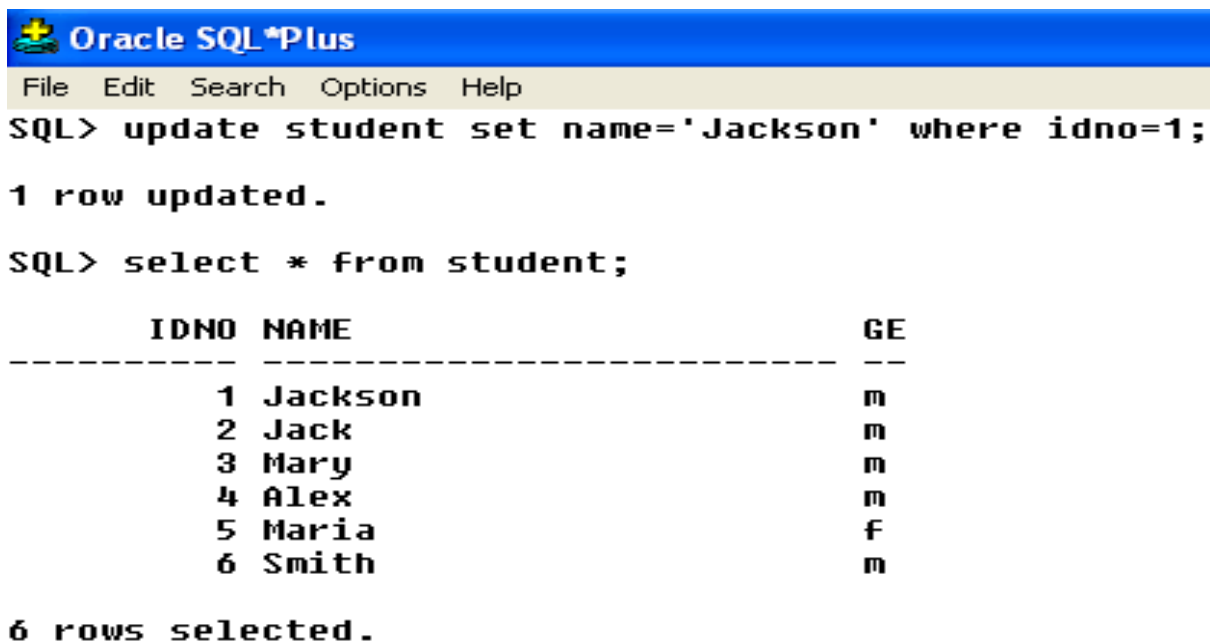
---

*Unless you indicate otherwise, SQL displays the results of a query without eliminating the duplicate rows.* To eliminate duplicate rows in the result, include the **DISTINCT** keyword in the **SELECT** clause.

Example:

Write a query to display the different types of gender used in the student table.

In the student table, there are two categories of gender 'm' and 'f'. A simple select query would return gender for each record, since there are six records, a total of six values would be returned with a number of duplicate values. The use of "DISTINCT" clause removes these duplicates as shown in the screenshot.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> update student set name='Jackson' where idno=1;

1 row updated.

SQL> select * from student;

  IDNO NAME                GE
-----
    1 Jackson              m
    2 Jack                  m
    3 Mary                  m
    4 Alex                   m
    5 Maria                  f
    6 Smith                  m

6 rows selected.
```

## 6.6 Defining a Column Alias

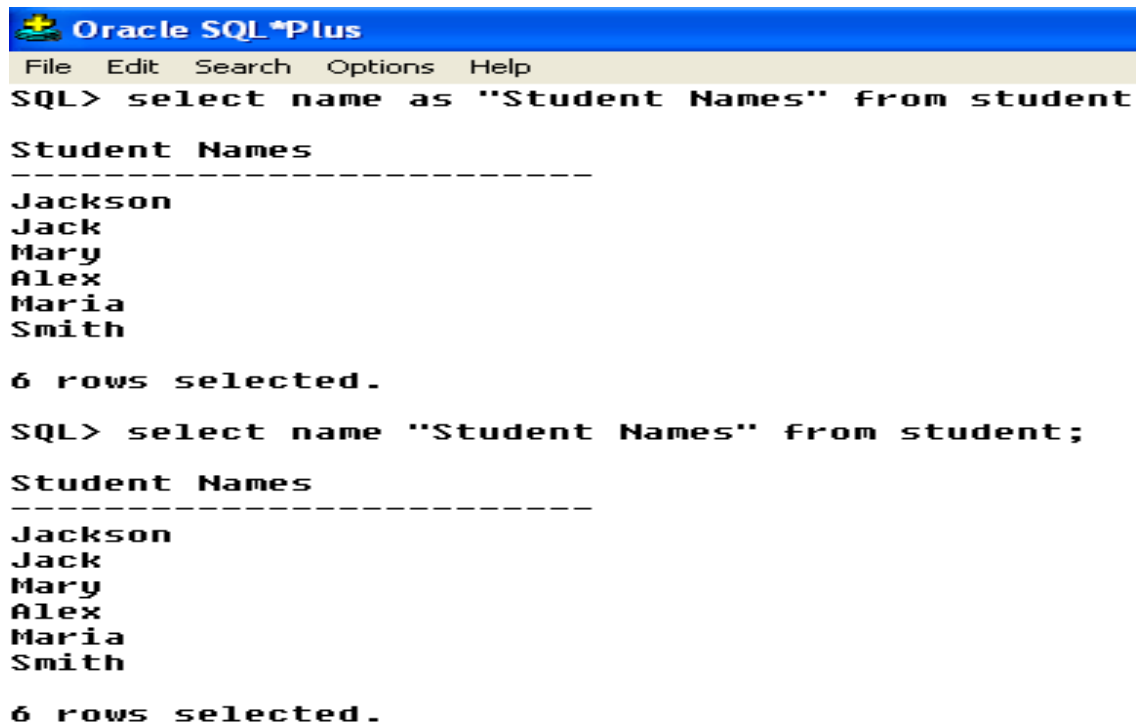
---

When displaying the result of a query, the name of the selected column is used as the column heading. This heading may not be descriptive and, therefore, may be difficult to understand. You can change a column heading by using a column alias.

Specify the alias after the column in the SELECT list using blank space or "AS" keyword as a separator. By default, alias headings appear in uppercase. If the alias contains spaces or special characters (such as # or \$), or if it is case-sensitive, enclose the alias in double quotation marks (" ").

Example:

Write a query to display all student names under the column heading Student Names.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select name as "Student Names" from student

Student Names
-----
Jackson
Jack
Mary
Alex
Maria
Smith

6 rows selected.

SQL> select name "Student Names" from student;

Student Names
-----
Jackson
Jack
Mary
Alex
Maria
Smith

6 rows selected.
```

