Select Statement

Select \* from <TableName>

\*-> point to the all columns

(Performance wise specifying column name is recommended instead of specifying \*)

Select Col1, Col2, Col3,….,Coln from <TableName>

Fully qualified name of table

We can run table from any db context by using fully qualified name

**[dbName].[schemaName].[tableName]**

Distinct

1. Select distinct colName from <tableName>
2. Select distinct ColName1, ColName2 from <tableName>

Where clause

Where clause is used to filter records.

Select \* from <TableName> where <columnName> = ’Value’

Select \* from <TableName> where <columnName> <> ’Value’

Select \* from <TableName> where <columnName> != ’Value’

Select \* from <TableName> where <columnName> in (’Value1’,’Value2’,’Value3’)

Select \* from <TableName> where <columnName> =’Value1’ or <columnName> = ‘Value2’ or <columnName> = ‘Value3’

Select \* from <TableName> where <columnName> between Value1 and Value2

Select \* from <TableName> where <columnName> >= Value1 And <columnName> <= Value2

Select \* from <TableName> where <columnName> like ‘L%’

Select \* from <TableName> where <columnName> not like ‘L%’

Select \* from <TableName> where <columnName> like ‘%@%’

Example:

Select \* from <tableName> where <columnName1> like ‘\_@\_.com’

Select \* from <tableName> where <columnName1> like ‘[MFTS]%’

Select \* from <tableName> where <columnName1> like ‘[^MFTS]%’

Select \* from <tableName> where <columnName1> like ‘[^MFTS]%’

Joining Multiple Conditions: (AND/OR)

Select \* from <tableName> where (<columnName1=’Value1’ or <columnName1>=’Value2’) and (<columnName2> > Value3)

Wild cards used

1. % - Specifies 0 or more characters
2. \_ - Specifies exactly one character
3. [ ] – Specifies any character within the bracket
4. [^ ] – Specifies not any characters within the bracket

Order By clause

Select \* from <tableName> order by <columnName1> [, <columnName2>,…] ASC/DESC

Select \* from <tableName> order by <columnName1> desc , <columnName2> Asc

Top N & Top n% Records

Select top n \* from <tableName>

Select top n col1, col2, col3 from <tableName>

Select top 1 percent \* from <tableName>

Select top 50 percent \* from <tableName>

Q. Finding Eldest person in table person.

Select top 1 \* from TblPerson order by Age Desc

Comparison Operators used in subquery

1. In
2. Any
3. All

select \* into & insert into select statements:

select \* into

It selects data from one table and insert selected data into new table (non-existing table).

It copies all rows and columns from existing table into a new table.

This is used for making backup copy of existing table.

select \* into statement can’t be used to transfer data from one table into another already existing table. It causes error if we do so.

1. **Select \* into <newTableName> from OldTable**

If we specify any existing table in newTableName then it will raise error

1. **Select \* into [dbName].[dbSchema].<newTableName> from [dbName].[dbSchema].OldTable**

It copies all rows and columns from existing tables into a new table in an external database.

1. **Select ColName1, ColName2, ColName3 into <newTableName> from <oldTableName> where <ColName> = ’Value’**

It copies only selected column and rows of existing table into new table

1. **Copies data from two or more tables into new table**

Select \* into EmployeeBackup from Employee inner join Bank

on Employee.AccountNumber=Bank.AccountNumber

Select \* into Customers Select \* from IndiaCustomer Union All Select \* from USACustomer

1. **Create new table only with same schema/structure but with no data**

Select \* into EmpBackup from Employee where 1<>1

Select top 0 \* into EmpBackup from Employee

Insert into Select

Insert into select is used to insert data into existing table. Here both tables must be exists and should have same schema/structure.

1. Insert into EmployeeBckup select \* from Employee
2. Insert into EmployeeBackup( Id, Name, Gender) select (Id, Name, Gender) from Employee

**Note:** here both tables already exist. If any of the table is non-existing then error.

Group By:

Group by is used to group a selected set of rows into a set of summary rows by the values of one/more columns or expressions.

It is generally, always used in conjugation with one/more aggregate functions.

Select City, sum(Salary) as TotalSalary from TblPerson group by City

If we omit ‘group by’ then we will get error

Select sum(salary) from TblPerson

Grouping by multiple Columns

Select city, gender, sum(salary) as TotalSalary from TblPerson

Group by City, Gender

Order by City, Gender

Using Multiple Aggregate Function

Select city, gender, sum(salary) as TotalSalary, Count(\*) as TotalEmp from TblPerson

Group by City, Gender

Filtering Groups

1. First Filter records then apply group by on filtered data.

Select Gender, City, Sum(Salary) as TotalSalary, Count(Id) as TotalEmp from TblPerson

Where Gender=’Male’

Group by Gender, City

Here from table only ‘Male’ records retrieved and then grouped i.e., aggregation done only filtered group

1. First group by data and then apply filter to grouped data

Select Gender, City, Sum(Salary) as TotalSalary, Count(Id) as TotalEmp from TblPerson

Group by Gender, City

Having Gender= ‘Male’

Here table first grouped by gender and then only male group is filtered from grouped table i.e., aggregation is done for every row of table.

Try to eliminate rows that we not need as early as possible to increase performance.

In above queries, performance wise (1) is good.

Where VS Having

1. Where filter the rows before aggregations (grouping)

Having filter groups i.e. after aggregations are performed.

1. Where clause can be used with select, insert and delete

Having clause can only used with select

1. Aggregate functions can not be used in where clause unless it is in a sub-query contained in having clause but

Aggregate functions can be used in having clause (i.e., where clause can’t be used with aggregate functions but having can be used with aggregate functions)

Where is used to filter individual rows but having is used to filter groups.

1. Where comes before Group By means – It filter rows before aggregation calculations are performed.

Having clause comes after Group By means – It filter rows after aggregation calculation are performed.

So, performance wise, having is slower than where.

Examples:

1. Calculate total sales by product and give the name of the product where total sales > 1000

Select Prod, sum(salesAmount) as TotalSales from TblProduct

Group by Prod having sum(salesAmount) > 1000

1. Select product, sum(salesAlmount) as TotalSales from TblProduct

where Prod in (‘iPhone’,’Tablet’)

Group by prod

Here where clause filter rows before aggregations (i.e., first filter rows then aggregations on filtered row).

1. Select product, sum(salesAmount) as TotalSales from TblProduct

Group by Product

Having product in (‘iPhone’,’Tablets’)

Here having clause filter the rows after aggregation (i.e., first aggregation and then filter)