OPERATORS

OPERATORS

- Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

Types of Operators:

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators used to negate conditions

- **WHERE** clause is used to perform operation(s), such as comparisons and arithmetic operations in SQL statement.

SAMPLE RELATION: STATIONERY

S_ID	S_Name	S_Price	S_Quantity	S_Tax	Supplier
AS154	Pencil	10	50	42.56	Robin
DW215	Pen	15	120		Jessy
DE589	Eraser	5	45	54.38	Robin
RT987	Ink Eraser	7	65		Stephen
MT987	Scale	20	70	38.64	Jessy

ARITHMETIC OPERATORS

Description	Operator	Precedence
Addition	+	Lowest
Subtraction	-	Lowest
Multiplication	*	Medium
Division	/	Medium
Brackets	( )	Highest

EXAMPLES

- Increment the price of all stationery items with Rs. 10.

```
SQL> select S_PRICE+10 from Stationery;

S_PRICE+10
----------
        20
        25
        15
        17
        30
```

- Decrement the quantity of all stationery items with 6.

```
SQL> select S_Quantity-10 from Stationery;

S_QUANTITY-10
-------------
           40
          110
           35
           55
           60
```

EXAMPLES

- Display 3 times of price for all items.

```
SQL> select S_Price*3 from Stationery;

 S_PRICE*3
----------
        30
        45
        15
        21
        60
```

- Display the Quantity divided by 2.

```
SQL> select S_quantity/2 from Stationery;

S_QUANTITY/2
------------
          25
          60
        22.5
        32.5
          35
```

EXAMPLES

- Display the Price added with 15 and quantity divided by 3.

```
SQL> select S_Price+15,S_Quantity/3 from Stationery;

S_PRICE+15 S_QUANTITY/3
---------- ------------
        25   16.6666667
        30           40
        20           15
        22   21.6666667
        35   23.3333333
```

- Display the total cost of all products.

```
SQL> select S_Price*S_Quantity+S_Tax from Stationery;

S_PRICE*S_QUANTITY+S_TAX
------------------------
                  542.56

                  279.38

                 1438.64
```

ALIASES

- ALIASES can be used to create a temporary name for columns or tables.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of the query

Syntax

```
column_name AS alias_name
```

Note

If the *alias_name* contains spaces, you must enclose the *alias_name* in **double** quotes

EXAMPLES

WITH AS KEYWORD:

```
SQL> select S_Price+10 AS New_Price from Stationery;

 NEW_PRICE
----------
        20
        25
        15
        17
        30
```

```
SQL> select S_Price+10 AS "New_Price" from Stationery;

 New_Price
----------
        20
        25
        15
        17
        30
```

```
SQL> select S_Price+10 AS "New Price" from Stationery;

 New Price
----------
        20
        25
        15
        17
        30
```

EXAMPLES

WITHOUT AS KEYWORD:

```
SQL> select S_Price+10 "New_Price" from Stationery;

 New_Price
----------
        20
        25
        15
        17
        30
```

```
SQL> select S_Price+10 New_Price from Stationery;

 NEW_PRICE
----------
        20
        25
        15
        17
        30
```

```
SQL> select S_Price+10 "New Price" from Stationery;

 New Price
----------
        20
        25
        15
        17
        30
```

DUAL

- The DUAL is special one row, one column table present by default in all Oracle databases.
- The owner of DUAL is SYS (SYS owns the data dictionary, therefore DUAL is part of the data dictionary) but DUAL can be accessed by every user.
- The table has a single VARCHAR2(1) column called DUMMY that has a value of 'X'.

