SQL

1.What is data?

Collection of facts/particulars

2.Where is data stored?

Excel files or even text files

Could also have database and table. Database can have multiple tables.

3. Each column has their specific data type.

4. For Query to work structured data is a must.

5. A table consist of Column/Fields and Rows/Records. And what actually is stored is called Value.

6. For a Database Query we need:

    1. Atleast 1 Table
    2. Username and password to access the data.

****

We need to make sure we select all the columns for which are going to be used in query to make sure query is working correctly.

****

How can we import the data from excel file

Select Database name right click select “Tasks” and then select “Import Data” then Select Data source as “Microsoft excel” and then “Browse” next “Copy Data” Select “sheets” the data we want to transfer and click “Finish”. Now refresh the data base to see the data.

Steps to disable Intellisense:

Tool->Options->Text Editor->Transact-SQL->Intelliesense->Disable the feature and click “OK.

/* “To start block comment” “End With” */

--To write single line comment.

SQL Syntax

What to Select???

SELECT [All/Distinct] column_name1, column _name2…….

SELECT* would select all the columns

SELECT column_name, column_name **Select Specified columns (“,” is must to)**

SELECT DISTINCT column_name, column_name **Select distinct values across specified columns**

Where to Select???

FROM table_name

Other Optional Statements

Where conditions e.g., column1_name = ‘value1’

WHERE Column_name Operator Value [[AND/OR]column_name operator value] **Select subsets of Data**

Operator	Description	Operator	Description
=	Equal	<= OR LE	Less than or equal
<> OR != OR NE	Not Equal		
> OR GT	Greater Than	BETWEEN	Between an inclusive range
< OR LT	Less Than	LIKE	Search for Pattern
>= OR GE	Greater than or Equal	IN	Specify multiple values for a column

Types of operators

Comparison Operators:

>, <, in, Between…And; LIKE; IN

Logical Operators:

AND OR

Arithmetic Operators:

+,-,/,*

ORDER BY column_name1, column_name2……. **ASC|DESC**

We include order by statement after **SELECT** and **FROM. WHERE** condition is optional if we have it we include it after **From** and Before **ORDER BY.** But if **WHERE** is eliminated then we can include **ORDER BY** directly after **FROM** clause.

- Default **SORT ORDER** is **ASCENDING.**
- To sort in Descending order we use key Word **DESC.**

****

Example:

Where id = 1 **"**Value doesn't need to be in quotes if the data type is **"integer" " where column name operator value**

Where firstname = 'John' "Firstname is **"varchar"** or **"nvarchar"** so the value needs to be placed between **'Value' Where columne_name Operator 'Value'** same applies for **datetime** even**.**

Where birthdate > '03/23/1985' and firstname = 'satya' for **using** and we can **fetch data** using **multiple columns** and even **mix and match operators**

Syntax for Between….And :

WHERE column_name **BETWEEN** 'value' **AND** 'value'

Syntax for IN :

Where column_name **IN (**'value1' , 'value2', 'value3')

Syntax for LIKE :

WHERE column_name **Like** 'Value%'

Exanples : WHERE FIRSTNAME LIKE 'S%' ----- Would execute all the firstnames starting with 's'

WHERE FIRSTNAME LIKE '%S%' ----- 'S' must exist anywhere in the name, prceded or followed by any character

Where Firstname Like '_a%' --- "Using the **"_" Wildcard** to **substitute** just **one character".**

WHERE FIRSTNAME LIKE '__e' --- "Using this, we would get results of name with **3 letters** ending with **'E'.**

WHERE FIRSTNAME LIKE '[abs]%' -- "First character can be any out of **a,b,s** followed by other characters**"**

WHERE FirstNAme LIKE '[a-e]%' --- "Would give all the **names** starting with **a,b,c,d,e** followed by other characters.

****

Logical Operators:

"AND" Operator Syntax:

WHERE condition1 And Condition2 ---All conditions must be true

"OR" Operator Syntax:

WHERE condition1 OR condition2 -----At Least One Condition must be true.

