Database Lab

CSE 3104

Lab-07

7.1 SQL DROP COMMAND

DROP command completely removes a table from a database. This command will also destroy the table structure. Also a database can be drop with this command.

Basic Structure-

```
DROP TABLE "table_name"

DROP DATABASE DatabaseName;
```

Example-

```
DROP TABLE COURSE

DROP DATABASE EMPLOYEE
```

NOTE: You can also DROP multiple by using comma (,)

7.2 SQL TRUNCATE COMMAND

To remove all the rows from a table, the TRUNCATE command is used. TRUNCATE deletes all the rows, but the table structure remains the same. TRUNCATE is a DDL command.

When we apply Truncate command on a table then its PRIMARY KEY is initialized.

Basic Structure-

```
TRUNCATE TABLE  "table_name"
```

Example-

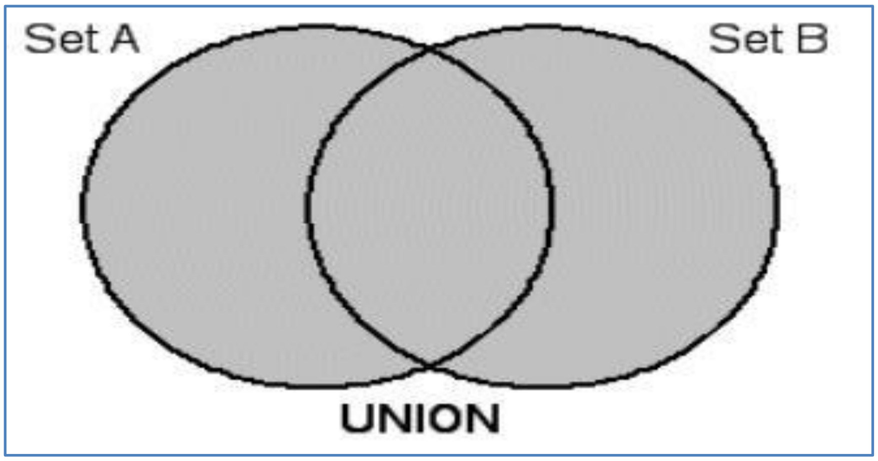
```
TRUNCATE TABLE STUDENT
```

NOTE: TRUNCATE & DELETE commands are not same. Delete command will delete all the rows from a table where as Truncate command re-initializes a table.

7.3 SQL UNION

The SQL **UNION operator** is used to combine the result sets of 2 or more SELECT statements. It removes duplicate rows between the various SELECT statements.

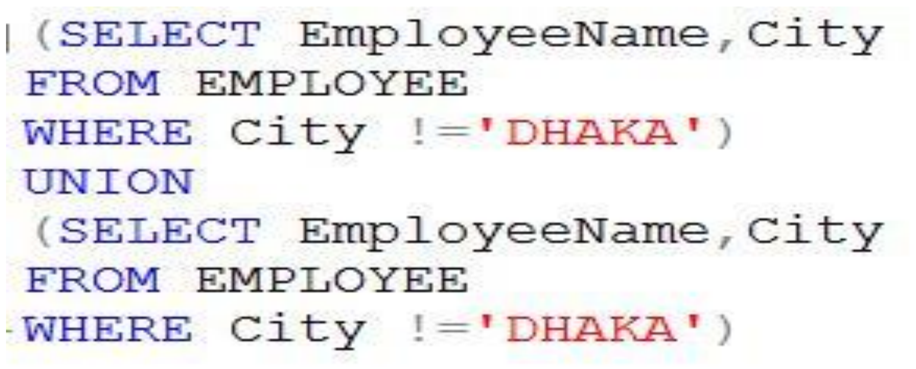
Each SELECT statement within the UNION must have the same number of fields in the result sets with similar data types, also same order, but they do not have to be the same length.