In the above screen shot both the methods of using column alias are explored. In the first case, the "AS" keyword is used; while in the second case a "space" is used as a separator.

## 7. Transaction Control Language Statements (TCL)

---

**Transaction control language** statements manage the changes made by DML statements. Changes to the data can be grouped together into logical **transactions**.



1. Commit

2. Rollback

3. Savepoint

## 7.1 Transactions

---

Before learning about the **transaction** control statements, let us understand **transactions**. **Transaction** basically refers to set of statements that must be executed either completely or not executed at all. They follow the four key properties namely atomicity, consistency, isolation and durability. For example consider a **transaction** in a bank where a transfer is made from one account to another account. This **transaction** consists of two update statements, in which both the bank accounts are updated. Assume that one of the statements has executed successfully while the other has failed. This would cause inconsistency, hence a **transaction** would basically group these statements and either execute all the statements or not execute any of them. **Transactions** give more flexibility while at the same time ensure consistency in the event of system failure. A **transaction** begins when the first DML statement is encountered and ends in the following scenarios:

- a) **Transaction Control statements** such as commit or rollback is encountered.
- b) DDL statement is encountered.
- c) Data Control Statement is encountered.
- d) System failure or exit from the local environment.

## 7.2 Commit

---

The commit statement ends the current **transaction** by making all the pending changes permanent. In the case of unexpected system failure, the changes could be lost if either an implicit or an explicit commit statement is not issued.

Syntax:  
Commit

Example:  
SQL> Commit;

## 7.3 Rollback

---

The rollback statement ends the current **transaction** by removing all the pending data changes. For example if we have issued a DML statement by mistake and wish to correct the mistake (akin to the undo feature in word processing applications), we can always rollback the changes either to a particular savepoint or previously committed state.

Syntax:

Rollback [<to savepoint savepoint-name>]

Examples:

SQL> rollback;

or

SQL> rollback to savepoint abc;

In the above statement, abc is the name of the savepoint. Savepoints are explained next.

## 7.4 Savepoint

---

The savepoint statement is used to set a marker within the current **transaction**. This allows the user to go back to a particular stage within the **transaction** i.e. to roll back to a particular point in the **transaction**.

Syntax:

Savepoint <savepoint-name>

Example:

SQL> savepoint abc;

## 8. Operators

---

While you might be familiar with the basic arithmetic operators (addition+, multiplication \*, subtraction -, and division/), there are some other operators, you would need to be familiar with to use **SQL** more effectively.

### 8.1 Savepoint

---

You can link multiple columns using the concatenation operator (||). For example if we want to combine both the first\_name and last\_name fields in a table, we can give the following select statement using the concatenation operator:

```
SQL> select first_name || ' ' || last_name from employees;
```

The above select statement would display the full name of the employees, with a space separating the first name and the last name of the employee.

### 8.2 Comparison Operators

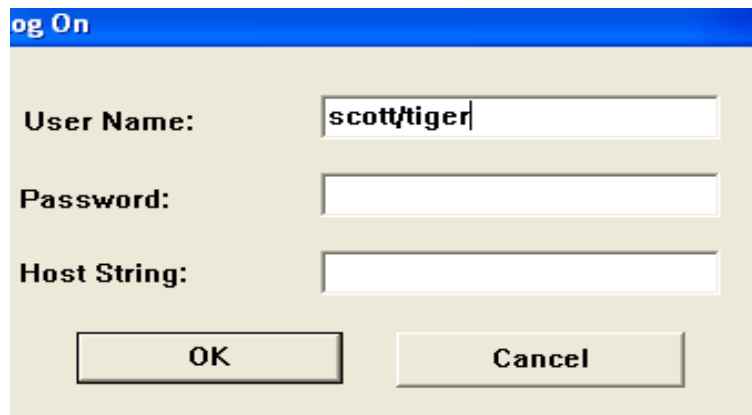
---

Given below is a list of comparison operators, with their respective meaning. You will understand these better when we use these operators in the exercises later in this book.