"NOT" Operator Syntax:

WHERE NOT condition -----condition must be false

Example:

Where lastname = 'Rao' AND firstname like 'p%' ---- would give output if both are true

Where Birthdate <= ' 01/06/1980' AND firstname = 'steve' ---- would give output if both are true

Where lastname = 'Rao' OR firstname like 's%' --------Would give results even if one condition is fulfilled.

Where AGE = 33 AND NOT Gender = 'F' ------- Would give us result of all the male names in Database with Age 33 and reject all the Females

Arithmetic Operators

Arithmetic Operator	Description	Precedence
/	**Division**	**1**
%	**Modulo….**returns the reminder of numerator divided by denominator	**2**
*	**Multiplication**	**2**
+	**Addition**	**3**
-	**Subtraction**	**4**

"BODMAS" rule of math applies while implementing arithmetic operators in querys.

Arithmetic Operator….Samples

1.>SELECT column_name **airtmetic operator value,** column_name…….

From table_name

Where column_name **comparison operator value**

2.> SELECT column_name**,** column_name……

From table_name

Where column_name **airtmetic operator value comparison operator value**

EXAMPLE:

Would give us age what would be after 2 years: addition

SELECT lastname, firstname, trip1_expense, age, age+2

FROM MainContact

Would give us age what would be before 2 years: subtraction

SELECT lastname, firstname, trip1_expense, age, age-2

FROM MainContact

****Multiplication****

SELECT lastname, firstname, age, trip1_expense, trip1_expense*2

From MainContact

Where trip1_expense*2<10000 **----would give results whose expenses would be less then 10000 after multiplying by 2**

SELECT lastname, firstname, age, trip1_expense, trip1_expense*2

From MainContact

Where age+2<=15 **-------would give us results of ppl who would be less or equal to 15 after 2 years**

******Division*****

SELECT lastname, firstname, age, trip1_expense, trip1_expense/12

From MainContact

Where age =13 **--------this would give us after dividing there expenses among other 12 ppl.**

******Operator Precedence*******

SELECT lastname, firstname, age, trip1_expense, trip1_expense*2, trip1_expense *2 -50

From MainContact

Where age+2<=15 **----Counsel would display the output for the data adding two new columns with (tripexpense*2) and (tripexpense*2-50)**

SELECT lastname, firstname, age, trip1_expense, trip1_expense*2, trip1_expense -50*2

From MainContact

Where age+2<=15 **----Counsel would display the output for the datalast column would subtract 100 from trip expense**

Aliases

What & Why?

Aliases are "alternate names" used for Columns and Tables to have more meaningful names.

WHERE TO USE?

Functions in query

Shorten lengthy column names or make them longer

Combine columns, or create a new column

Join two or more tables

Syntax:

SELECT column_name as column_alias, column_name…… **–Can use "as column alias"**

FROM table_name

SELECT column_name column_alias, column_name…… **–Can use only "Column alias"**

FROM table_name

SELECT table_alias.column_name, table_alias.column_name… **-When we use Table alias**

FROM table_name table_alias

SELECT table_alias.column_name as column_alias ---When we combine table alias also with column alias

FROM table_name table_alias

****While Using Alias we can implement any operators in its standard syntax format.****

Functions

Aggregate Functions

Functions	Description
COUNT	Number of records
MIN	Smallest value of a column
MAX	Largest value of column
SUM	Summation of a column value
AVG	Average value of a Column

Syntax for COUNT

```
SELECT count(*)  -----Count of records in the table
From table_name

SELECT count(column_name)  ---Count of values for a specific column
FROM table_name

SELECT count( Distinct column_name)  ---Count distinct values for a specific column
FROM table_name
```

Syntax for MIN() & MAX() Functions:

```
SELECT Min(column_name) as column_alias  --- smallest value of specific column
FROM table_name

SELECT Min(column_name) as column_alias  --- Largest value of specific column
FROM table_name
```

Syntax for SUM() & AVG() Functions:

```
SELECT SUM(column_name) as column_alias  --- sumation value of specific column
FROM table_name

SELECT AVG(column_name) as column_alias  --- average values of specific column
FROM table_name
```

Count, Min(), Max(), Sum(), Avg() displays in different color while writing query by which we know we using an function.