Basic Structure-

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2
```

Example-

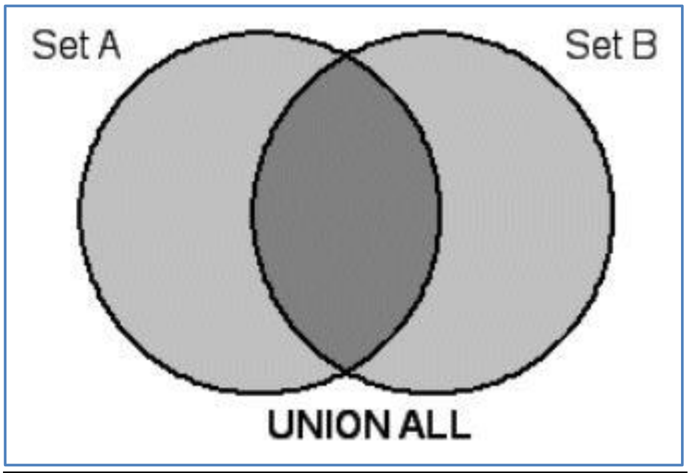


```
(SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA')
UNION
(SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA')
```

7.4 SQL UNION ALL

Same as UNION. The purpose of the SQL **UNION ALL** command is to combine the results of two queries together.

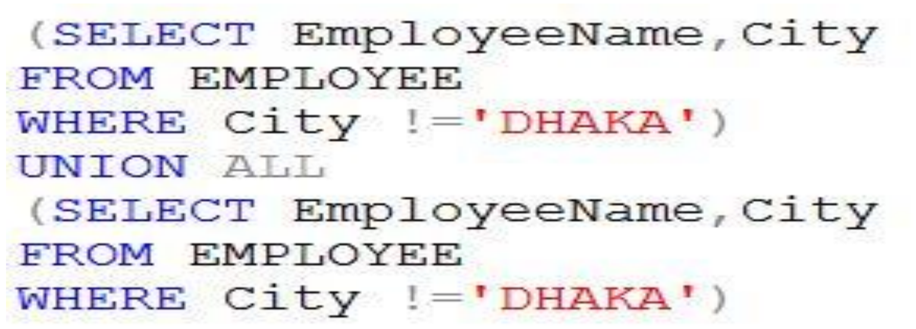
UNION and **UNION ALL** both combine the results of two SQL queries. The difference is that, while **UNION** only selects distinct values, **UNION ALL** selects all values.



Basic Structure-

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2
```

Example-

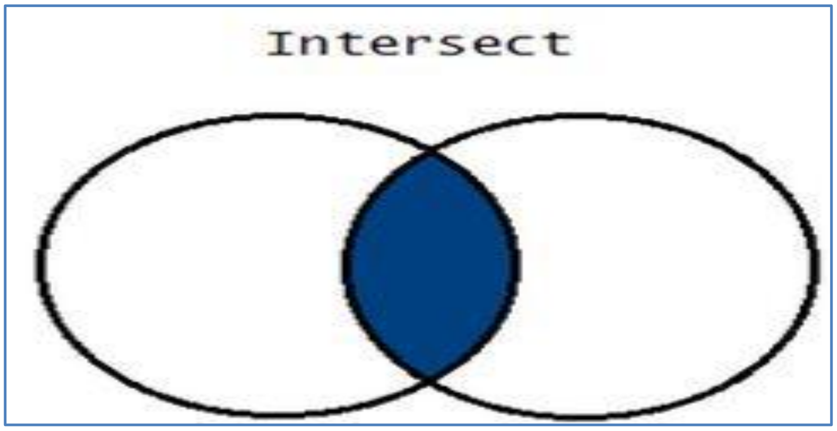


```
 (SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA')
UNION ALL
 (SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA')
```

7.5 SQL INTERSECT

The SQL **INTERSECT** clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

Just as with the UNION operator, the same rules apply when using the INTERSECT operator.