DUAL

Structure of Dual

```
SQL> DESC DUAL;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------

 DUMMY                                              VARCHAR2(1)
```

Values in Dual

```
SQL> select *from DUAL;

D
-
X
```

EXAMPLES

```
SQL> select 15+12-5*4/2 from dual;

15+12-5*4/2
-----------
         17
```

```
SQL> select 15+(12-5)*4/2 AS Result from dual;

    RESULT
----------
        29
```

```
SQL> select 24*7+23-2 Result from Dual;

    RESULT
----------
       189
```

EXAMPLES

```
SQL> select 5>7 from Dual;
select 5>7 from Dual
         *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

```
SQL> select 1 AND 0 as Result from Dual;
select 1 AND 0 as Result from Dual
           *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

Relational and Logical Expressions cannot be evaluated. It throws error.

EXAMPLES

```
SQL> select ‘Good Morning’ from dual;

’GOODMORNING
------------
Good Morning
```

```
SQL> select ’Good Morning’ AS Welcome from dual;

WELCOME
-----------
Good Morning
```

```
SQL> select "good morning" AS Welcome from dual;
select "good morning" AS Welcome from dual
       *
ERROR at line 1:
ORA-00904: "good morning": invalid identifier
```

Alias name can be given in double quotes but not the input string

EXAMPLES

```
SQL> select SYSDATE from DUAL;

SYSDATE
---------
01-AUG-18
```

```
SQL> select USER from DUAL;

USER
------------------------------
SYSTEM
```

COMPARISON OPERATORS

= Equality Operator

used to test for equality in a query. The = operator can only test equality with values that are not NULL.

Example:

Get the details from Stationery relation whose supplier name is ‘Robin’.

```
SQL> select *from Stationery where Supplier='Robin';

S_ID   S_NAME            S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
```

COMPARISON OPERATORS

! = or <> Inequality Operator

It checks for inequality

Example:

Get the details from Stationery relation whose supplier name is not ‘Robin’.

```
SQL> select *from Stationery where Supplier!=’Robin’;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                    15        120            Jessy
RT987  Ink Eraser              7         65            Stephen
MT987  Scale                  20         70      38.64 Jessy

SQL> select *from Stationery where Supplier <> ’Robin’;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                    15        120            Jessy
RT987  Ink Eraser              7         65            Stephen
MT987  Scale                  20         70      38.64 Jessy
```

COMPARISON OPERATORS

> Greater than Operator

Example:

Get the details from Stationery relation whose Price is greater than 10.

```
SQL> select *from Stationery where S_Price>10;

S_ID   S_NAME           S_PRICE S_QUANTITY     S_TAX SUPPLIER
------ -------------- ---------- ---------- ---------- ----------
DW215  Pen                    15        120            Jessy
MT987  Scale                  20         70      38.64 Jessy
```

COMPARISON OPERATORS

< Less than Operator

Example

Get the details from Stationery relation whose Price is less than 10.

```
SQL> select *from Stationery where S_Price<10;

S_ID   S_NAME            S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- --------- ---------- ---------- ----------
DE589  Eraser                  5         45      54.38 Robin
RT987  Ink Eraser              7         65            Stephen
```

COMPARISON OPERATORS

>= Greater Than or Equal To Operator

Example

Get the details from Stationery relation whose quantity is greater than or equal to 50.

S_ID	S_NAME	S_PRICE	S_QUANTITY	S_TAX	SUPPLIER
AS154	Pencil	10	50	42.56	Robin
DW215	Pen	15	120		Jessy
RT987	Ink Eraser	7	65		Stephen
MT987	Scale	20	70	38.64	Jessy

COMPARISON OPERATORS

<= Less Than or Equal To Operator

Example

Get the details from Stationery relation whose quantity is less than or equal to 90.

```
SQL> select *from Stationery where S_Quantity <= 90;

S_ID   S_NAME            S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
RT987  Ink Eraser               7         65            Stephen
MT987  Scale                   20         70      38.64 Jessy
```

LOGICAL OPERATORS

AND OPERATOR

Should satisfy both the conditions.

Example:

Display the details of Stationery whose S_ID is ‘MT987’ and Supplier is ‘Jessy’.

```
SQL> select *from Stationery where S_ID='MT987' AND Supplier='Jessy';

S_ID   S_NAME            S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
MT987  Scale                   20         70      38.64 Jessy
```

LOGICAL OPERATORS

OR OPERATOR

Satisfy any one condition.

