Persistent

Nugget 4: The SELECT Statement

Persistent University

Key Learning Points

1. Basic SELECT statement
2. Using Arithmetic operators
3. Using Concatenation operator
4. Filtering data using WHERE clause
5. Various Comparison operators
6. Logical conditions
7. Sorting data using ORDER BY clause

Syntax of Basic Select statement

- Syntax

  Select c1,c2..cn

  from t1,t2..tn

  - SELECT clause identifies: which column/s to display.
  - FROM clause identifies: from which table/s to select.

Capabilities of SQL Select Statement

- **Projection** : User can use projection capability **to choose the columns** that user want to return by the query. User can use few or many columns as required.

- **Selection**: User can use Selection capability **to choose the rows** that user want to return by the query. User can use various ways to restrict the rows.

- **Joining**: User can use Join capability **to bring together data** that is stored **in different tables** by creating the link between them.

Guidelines for Writing SQL Statements

- SQL Statements are **not case sensitive**
- SQL Statements can **be on one or more lines**
- Keywords **cannot be abbreviated** or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- Keywords are typically entered in Upper case; all other words such as table name, column name are entered in lower case.

Sample Data

EmployeeId	FirstName	LastName	Email	PhoneNumber	HireDate	JobId	Salary	CommitionPct	ManagerId	DepartmentId
1	John	Demn	JohnD@yahoo.com	9898780979	1/10/2001	IT_PROF	70000	0.5	NULL	10
2	Ken	Dale	kendaleD@gmail.com	7877787655	4/1/2001	SALES_HEAD	50000	NULL	NULL	10
3	James	Walton	JW@yahoo.com	5787887888	1/1/2001	IT_REP	30000	0.2	1	20
4	robin	sngal	robin@gmail.com	4990988839	5/1/2001	SALES_REP	40000	0.3	2	20
5	ajay	ghosala	ghosala@hotmail.com	9809888898	6/10/2002	SALES_REP	30000	0.4	2	20

Basic SELECT statement

Here, we will discuss with the help of examples.

- Select *

  From Department;

  -- will retrieve all column data

  -- from Department table

- Select dept_id, dept_name

  From Department;

  -- will retrieve only 2 columns data

  -- from Department table

- Select job_id from employees;

  -- Default display of query is all job ids; including duplicate rows.
  We can eliminate duplicate rows by using DISTINCT keyword

- Select distinct job_id from employees;

  -- It will retrieve only unique job ids from employee table

Arithmetic Operators and precedence

- Arithmetic Operators are used to perform calculations or to modify the way in which data is displayed.
- Arithmetic expression can contain column names, constants and arithmetic operators.
- Arithmetic Operators can be used in any clause of SELECT statement except FROM clause.

Arithmetic Operator	Meaning	Precedence
*	Multiply	1
/	Divide	1
+	Add	2
-	Subtract	2

Arithmetic Operators and precedence

- Arithmetic Operators can be used with only NUMBER and DATE datatype.
- Any arithmetic operation involving NULL value return NULL Value.
- If any expression contains 2 arithmetic operators of same precedence, then such expression will be evaluated from left to right.

- Display salary of employees as salary increased by 100.

```
SELECT employee_id, firstname, salary + 100
FROM Employee;
```

EmployeeId	FirstName	Salary+100
1	John	70100
2	Ken	50100
3	James	30100
4	robin	40100
5	ajay	30100

Arithmetic Operators and precedence

- Display salary of employees as annual salary increased by 100.

```
SELECT employee_id,firstname, salary * 12 + 100 FROM Employee;

-- Here multiplication takes precedence over addition; thus salary will be multiplied by 12 first and then 100 will be added to it.
```

- Display commission for all the employees.

```
SELECT employeeid, firstname, salary, CommitionPct, (12*salary)*CommitionPct  FROM Employee;

-- Here, as the commition % for employee Ken is NULL so output of  above calculation is also NULL.
```

- Points to remember:
  - A NULL is a value which is unknown, undefined, unavailable or inapplicable. It is not same as Zero or Blank or Space.
  - Any Arithmetic operation involving NULL values will return NULL.