Basic Structure-

SELECT column_name(s)

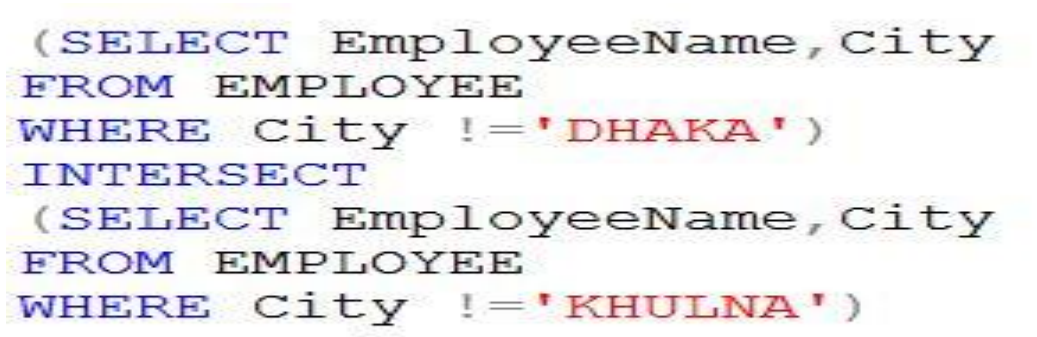
FROM table

INTERSECT

SELECT column_name(s)

FROM table

Example-

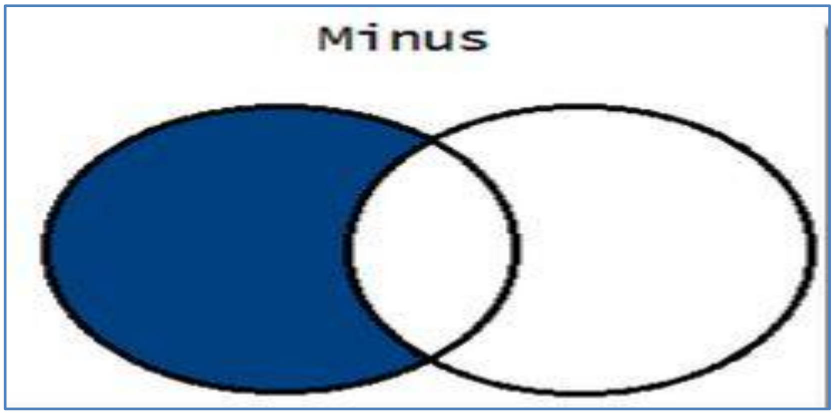


```
(SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA')
INTERSECT
(SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='KHULNA')
```

7.5 SQL EXCEPT

The **EXCEPT** command operates on two SQL statements. It takes all the results from the first SQL statement, and then subtract out the ones that are present in the second SQL **statement to get the final answer. If the second SQL statement includes results not present in the first SQL statement, such results are ignored**.

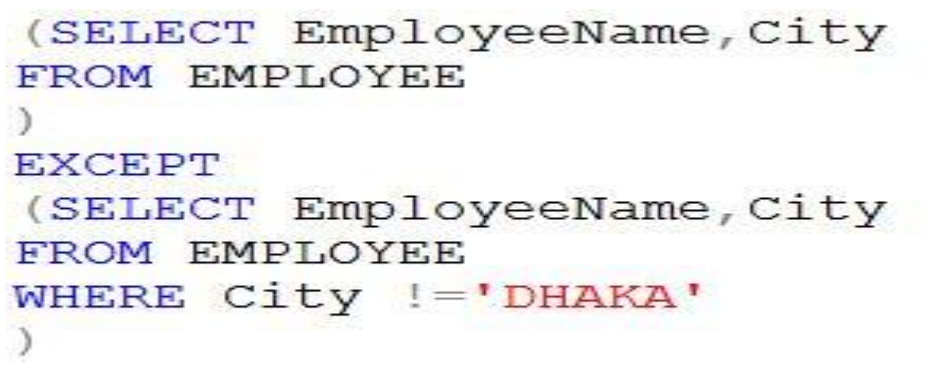
Each SELECT statement within the EXCEPT query must have the same number of fields in the result sets with similar data types.



Basic Structure-

```
SELECT column_name(s)

FROM table

MINUS

SELECT column_name(s)

FROM table
```

Example-



```
(SELECT EmployeeName,City
FROM EMPLOYEE
)
EXCEPT
(SELECT EmployeeName,City
FROM EMPLOYEE
WHERE City !='DHAKA'
)
```

7.6 ANY, ALL with a Multiple Row Sub query:

Both are used for comparing one value against a set of values. ALL specifies that all the values given in the list should be taken into account, whereas ANY specifies that the condition is satisfied when any of the values satisfies the condition.

