LAB # 02 CLO2

SQL Commands Introduction

SQL Server Components

Server components	Description
SQL Server Database Engine	SQL Server Database Engine includes the Database Engine, the core service **for storing, processing, and securing data**, Replication, full-text search, and tools for managing relational and XML data.
Analysis Services	Analysis Services includes the tools for **creating and managing online analytical processing (OLAP)** and data mining applications.
Reporting Services	Reporting Services includes **server and client components** for creating, managing, and deploying tabular, matrix, graphical, and free-form reports. Reporting Services is also an extensible platform that you can use to develop report applications.
Integration Services	Integration Services is a set of graphical tools and programmable objects for **moving, copying, and transforming** data.

Management tools	Description
SQL Server Management Studio	SQL Server Management Studio is an integrated environment to access, configure, manage, administer, and develop components of SQL Server. Management Studio lets developers and administrators of all skill levels use SQL Server. Internet Explorer 6 SP1 or a later version is required for Management Studio installation.
SQL Server Configuration Manager	SQL Server Configuration Manager provides basic configuration management for SQL Server services, server protocols, client protocols, and client aliases.
SQL Server Installation Centre	SQL Server Installation Centre is used for the installation of new Instances, Up gradating and updating the SQL Server.

R un it and see the output:

```
SELECT getdate(); -- Selects the current (server) date and time.
```

On the left side you will see databases named master etc. On clicking any of the database, you'll see default tables in that database. You can also right click on any of the table and select 'return all rows' to see the entire values in the table.

But you have to create your own database with your own name.

Run the following query by pressing F5 key:

```
CREATE DATABASE WXYZ; -- Creates a database named WXYZ;
```

Remember…!! SQL is not case sensitive.

Are you able to see the database created…???

If no, then refresh the services again…!!

Now create the table in the above created database using the CREATE TABLE command:

Syntax:

```
CREATE
TABLE        table_name (
column_name1  data_type(size),
column_name2  data_type(size),
column_name3 data_type(size),
....
); --// CREATE TABLE is the keyword.
```

Suppose if you want to create the following table:

Name	Reg_No	Courses	Course_Code	Offered_By

You'll have to run the following query:

```
CREATE TABLE Student
(
Name varchar(255),
Reg_No varchar(255),
Courses varchar(255),
Course_Code int,
Offered_by varchar(255)
);
```

Run the query and see the results.

Was the table created named STUDENT….???

Inorder to verify the results run the following query:

```
SELECT *
FROM Student;
```

Was the table displayed…???

SQL Server Data Types

String types:

Data type	Description	Storage
char(n)	Fixed width character string. Maximum 8,000 characters	Defined width
varchar(n)	Variable width character string. Maximum 8,000 characters	2 bytes+no. of chars
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string. Maximum 2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string. Maximum 4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string. Maximum 4,000 characters	
nvarchar(max)	Variable width Unicode string. Maximum 536,870,912 characters	
ntext	Variable width Unicode string. Maximum 2GB of text data	
bit	Allows 0, 1, or NULL	
binary(n)	Fixed width binary string. Maximum 8,000 bytes	
varbinary	Variable width binary string. Maximum 8,000 bytes	
varbinary(max)	Variable width binary string. Maximum 2GB	
image	Variable width binary string. Maximum 2GB	

Number types:

Data type	Description	Storage
tinyint	Allows whole numbers from 0 to 255	1 byte
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes

bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
decimal(p,s)	Fixed precision and scale numbers.	5-17 bytes
	Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Date types:

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes

Other data types:

Data type	Description
sql_variant	Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML formatted data. Maximum 2GB
cursor	Stores a reference to a cursor used for database operations
table	Stores a result-set for later processing
timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable

SQL DML Commands

SELECT, SELECT DISTINCT, INSERT, UPDATE, DELETE, ORDER BY, AND/OR

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).
The query and update commands form the DML part of SQL:

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database

The DDL part of SQL permits database tables to be created or deleted. It also defines indexes (keys), specify links between tables, and impose constraints between tables.

The most important DDL statements in SQL are:

- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new record or row in a table.

SQL INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

SQL INSERT INTO Example

We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger

Now we want to insert a new row in the "Persons" table.
We use the following SQL statement:

```
INSERT INTO Persons
VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

Insert Data Only in Specified Columns

It is also possible to only add data in specific columns.
The following SQL statement will add a new row, but only add data in the "P_Id", "LastName" and the "FirstName" columns:

```
INSERT INTO Persons (P_Id, LastName, FirstName)
VALUES (5, 'Tjessem', 'Jakob')
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

SQL UPDATE Statement

The UPDATE statement is used to update records in a table

The UPDATE statement is used to update existing records in a table.

SQL UPDATE Syntax

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

SQL UPDATE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

Now we want to update the person "Tjessem, Jakob" in the "Persons" table.
We use the following SQL statement:

```
UPDATE Persons
```

```
SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

SQL UPDATE Warning

Be careful when updating records. If we had omitted the WHERE clause in the example above, like this:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
```

The "Persons" table would have looked like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Nissestien 67	Sandnes
2	Svendson	Tove	Nissestien 67	Sandnes
3	Pettersen	Michael	Nissestien 67	Sandnes

4	Nilsen	Johan	Nissestien 67	Sandnes
5	Tjessem	Jakob	Nissestien 67	Sandnes

SQL DELETE Statement

The DELETE statement is used to delete records in a table.
The DELETE statement is used to delete rows in a table.