Column Alias

- By default, Column headings are name of the selected columns and they are displayed in uppercase. The name of the column may not be descriptive or in case of calculated fields, heading might be very difficult to understand. In such case, column aliases can be used to rename column heading.
- Column Alias:
  - Renames a column heading
  - Is useful with calculations
  - Immediately follows the column name. There can be optional AS keyword between column name and alias
  - Required double quotation marks if it contains space or special characters or is case sensitive.
- Example:

```
SELECT firstname "First Name", salary sal ,
CommitionPct AS comm
FROM Employee
```

First Name	SAL	COMM
John	70000	0.5
Ken	50000	NULL
James	30000	0.2
robin	40000	0.3
ajay	30000	0.4

Concatenation Operator

- A Concatenation Operator
  - Concatenates column or character string to other column.
  - Is represented by two vertical bars(||)
  - Creates a resultant column that is character expression.
- For example:

```
Select FirstName ||' with job type: ' || Jobid  "Employee details"
from Employee;
```

Employee details
John with job type: IT_PROF
Ken with job type: SALES_HEAD
James with job type: IT_REP
robin with job type: SALES_REP
ajay with job type: SALES_REP

```
Select FirstName ||' 1 Month salary: ' || salary   “salary details”
from Employee;
```

salary details
John 1 Month salary: 70000
Ken 1 Month salary: 50000
James 1 Month salary: 30000
robin 1 Month salary: 40000
ajay 1 Month salary: 30000

Note: ‘ With job type: ‘ is a literal character string. Literal is hard coded character or date or number included in the select list. Date and character literals are included in single quotation marks.

Filtering data using WHERE clause

- User can restrict the rows returned from the query by using WHERE clause.
- A WHERE clause contains a condition that must be met. If the condition is true, the row meeting the condition is returned.
- A WHERE clause directly follows FROM clause
- WHERE clause can compare values in columns, literal values, arithmetic expressions, or functions. It consists of 3 elements
  - Column name
  - Comparison condition
  - Column name, constant or list of values.
- Syntax: SELECT *|column|expression [alias],…

  FROM table

  WHERE condition[s];

Filtering data using WHERE clause

- Examples:
  - Select employeeid, firstname, salary, hiredate, Jobid, departmentid

    From Employee

    Where Departmentid=10;

EMPLOYEEID	FIRSTNAME	SALARY	HIREDATE	JOBID	DEPARTMENTID
1	John	70000	2001-01-10	IT_PROF	10
2	Ken	50000	2001-04-01	SALES_HEAD	10

  - Select employeeid,firstname,salary,hiredate,Jobid, departmentid

    From Employee

    Where JobId='SALES_REP' ;

EMPLOYEEID	FIRSTNAME	SALARY	HIREDATE	JOBID	DEPARTMENTID
4	robin	40000	2001-05-01	SALES_REP	20
5	ajay	30000	2002-06-10	SALES_REP	20

Note: Character and Date values are enclosed in single quotation marks.

Character values are case sensitive. Date values are format sensitive.

Default date format is DD-MON-YY.

Comparison Conditions

- Comparison operators are used in conditions that compare one expression to another value or expression. They are used in WHERE clause in following format

  WHERE Expr Operator Value

- Various comparison operators along with its meaning. =

=	Equal To	>=	Greater than or equal to
>	Greater than	<=	Less than or equal to
<	Less than	<>	Not equal to

- Examples:

  .. WHERE hire_date = ‘01-JAN-1999’

  .. WHERE salary >= 6000

  .. WHERE last_name = ‘Smith’

  etc.