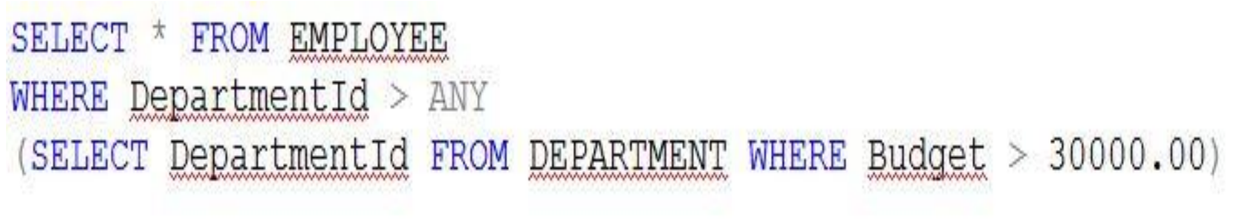
The operator can be any one of the standard relational operators (=, >=, >, <, <=, !=) , and

list is a series of values.

```
operator ANY list

operator ALL list
```

The ANY keyword denotes that the search condition is TRUE if the comparison is TRUE for at least one of the values that is returned. If the subquery returns no value, the search condition is FALSE. **The SOME keyword is a synonym for ANY**.

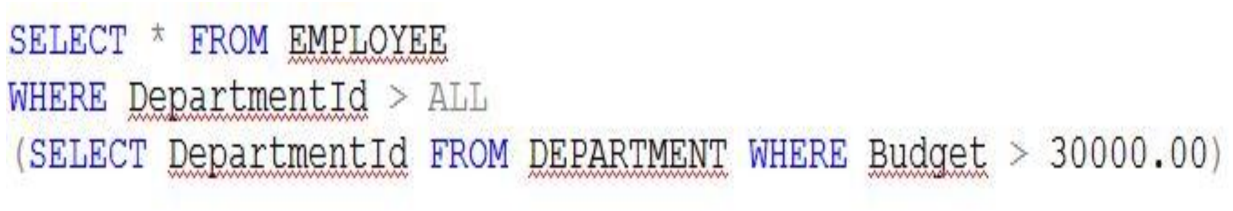
Example-



```
SELECT * FROM EMPLOYEE
WHERE DepartmentId > ANY
(SELECT DepartmentId FROM DEPARTMENT WHERE Budget > 30000.00)
```

The ALL keyword specifies that the search condition is TRUE if the comparison is TRUE for every value that the subquery returns. If the subquery returns no value, the condition is FALSE.

Example-



```
SELECT * FROM EMPLOYEE
WHERE DepartmentId > ALL
(SELECT DepartmentId FROM DEPARTMENT WHERE Budget > 30000.00)
```

7.7 Using EXISTS, NOT EXISTS with a Subquery:

The EXISTS operator checks if a subquery returns any rows. The following illustrates the syntax of the EXISTS operator:

WHERE EXISTS (subquery)

The expression EXISTS (subquery) returns TRUE if the subquery returns at least one row, otherwise it returns FALSE. Notice that you put the subquery inside the parentheses followed by the EXISTS operator.

NOT operator with the EXISTS operator to inverse the meaning of the EXISTS operator.

WHERE NOT EXISTS (subquery)

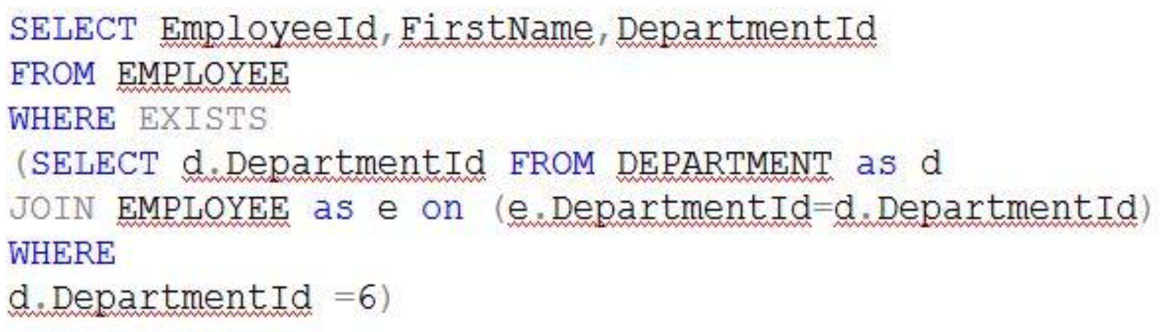
The expression NOT EXISTS (subquery) returns TRUE if the subquery returns no row, otherwise it returns FALSE.