=	Equal To
<>	Not Equal To
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To
BETWEEN...AND	Between two values
LIKE	Match a character pattern
IN	Match any of a list of values
IS NULL	Is a null value

In the examples below the "emp" table is used because these operators can be better explained with the help of a table having more columns and rows as compared to "student" table which had a few columns and rows.

You can access the emp table by logging in as user scott with password tiger as shown in the screenshot below.



Example:

Write a query to display the names and salaries of employees whose salaries are in the range 2000 and 3000.

Query:

```
select ename,sal from emp where sal between 2000 and 3000;
```

In the above query, the between operator checks for all the salaries in a particular range, this range includes the values of both the upper as well as lower limit. In the example, the lower and upper limits are 2000 and 3000 respectively. So, the select query returns all employee names and salaries that are between these lower and upper limits including the values of lower and upper limits. The above query would be same as querying:  

```
select ename, sal from emp where sal>=2000 and sal<=3000;
```

Write a query to display names of all employees of all employees whose name begins with 'S' followed by any number of characters.

Query:

```
select ename from emp where ename like 'S%';
```

There "like" operator is always used in conjunction with the wild cards '%' and '\_' and is used to check for a particular pattern. In this case the pattern we are looking for is – "all employee names starting with 'S' followed by any number of characters. Since the '%' wildcard literally means any number of characters, the 'like' operator would be looking for matches starting with 'S' followed by any number of characters.



Write a query to display all the employee names that have exactly four characters.

Query:

```
select ename from emp where ename like '____';
```

The above query again uses the 'like' operator but in conjunction with the '\_' operator. Please note that the above pattern in four underscore characters written in sequence, and not underline as it looks in the above query. The underscore ('\_') character stands for any one character, since we are looking for a pattern of 4 characters, we can use the underscore character four times. The 'like' operator would look for this pattern and display all the names that satisfy this requirement.

Write a query to display names of employees who do not earn a commission.

Query:

```
select ename, sal from emp where comm is null;
```

In the 'emp' table, the column 'comm.' contains the commission earned by an employee. However, all employees need not receive commissions. In case an employee does not receive commission the corresponding entry in the 'comm' field is equal to null. Since null itself is not a defined value, we cannot use the '=' operator. So we use 'is' operator instead of '=' operator.



## 9. Common Pitfalls

---

The most common errors while executing SQL queries include:

a) Not placing character and date fields in single quotes. All character and date field values should be placed in single quotes. Numeric fields on the other hand do not have to be placed in single quotes.

b) Incorrect usage of Select Statement. Remember that the asterisk sign (\*) indicates all columns, and hence cannot be used in conjunction with other field names.

Example:

**SQL**> select ename,\* from emp;

The above query would result in a syntax error as we cannot use both \* and column names at the same time. In case you need only a few fields from the table, do not use the asterisk sign and instead list all the fields in the select query.

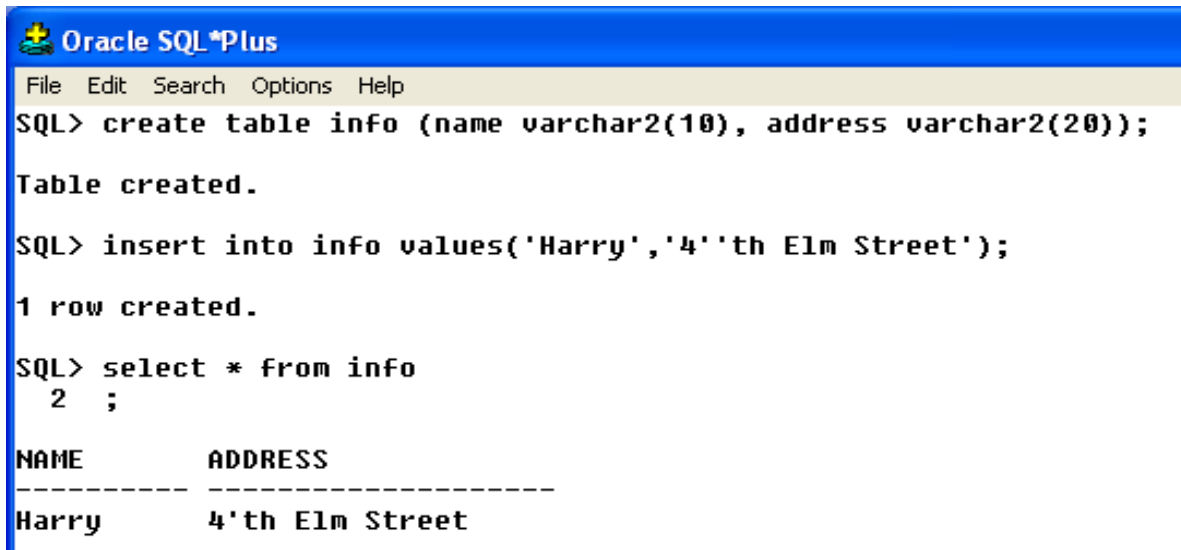
c) Forgetting statement terminator (;). All **SQL** statements in Oracle must terminate with a semicolon. The **SQL** statement is not executed if we do not place the semicolon at the end of the statement.

d) Imbalanced single quotes and parentheses. Another common mistake is imbalance in single quotes. All string and date values must be placed in single quotes. For example 'Hello will result in a syntax related error as the single quote is not closed. Hence, it should be 'Hello'.