GROUP BY Clause:

Utilize to aggregate or group results for one or more columns

Used along with Aggregate Functions, eg. SUM, AVG etc.

***All columns without aggregate functions in the **SELECT CLAUSE, MUST APPEAR in the Group by clause.**

Must include the **column_name** in **SELECT** on which we implementing function **GROUP BY** to see the function is implemented **correctly.**

IF you put the **Column_name** on which **Aggregate Function** is performed under **Group BY** then the output would display whole data without sorting out **Anything.**

GROUP BY CLAUSE SYNTAX:

SELECT column_name, **Aggregate_function(column_name) as column_alias**

FROM table_name

WHERE column_name operator value ---optional

GROUP BY column_name

ORDER BY column_name ---OPTIONAL

Having Clause:

Filters records for aggregated data**;** it's like **Where** clause

Having works with **Aggregate Functions,** but **Where** doesn't Column(s) in **Having Clause Must Appear** in **Group By or Must Have an Aggregate function.**

Where clause is always processed first before grouping data and applying **Having Clause** filters.

Using **Aggregate_function** post to **Having** is very Important if we miss it then we would get an **error**.

Having Clause- Syntax

SELECT column_name, **Aggregate_function(column_name) as column_alias**

FROM table_name

WHERE column_name operator value ---optional

GROUP BY column_name

Having aggregate_function(Column_name) Operator Value

ORDER BY column_name ---OPTIONAL

String Functions:

Function	Description
UPPER, UCASE	CONVERTS DATA TO UPPER CASE
LOWER, LCASE	CONVERTS DATA TO LOWER CASE
TRIM	REMOVES THE TRAILING AND LEADING BLANK SPACES
LTRIM	REMOVES THE LEADING BLANK SPACES (LEFT)
RTRIM	REMOVES THE TRAILING BLANK SPACES (RIGHT)
CONCAT, +, ||	CONCENTENATES TWO OR MORE STRING CHARACTERS
LEN, LENGTH	GETS THE LENGTH OF DATA
SUBSTRING, SUBSTR	GETS A SPECIFIED PORTION OF THE DATA IN A COLUMN

UPPER &LOWER FUNCTION SYNTAX

SELECT UPPER (column_name) as column_alias --- CONVERTS DATA TO UPPER CASE

FROM table_name

SELECT LOWER (column_name) as column_alias --- CONVERTS DATA TO LOWER CASE

FROM table_name

TRIM, LTRIM & RTRIM FUNCTIONS

TRIM --- REMOVES THE LEADING & TRAILING BLANK SPACES ---Does not work with Microsoft SQL sever 2008

```
SELECT TRIM(column_name) as column_alias
FROM table_name

SELECT LTRIM(column_name) as column_alias  --- Removes the leading blank spaces (Left)
FROM table_name

SELECT RTRIM(column_name) as column_alias  --- Removes the Trailing blank spaces (Right)
FROM table_name
```

CONCAT Function **--Concatenates two or more string**

--Works with SQL server 2012 server and above

```
SELECT CONCAT(( string_value1, string_value2, [,string_valueN]))
FROM table_name
```

--Other ways to concatenate strings

```
SELECT column_name + column_name as column_alias
FROM table_name

SELECT column_name || column_name as column_alias
FROM table_name
```

LEN Function **-- Length of data in a column**

```
SELECT column_name, LEN(Column_name) as column_alias
FROM table_name

SELECT column_name, LENGTH(Column_name) as column_alias
FROM table_name
```

Substring Function **--Select a portion of data in a column**

```
SELECT SUBSTRING(Column_name,starting_position,length)
FROM table_name

SELECT SUBSTR(Column_name,starting_position,length)
FROM table_name
```