NOTE:

EXISTS is different from other operators like IN,ANY etc., because it doesn't compare values of columns, instead, it checks whether any row is retrieved from subquery or not.

Example-

<u>EXISTS</u>



```
SELECT EmployeeId,FirstName,DepartmentId
FROM EMPLOYEE
WHERE EXISTS
(SELECT d.DepartmentId FROM DEPARTMENT as d
JOIN EMPLOYEE as e on (e.DepartmentId=d.DepartmentId)
WHERE
d.DepartmentId =6)
```

Here in DepartmentId 6 there is no employee, so sub query returns Nothing. For this, overall outer query shows nothing.

Example-

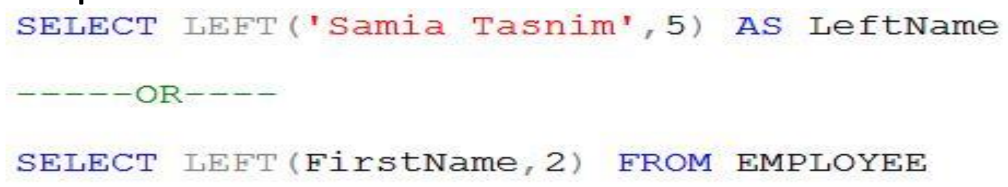
NOT EXISTS

'TRY FOR ABOVE QUERY'

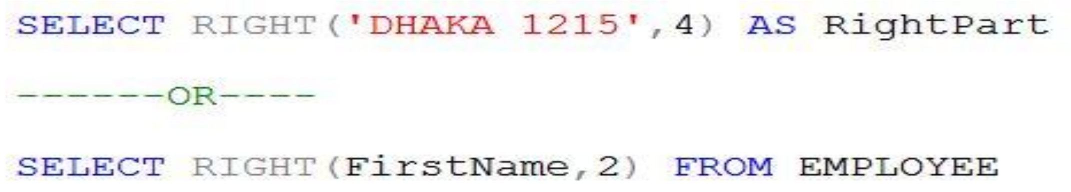
SQL STRING FUNCTIONS

Sql string function is a built-in string function.
It perform an operation on a string input value and return a string or numeric value.

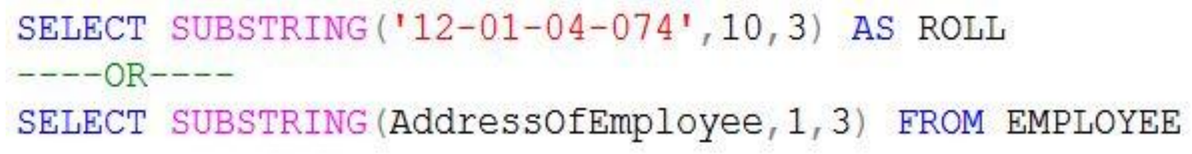
Some common sql string functions:

1. LEFT - Returns left part of a string with the specified number of characters.
   Syntax-
   LEFT (string , integer)
   Example-
   

```
SELECT LEFT('Samia Tasnim',5) AS LeftName

------OR----

SELECT LEFT(FirstName,2) FROM EMPLOYEE
```

2. RIGHT - Returns Right part of a string with the specified number of characters.
   Syntax-
   RIGHT (string , integer)
   Example-
   

```
SELECT RIGHT('DHAKA 1215',4) AS RightPart

-------OR----

SELECT RIGHT(FirstName,2) FROM EMPLOYEE
```