Similarly imbalance in parentheses should be avoided. This mainly happens while using functions.

e) Handling data containing special characters. There would be cases when we would like to insert data containing single quotes. However, since single quotes are used to encapsulate strings, these could be interpreted as the end of the string rather than part of the string. For example, let us take the following example into consideration. I have created a table "info" consisting of two columns "name" and "address". When I try to insert a value "4'th Elm Street" into the address column the single quote character present in the address field conflicts with the single quote character that is part of the syntax. In order to avoid this conflict, you would need to place another

single quote character before the single quote character as shown below in the screen shot.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create table info (name varchar2(10), address varchar2(20));
Table created.
SQL> insert into info values('Harry','4''th Elm Street');
1 row created.
SQL> select * from info
2 ;
```

NAME	ADDRESS
Harry	4'th Elm Street

### ***Be Aware: Common Pitfalls:***

*Not placing character and date fields in single quotes.*

*Incorrect usage of Select Statement.*

*Forgetting statement terminator.*

*Imbalanced single quotes and parentheses.*

*Handling data containing special characters.*

## 10. Solved Examples

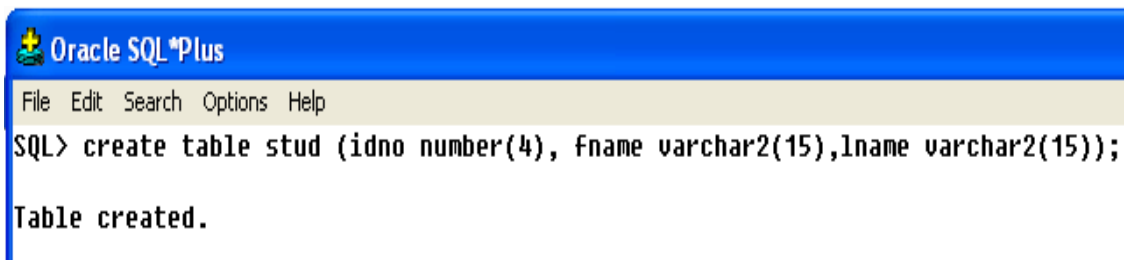
---

1. Create a table called as Stud having the following structure:

Field Name	Data Type	Width
Idno	Number	4
Fname	Character	15
Lname	Character	15

Query

**SQL>** create table stud (idno number(4), fname varchar2(15),lname varchar2(15));



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create table stud (idno number(4), fname varchar2(15),lname varchar2(15));
Table created.
```

Explanation:

In the above create table statement, idno can be any valid 4-digit number, while first and last name fields can have up to 15 characters. Note that the "varchar2" data type has been used for character fields. The "char" data type could have been used as well. In order to understand the reason for using "varchar2" data type, please refer to the section 1.3 explaining the advantages of "varchar2" as compared to "char".

2. Insert the following set of records into the database.

Idno	Fname	Lname
122	Jack	Collins
123	Jill	Mathew

```
SQL> insert all into stud values (122,'Jack','Collins')
2 into stud values (123,'Jill','Mathew')
3 select * from dual;
```

2 rows created.

```
SQL> select * from stud;
```

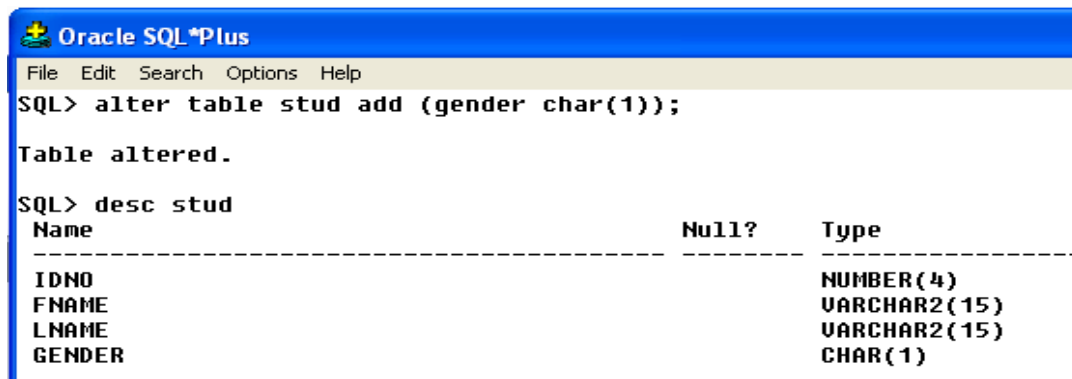
IDNO	FNAME	LNAME
122	Jack	Collins
123	Jill	Mathew

Explanation:

The above query uses "insert all" statement to insert multiple records into the database. The "select \* from dual" statement is required at the end of "insert all" statement to insert multiple records into the database.

3. Write a statement to add a column to the "stud" table named "gender". The data type of this column is "char" with width as 1.