Other comparison conditions

- Below are some more Comparison conditions
  - BETWEEN … AND ...
  - IN (Set)
  - LIKE
  - IS NULL

- Using BETWEEN Condition:
  - Between condition is used to display rows based on range of values.
  - Range specified contains the lower limit and upper limit. Lower limit should be specified first.
  - Values specified in the BETWEEN clause are inclusive.
  - Example:

```
Select lastName, salary From Employee

Where salary between 30000 and 60000;
```

Other comparison conditions

- Using IN Condition:
  - IN Condition is used to test a value in a specified list of values.
  - It is also called as membership condition.
  - IN condition can be used with any data type.
  - If character or date values are used in the list, they must be enclosed in single quotation marks.
  - Example:

```
SELECT lastName,salary,ManagerId,JobId
FROM Employee
WHERE JobId IN (‘SALES_REP’,’IT_REP’);
```

Other comparison conditions

- Using LIKE Condition:
  - LIKE condition is used to perform wildcard search.
  - It is used to select rows that match a character pattern.
  - Search string can contain either literal character or number.
  - Two symbols can be used to construct the search string.
    - % : represents any sequence of 0 or more characters.
    - _ : represents any single character.
  - Example :
    - ..Where FirstName like ‘S%’ --where firstname starts with capital S.
    - ..Where FirstName like ‘_a%’ -- where second char of name contains a.
    - ..Where HireDate like ‘%1995’ -- where hiredate belongs to 1995

Other comparison conditions

- Using IS NULL Condition:
  - IS NULL is used to test, whether particular column contains NULL value.
  - A NULL is a value which is unavailable, unassigned, unknown or inapplicable. So, user can not test with ‘=‘ . NULL can not be equal or unequal to any value. Thus, IS NULL condition is used to test for NULLs.
  - Example:

```
SELECT lastName, HireDate, salary, CommitionPct
FROM Employee
WHERE commitionPct is NULL;
```

- o

EMPLOYEEID	LASTNAME	JOBID	SALARY	COMMITIONPCT
2	Dale	SALES_HEAD	50000	

Logical Conditions

- A Logical condition combines the result of two component conditions to produce a single result based on them or inverts the result of single condition. A Row is returned only if the overall result of the condition is TRUE. Following three logical conditions are available in SQL
  - AND : Returns TRUE if both component conditions are true.
  - OR : Returns TRUE if either component condition is true.
  - NOT : Returns TRUE if condition is false.

  Till now, we have seen only one condition in WHERE clause. Using AND and OR operators, we can use multiple conditions in WHERE clause.

Logical Conditions

- AND Truth Table: The table shows results of combining two condition with AND operator.

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

- OR Truth Table: The table shows results of combining two condition with OR operator.

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

- NOT Truth Table: The table shows result of applying NOT operator to a condition.

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Logical Conditions

- Examples:
  - AND Operator

```
Select Employeeid, LastName, JobId, Salary
  from Employee
  where Salary>=40000
  AND LastName like ‘D%’
```

Output:

EMPLOYEEID	LASTNAME	JOBID	SALARY
1	Demn	IT_PROF	70000
2	Dale	SALES_HEAD	50000

  - OR Operator

```
Select Employeeid, LastName, JobId, Salary
  from Employee
  where Salary<50000
  OR JobId like '%IT%’
```

Output:

EMPLOYEEID	LASTNAME	JOBID	SALARY
1	Demn	IT_PROF	70000
3	Walton	IT_REP	30000
4	sngal	SALES_REP	40000
5	ghosala	SALES_REP	30000

Logical Conditions

- Examples:
  - NOT Operator:

    NOT operator can be used with all comparison operators as below:

    … WHERE JobId NOT IN (‘SALES_REP’,’SALES_HEAD’)

    … WHERE JobId NOT LIKE ‘%SALES%’

    … WHERE SALARY NOT BETWEEN 40000 and 50000

    … WHERE CommitionPct IS NOT NULL

Rules of precedence

- Rules of precedence determine the order in which expressions are evaluated and calculated. The below table lists the default order of precedence.
- User can override the default order by using parenthesis around the expression you want to calculate first.