3. SUBSTRING - Returns part of a string.
   Syntax-
   SUBSTRING (string, startindex , length)
   Example-
   

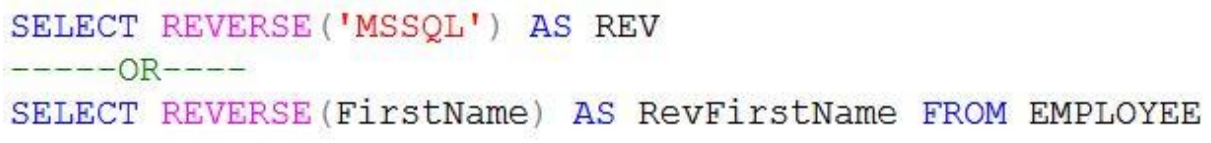
```
SELECT SUBSTRING('12-01-04-074',10,3) AS ROLL
----OR----
SELECT SUBSTRING(AddressOfEmployee,1,3) FROM EMPLOYEE
```

4. REVERSE - Returns reverse a string.

Syntax-

REVERSE(string)

Example-



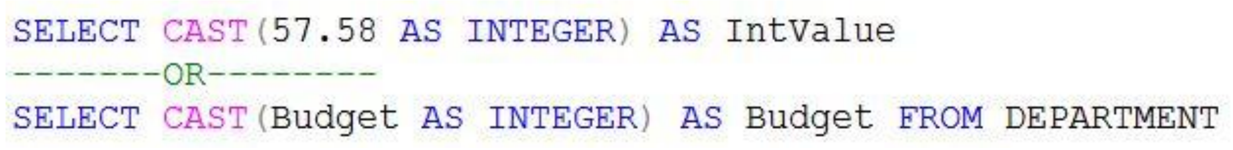
```
SELECT REVERSE('MSSQL') AS REV
------OR----
SELECT REVERSE(FirstName) AS RevFirstName FROM EMPLOYEE
```

5. CAST - Returns the value of an expression converted to a supplied data type.

Syntax-

CAST (expression AS [data type])

Example-



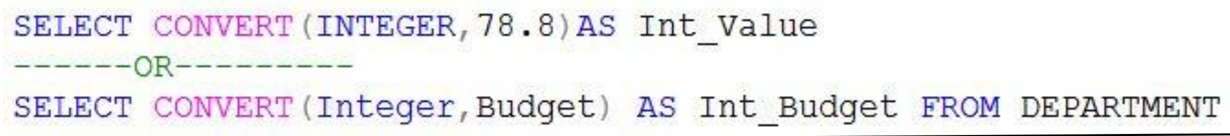
```
SELECT CAST(57.58 AS INTEGER) AS IntValue
--------OR---------
SELECT CAST(Budget AS INTEGER) AS Budget FROM DEPARTMENT
```

6. CONVERT - Converts a value to another data type. Similar to CAST.

Syntax-

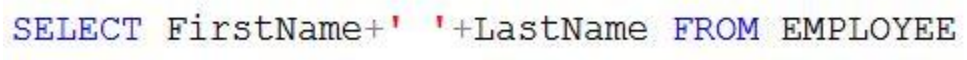
CONVERT (expression, [data type])

Example-



```
SELECT CONVERT(INTEGER,78.8)AS Int_Value
-------OR---------
SELECT CONVERT(Integer,Budget) AS Int_Budget FROM DEPARTMENT
```

7. CONCAT - This Keyword not use in SQL, But we can CONCAT two part as –



```
SELECT FirstName+' '+LastName FROM EMPLOYEE
```

SQL DATE FUNCTIONS

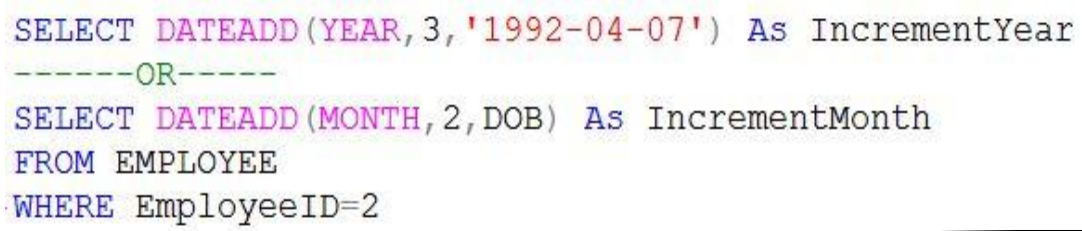
SQL Server date and time functions are scalar functions that perform an operation on a date and time input value and returns either a string, numeric, or date and time value.
Some useful Date Functions-