Query:



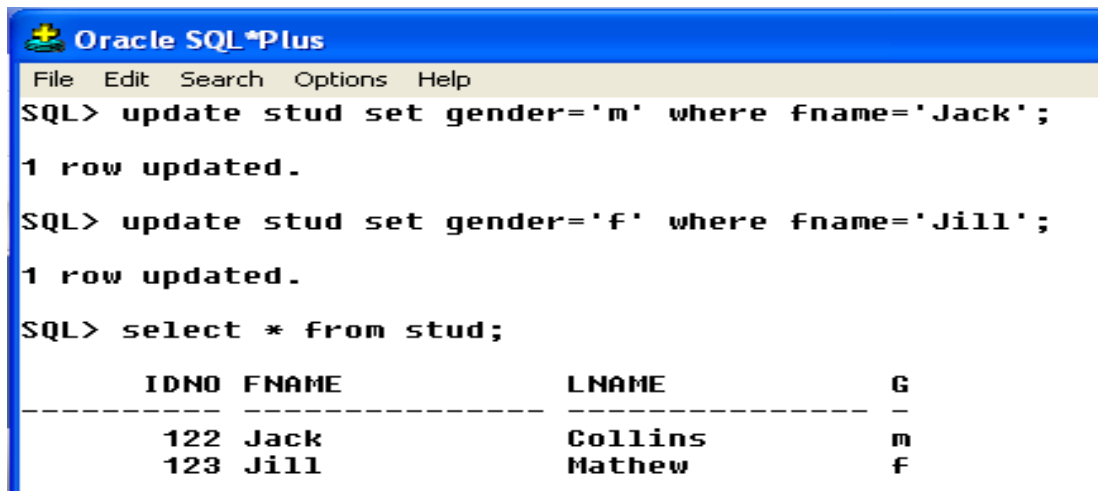
```
Oracle SQL*Plus
File Edit Search Options Help
SQL> alter table stud add (gender char(1));
Table altered.
SQL> desc stud
Name                                     Null?      Type
-----
IDNO                                     NUMBER(4)
FNAME                                   VARCHAR2(15)
LNAME                                   VARCHAR2(15)
GENDER                                  CHAR(1)
```

Explanation:

The above statement uses the "add" clause in the "alter table" statement as a new column is added to an existing table. We would have used the "modify" clause if we had to modify an existing column.

4. Write a statement to update the two records by modifying the gender of the student 'Jack' to 'm' and gender of Jill to 'f'.

Query:



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> update stud set gender='m' where fname='Jack';
1 row updated.
SQL> update stud set gender='f' where fname='Jill';
1 row updated.
SQL> select * from stud;
-----
      IDNO  FNAME          LNAME          G
-----
      122  Jack          Collins         m
      123  Jill          Mathew         f
```

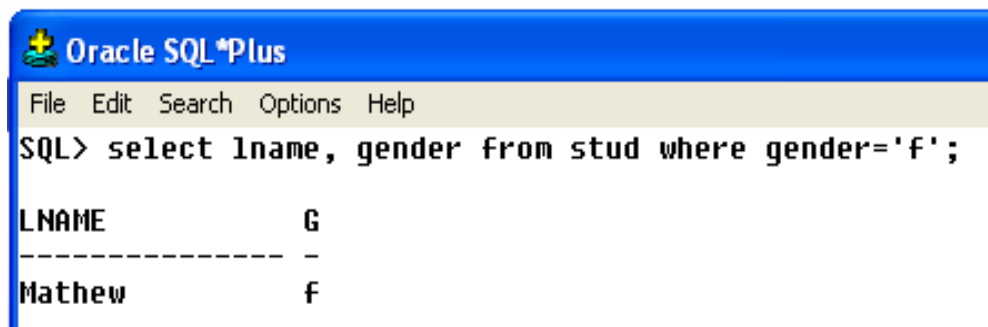
Explanation:

We can use two update statements to update the records. The “where” condition ensures that only the records meeting the criteria stated are updated.

5. Write a query to display the last name and gender of all the female students in the class.

Query:

**SQL>** select lname, gender from student where gender='f';



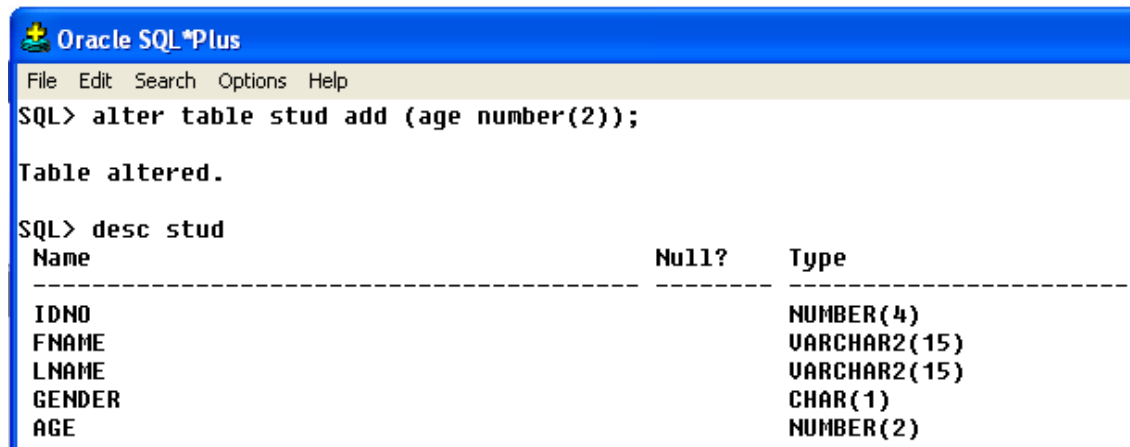
```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select lname, gender from stud where gender='f';
-----
LNAME          G
-----
Mathew         f
```

Explanation:

The above query uses the “=” operator. In this case the last names and gender of all students whose gender is ‘f’ are displayed. Since the comparison is with a fixed string (‘f’), we do not have to use “like”. We would have used “like” if there was a pattern to be compared.

6. Write a statement to add the column “age” to the “stud” table with the width 2.

Query:



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> alter table stud add (age number(2));

Table altered.

SQL> desc stud
Name                                     Null?      Type
-----
IDNO                                     NUMBER(4)
FNAME                                   VARCHAR2(15)
LNAME                                   VARCHAR2(15)
GENDER                                  CHAR(1)
AGE                                     NUMBER(2)