Order evaluated	Operator
1	Arithmetic Operator
2	Concatenation Operator
3	Comparison Conditions
4	IS [NOT] NULL,LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT Logical condition
7	AND logical condition
8	OR Logical Condition

Sorting data using ORDER BY clause

- The order of rows returned by query is undefined. The ORDER BY clause can be used to sort the rows.
- ORDER BY clause comes last in the SQL statement.
- User can sort by column/expression or column position or column alias.
- Using ORDER by clause, data can be sorted in ascending or descending order.
- User can sort the data based on multiple columns.
- User can sort rows based on column that is not present in SELECT list.
- If ORDER BY column contains NULL values, they will appear last in case of ascending and first in case of descending order.

Sorting data using ORDER BY clause

- Syntax:

```
SELECT expr
FROM table
WHERE condition
ORDER BY {column/expression} [ASC|DESC];
```

- In the syntax
  - ORDER BY: specifies the order in which the rows are sorted.
  - ASC: Orders the rows in ascending order. (This is default order)
  - DESC: Orders the rows in descending order.

Sorting data using ORDER BY clause

- Examples:

```
SELECT EmployeeId,FirstName,JobId,Salary
FROM Employee
order by salary;  -- Default is in ascending order
```

EMPLOYEEID	FIRSTNAME	JOBID	SALARY
3	James	IT_REP	30000
5	ajay	SALES_REP	30000
4	robin	SALES_REP	40000
2	Ken	SALES_HEAD	50000
1	John	IT_PROF	70000

```
SELECT EmployeeId,FirstName,JobId,Salary*12 ANNSAL
FROM Employee
order by ANNSAL;  -- User can sort based on column alias.
```

EMPLOYEEID	FIRSTNAME	JOBID	ANNSAL
3	James	IT_REP	360000
5	ajay	SALES_REP	360000
4	robin	SALES_REP	480000
2	Ken	SALES_HEAD	600000
1	John	IT_PROF	840000

```
SELECT EmployeeId,FirstName,JobId,salary
FROM Employee
order by 4;       --It will sort data based on 4th column in the select list.
```

Sorting data using ORDER BY clause

- Examples:

```
SELECT EmployeeId, FirstName, JobId, salary
FROM Employee
order by salary desc;
-- It will sort the data, based salary in descending order.
```

EMPLOYEEID	FIRSTNAME	JOBID	SALARY
1	John	IT_PROF	70000
2	Ken	SALES_HEAD	50000
4	robin	SALES_REP	40000
5	ajay	SALES_REP	30000
3	James	IT_REP	30000

```
SELECT EmployeeId, FirstName, JobId, salary
FROM Employee
order by  jobid, salary desc;
/* It will sort the data based on two column;
based on Jobid in ascending order
and based on salary in descending order. */
```

EMPLOYEEID	FIRSTNAME	JOBID	SALARY
1	John	IT_PROF	70000
3	James	IT_REP	30000
2	Ken	SALES_HEAD	50000
4	robin	SALES_REP	40000
5	ajay	SALES_REP	30000

Reference Material: Sites

http://www.oracle-dba-online.com/sql/oracle_sql_select_statement.htm

http://beginner-sql-tutorial.com/sql-select-statement.htm

http://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_10002.htm

http://www.w3schools.com/sql/sql_orderby.asp

http://www.dba-oracle.com/concepts/sql_query_basics.htm

Session 4: Summary

With this we have come to an end of our fourth session where we discussed Basic SELECT statement, Guidelines for writing SELECT statements, Various operators, Use of WHERE clause and Order By clause.

- At the end of Nugget 4, we see that you are now able to answer following questions:
  - Explain “Basic Syntax and Usage of SELECT statement”.
  - Explain “Use of various comparison Operators in WHERE clause”
  - Explain “Use of various Logical conditions”
  - Explain “ORDER BY clause in detail”

Persistent

Thank you!

Persistent University