1. DATEADD - Returns a new datetime value based on adding an interval to the specified date.

Syntax-

DATEADD (datepart , number, date)

Example-



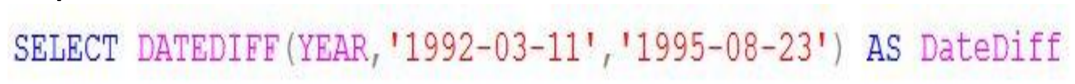
```
SELECT DATEADD(YEAR,3,'1992-04-07') As IncrementYear
------OR-----
SELECT DATEADD(MONTH,2,DOB) As IncrementMonth
FROM EMPLOYEE
WHERE EmployeeID=2
```

2.DATEDIFF - Returns the number of date and time boundaries crossed between two specified dates.

Syntax-

DATEDIFF (datepart , startdate , enddate)

Example-



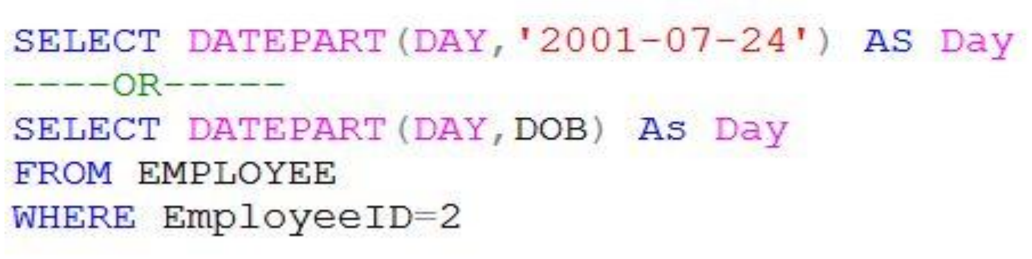
```
SELECT DATEDIFF(YEAR,'1992-03-11','1995-08-23') AS DateDiff
```

3. DATEPART - Returns an integer that represents the specified datepart of the specified date.

Syntax-

DATEPART (datepart , date)

Example-



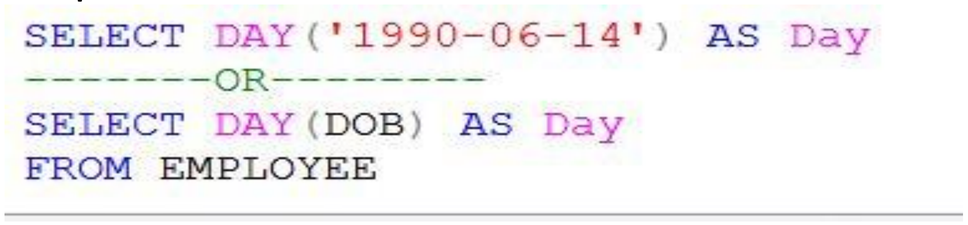
```
SELECT DATEPART(DAY,'2001-07-24') AS Day
-----OR------
SELECT DATEPART(DAY,DOB) As Day
FROM EMPLOYEE
WHERE EmployeeID=2
```

4.DAY - Returns an integer representing the day datepart of the specified date.

Syntax-

DAY (date)

Example-



```
SELECT DAY('1990-06-14') AS Day
--------OR---------
SELECT DAY(DOB) AS Day
FROM EMPLOYEE
```

Like DAY Function MONTH, YEAR also hold same structure.

5.The GETDATE function is used to retrieve the current database system time in SQL Server.

Syntex-

SELECT GETDATE()

Self Study:

Practice all the commands related to today's class. A practical session will be conducted at the very beginning of the next lab.