```

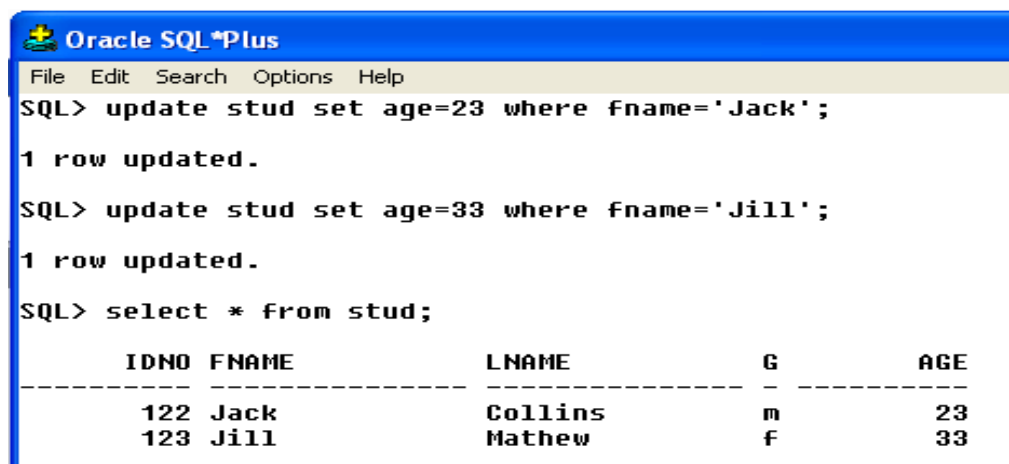
Explanation:

The above statement again uses "add" clause as we are adding a new column. The above column is added in the context of the remaining queries where we are going to use this column.

7. Update the existing records with value for the "age" column.

Query and Explanation:

So following the method as shown in question (4), update the records by adding age to the existing records. The screen shot after updating the records with age is shown below.



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> update stud set age=23 where fname='Jack';

1 row updated.

SQL> update stud set age=33 where fname='Jill';

1 row updated.

SQL> select * from stud;

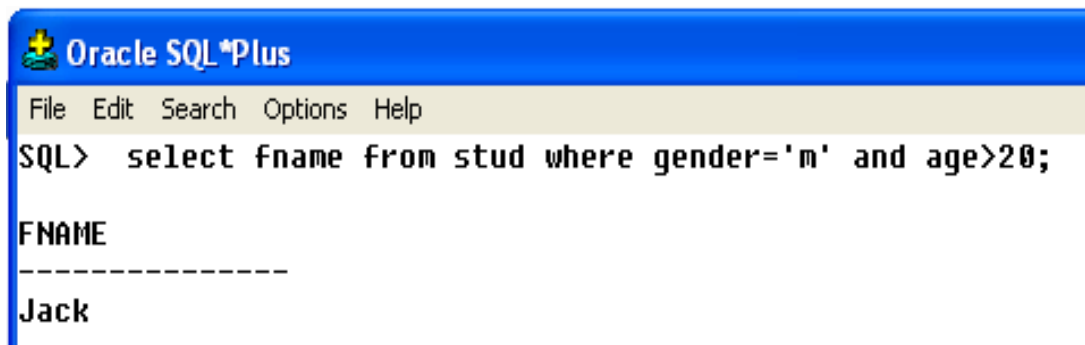
      IDNO FNAME      LNAME      G      AGE
-----
      122 Jack      Collins      m      23
      123 Jill      Mathew      f      33

```

8. Write a query to display the first names of all the male students whose age is greater than 20.

Query:

SQL> select fname from stud where gender='m' and age>20;



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> select fname from stud where gender='m' and age>20;

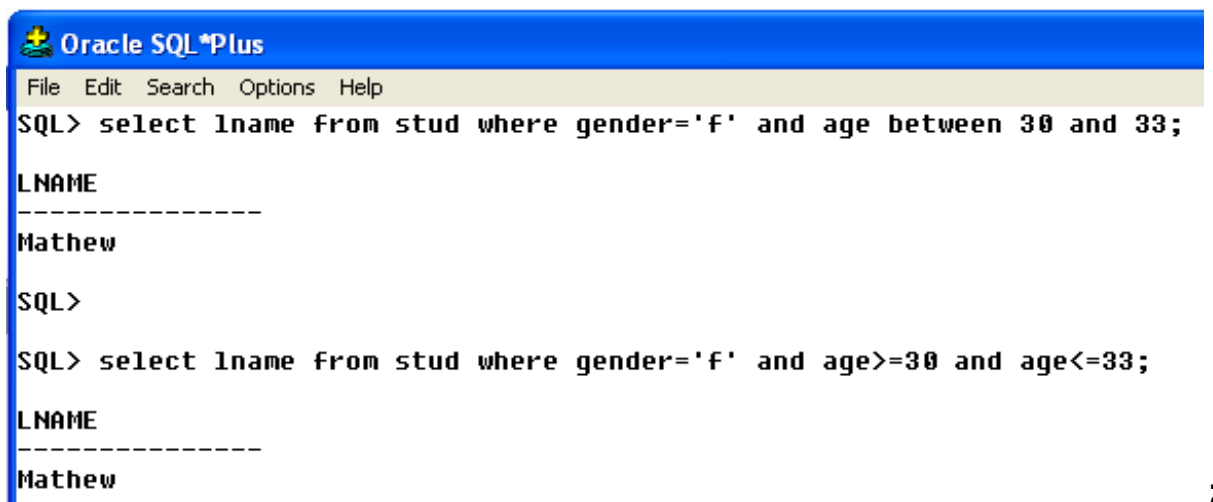
FNAME
-----
Jack
  