SQL DELETE Syntax

```
DELETE FROM table_name
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

SQL DELETE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Now we want to delete the person "Tjessem, Jakob" in the "Persons" table.
We use the following SQL statement:

```
DELETE FROM Persons
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name or
DELETE * FROM table_name
```

Note: Be very careful when deleting records. You cannot undo this statement!

SQL SELECT Statement

Structure of basic select statement

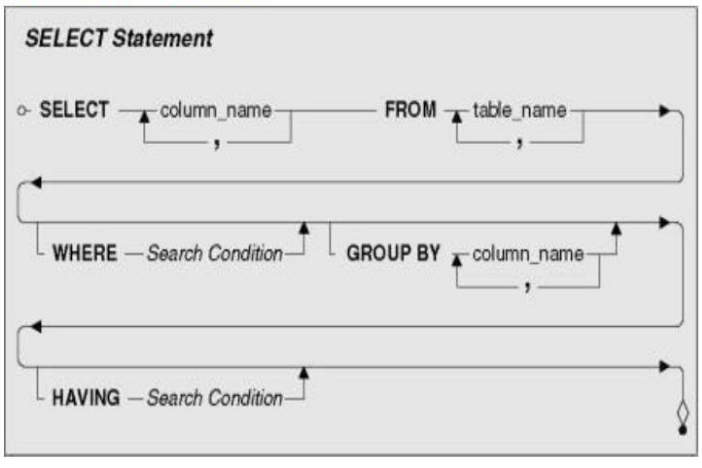

Figure 1: Select statement graphical flow

The SELECT statement is used to select data from a database.The result is stored in a result table, called the result-set.
The syntax used for SELECT query is:

```
SELECT column_name(s)
FROM table_name
```

and

```
SELECT * FROM table_name
```

Note: SQL is not case sensitive. SELECT is the same as select.

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above.
We use the following SELECT statement:

```
SELECT LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
Hansen	Christ
Svendson	Tove
Pettersen	Michael

SELECT * Example

Now we want to select all the columns from the "Persons" table.
We use the following SELECT statement:

```
SELECT * FROM Persons
```

Tip: The asterisk (*) is a quick way of selecting all columns!
The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger

The SQL SELECT DISTINCT Statement

In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table. The DISTINCT keyword can be used to return only distinct (different) values.
Its Syntax is:

```
SELECT DISTINCT column_name(s)
FROM table_name
```

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger

Now we want to select only the distinct values from the column named "City" from the table above.
We use the following SELECT statement:

```
SELECT DISTINCT City FROM Persons
```

The result-set will look like this:

City
Sandnes
Stavanger

The WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion. The syntax of the command is:

```
SELECT column_name(s)
FROM table_name
        WHERE column_name operator value
```

Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Michael	Storgt 20	Stavanger

Now we want to select only the persons living in the city "Sandnes" from the table above. We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE City='Sandnes'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Christ	Timoteivn 10	Sandnes

2	Svendson	Tove	Borgvn 23	Sandnes

Quotes Around Text Fields

SQL uses single quotes around text values (most database systems will also accept double quotes).
Although, numeric values should not be enclosed in quotes.

For text values:

This is correct:

```
SELECT * FROM Persons WHERE FirstName='Tove'
```

This is wrong:

```
SELECT * FROM Persons WHERE FirstName=Tove
```

For numeric values:

This is correct:

```
SELECT * FROM Persons WHERE Year=1965
```

This is wrong:

```
SELECT * FROM Persons WHERE Year='1965'
```

Operators Allowed in the WHERE Clause

With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

Note: In some versions of SQL the <> operator may be written as !=

Practice all SQL commands described above for any table…!!!

TASK 1:

Create the following table using SQL and using the INSERT INTO command, insert the following values in the table created.

Name	Reg_No	Courses	Course_Code	Offered_By
Ali	01	DIP	1001	Mr. A
Basit	02	DBMS	1002	Mr. X
Akram	03	OS	1003	Mr. Y
Asad	04	DBMS	1002	Mr. X
Zeeshan	05	DIP	1001	Mr. A
Muneer	06	OS	1003	Mr. Y
Shafqat	07	NM	1004	Mr. H
Ahsan	08	OS	1003	Mr. Y
Ikram	09	DIP		
Hassan	10			

TASK 2:

Using the UPDATE statement, update the above table for the following values:

Name	Reg_No	Courses	Course_Code	Offered_By
Ali	01	DIP	1001	Mr. A
Basit	02	DBMS	1002	Mr. X
Akram	03	OS	1003	Mr. Y
Asad	04	DBMS	1002	Mr. X
Zeeshan	05	DIP	1001	Mr. A
Muneer	06	OS	1003	Mr. Y
Shafqat	07	NM	1004	Mr. H
Ahsan	08	OS	1003	Mr. Y
Ikram	09	DIP	1001	Mr. A
Hassan	10	DSP	1005	Mr. Z

TASK 3:

Using the DELETE statement, delete the record for the student having name Akram and Ahsan in the above table. Also delete the record for the course having course code=1001.

TASK 4:

Select distinct values from the above table for the last three columns.

TASK 5:

For the table in task 2, generate a query for updating the table with fully qualified names and update the following values:

Ali	01	SE	1001	Mr. Z
Basit	02	CG	1002	Mr. X