```

Explanation:

In this question, we have to take two criteria into consideration. First, the gender must be male and second, the age must be greater than 20. So we have used the "and" operator in order to test both the conditions. So the records displayed as result of this query must meet both the conditions.

9. Write a query to display the last names of all female students who are in the age group 20-23 years.

Query:



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> select lname from stud where gender='f' and age between 30 and 33;

LNAME
-----
Mathew

SQL>

SQL> select lname from stud where gender='f' and age>=30 and age<=33;

LNAME
-----
Mathew
;
  
```

Explanation:

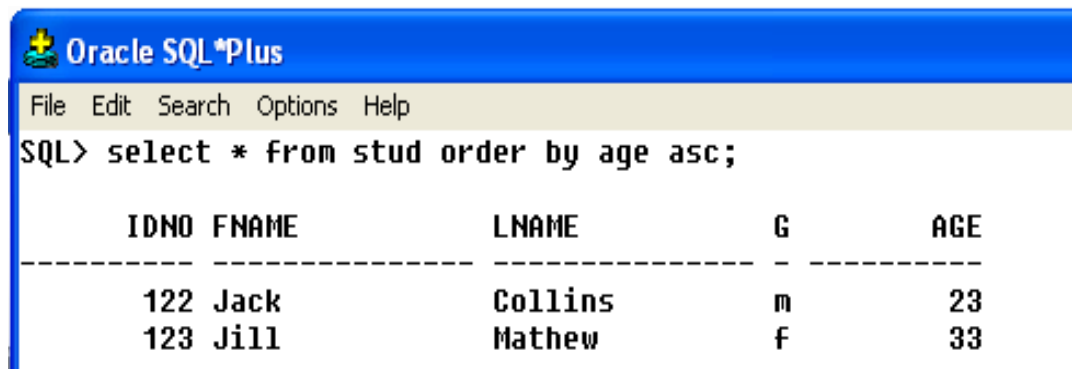
In this question, the "between" clause is used in the where clause of the select statement to display the last name of all female students in the age group. Note that between includes both the lower and upper limit. This query can also be written without using "between" clause, by using the "less



than or equal to "(<=)" and "greater than or equal to (>=)" comparison operators.

10. Write a query to sort all the student records on the basis of their age.

Query:



The screenshot shows the Oracle SQL\*Plus interface. The title bar is blue with the Oracle logo and 'Oracle SQL\*Plus'. The menu bar is yellow with 'File Edit Search Options Help'. The command prompt shows 'SQL> select \* from stud order by age asc;'. The result is a table with columns IDNO, FNAME, LNAME, G, and AGE. The data is sorted by age in ascending order.

IDNO	FNAME	LNAME	G	AGE
122	Jack	Collins	m	23
123	Jill	Mathew	f	33

Explanation:

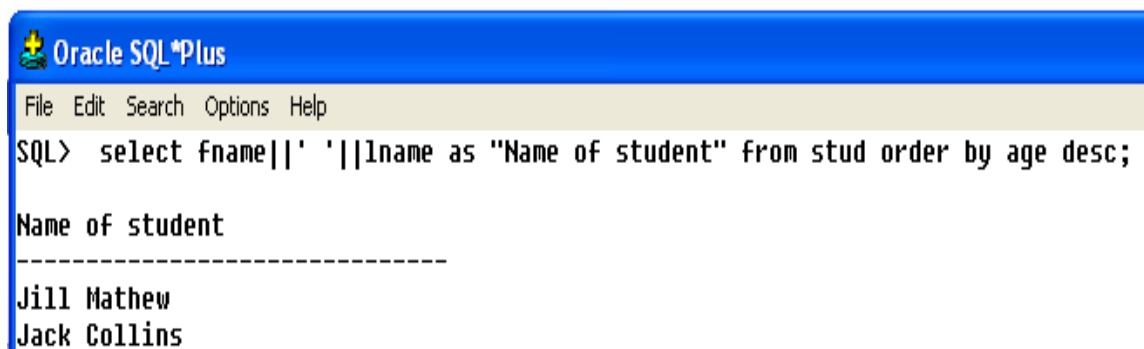
The above **SQL** statement sorts all the student records in the ascending order of age. Note that in this case all the records are sorted in the ascending order of age. The following query would also result in the same result:

**SQL>** select \* from stud order by age;

Since no option was mentioned in the above select statement, all records in result are sorted in ascending order of age because this is the default option.

11. Write a query to display the full names of all male students in descending order of their age.

Query:



The screenshot shows the Oracle SQL\*Plus interface. The title bar is blue with the Oracle logo and 'Oracle SQL\*Plus'. The menu bar is yellow with 'File Edit Search Options Help'. The command prompt shows 'SQL> select fname||' '||lname as "Name of student" from stud order by age desc;'. The result is a table with one column 'Name of student'. The data is sorted by age in descending order.

Name of student
Jill Mathew
Jack Collins

Explanation:

In the above select query, the full names of the students are displayed. The full name is a combination of first name as well as last name of the student. In the above query, this was done using the concatenation operator (||). "Name of student" is the alias name for the string that is obtained as a result of the concatenation of the first and last name field. The final full names are then sorted in the descending order of age.

## 11. Practice Questions

---

1. Write a query to create a table called as Employee with the following fields:

Field Name	Data Type	Width
Empno	Number	4
Ename	Character	20
Salary	Number	8,2

2. Write a query to insert the following records in the Employee table:

1234	Scott	1200.00
3456	Alan	2300.00
1298	King	3200.50
4323	James	4500.00
2857	John	4367.90
2857	John	4637.90

3. As you can see in question 2, the record with empno as 2857 is inserted twice. Delete only one of them.

4. Write a query to increase the width of salary field from 8,2 to 10,2.

5. Write a query to display the names of all employees in ascending order of their salary.

6. Write a query to update the salary of John from 4367.90 to 4400.00.

7. Write a query to display the names of all employees whose salary is in the range 2000 and 4000.

8. Write a query to give a raise of 10% to salary of all employees.

9. Write a query to give a hike of 20 % to salary of all employees whose salary is less than 2000.

10. Write a query to insert a new column called as "joindate", of data type "date".

## 12. Solutions to Practice Questions

---

1. create table Employee (Empno number(4), Ename varchar2(20), Salary number(8,2));

2. insert into Employee values(1234,'Scott', 1200.00);  
insert into Employee values(3456,'Alan', 2300.00);  
insert into Employee values(1298,'King', 3200.50);  
insert into Employee values(1234,'James', 4500.00);  
insert into Employee values(2857,'John', 4367.90);  
insert into Employee values(2857,'John', 4437.90);

3. select rowid, salary from Employee where empno = 2857 and ename = 'John';

This would display all the rowids for the concerned duplicate records. Note that row id is unique for each row or record. In this case we can take the rowid of the record that we intend to delete (salary differs in two records). For example, if the value of rowid is AAAQaQAAEAAAARgAAL , we could issue the following delete statement:

SQL> delete from Employee where rowid='AAAQaQAAEAAAARgAAL';

4. alter table Employee modify (Salary number(10,2));

5. select Ename, Salary from Employee order by Salary asc;

6. update Employee set salary=4400.00 where Empno=2857;

7. select Ename, Salary from Employee where Salary between 2000 and 4000;

8. update Employee set salary=1.1\*Salary;

9. update Employee set Salary=1.2\*Salary where Salary<2000;

10. alter table Employee add (joindate date);

# Glossary

---

**Structured Query Language (SQL)** - This is the standard language for relational databases accepted by both ANSI (American National Standards Institute) and ISO (International Standards Organization).

**Data Definition Language (DDL)** - Data Definition Language statements set up, change and remove structures from tables.

**Data Manipulation Language (DML)** - Data Manipulation Language statements retrieve data from the database, enter new rows, change existing rows, and remove unwanted rows from tables in the database.

**Transaction Control Language (TCL)** - Transaction control language statements manage the changes made by DML statements.

**Transaction** - Transaction basically refers to set of statements that must be executed either completely or not executed at all.

# References

---

- Beaulieu, Alan. 2009. Learning SQL. California: O'Reilly Media.
- Casteel, Joan. 2006. Oracle 10g: SQL. New York: Course Technology.
- Pratt, Phillip. 2008. A Guide to SQL. New York: Course Technology.
- Rischert, Alice. 2009. Oracle SQL by Example. New Jersey: Prentice Hall.
- Rockoff, Larry. 2010. The Language of SQL: How to Access Data in Relational Databases. New York: Course Technology.

# About the Author

---

Rao V.N. has completed Masters in Computer Applications (MCA) from Osmania University in India as well as Masters in Business Administration (MBA) from IUKB, Switzerland, and is currently pursuing graduate studies from University of Liverpool. He holds several certifications including Oracle Certified Associate (OCA), Microsoft Certified Trainer (MCT), Sun Certified Java Professional (SJCP) and Cisco Certified Network Associate (CCNA). He has over 10 years of experience in teaching database related courses at both graduate and under-graduate level.