Examples:

Display the details of Stationery whose S_ID is either ‘MT987’ or ‘RT987’.

S_ID	S_NAME	S_PRICE	S_QUANTITY	S_TAX	SUPPLIER
RT987	Ink Eraser	7	65		Stephen
MT987	Scale	20	70	38.64	Jessy

LOGICAL OPERATORS

Display the details of Stationery whose S_ID is ‘MT987’ and ‘RT987’.

S_ID	S_NAME	S_PRICE	S_QUANTITY	S_TAX	SUPPLIER
RT987	Ink Eraser	7	65		Stephen
MT987	Scale	20	70	38.64	Jessy

Display the details of Stationery whose Name is ‘Eraser’ or Quantity is 65.

```
SQL> select *from Stationery where S_Name='Eraser' OR S_Quantity=65;
```

S_ID	S_NAME	S_PRICE	S_QUANTITY	S_TAX	SUPPLIER
DE589	Eraser	5	45	54.38	Robin
RT987	Ink Eraser	7	65		Stephen

LOGICAL OPERATORS

BETWEEN OPERATOR

Used to retrieve/get values from the table within a specific range.

Example:

Display the details of Stationery whose price ranges from 5 to 10.

```
SQL> select *from Stationery where S_Price Between 5 AND 10;

S_ID   S_NAME            S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
RT987  Ink Eraser               7         65            Stephen
```

LOGICAL OPERATORS

IN OPERATOR

It is used to help reduce the need to use multiple OR conditions.

Example:

Display the details of the Stationery whose S_ID are AS154, MT987 and DE589.

```
SQL> select *from Stationery where S_ID IN ('AS154','MT987','DE589');

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
MT987  Scale                   20         70      38.64 Jessy
```

LOGICAL OPERATORS

LIKE OPERATOR

Used for pattern matching. It checks for the matching pattern using wild card operators.

% (percentage) – matches for zero or more characters.

_ (Underscore) – matches for a single character.

Examples:

Display the details of the stationery whose name begins with ‘P’.

```
SQL> select *from Stationery where S_Name LIKE 'P%';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DW215  Pen                     15        120            Jessy
```

LOGICAL OPERATORS

Display the details of the stationery whose name begins with ‘Pen’.

```
SQL> select *from Stationery where S_Name LIKE 'Pen%';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- --------- ---------- ---------- ----------
AS154  Pencil                 10         50      42.56 Robin
DW215  Pen                    15        120            Jessy
```

Display the details of the stationery whose name ends with ‘r’.

```
SQL> select *from Stationery where S_Name LIKE '%r';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- --------- ---------- ---------- ----------
DE589  Eraser                 5         45      54.38 Robin
RT987  Ink Eraser             7         65            Stephen
```

LOGICAL OPERATORS

Display the details of the stationery whose name ends with ‘ser’.

```
SQL> select *from Stationery where S_Name LIKE '%ser';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ -------------- --------- ---------- ---------- ----------
DE589  Eraser                 5         45      54.38 Robin
RT987  Ink Eraser             7         65            Stephen
```

Display the details of the stationery whose name contains the letter ‘e’.

```
SQL> select *from Stationery where S_Name LIKE '%e%';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ -------------- --------- ---------- ---------- ----------
AS154  Pencil                10         50      42.56 Robin
DW215  Pen                   15        120            Jessy
DE589  Eraser                 5         45      54.38 Robin
RT987  Ink Eraser             7         65            Stephen
MT987  Scale                 20         70      38.64 Jessy
```

LOGICAL OPERATORS

Display the details of the stationery whose name exactly contains 6 characters.

```
SQL> select *from Stationery where S_Name LIKE '______';

S_ID   S_NAME          S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
```

Display the details of the stationery whose name has ‘E’ as first character and ‘a’ as third character.

```
SQL> select *from Stationery where S_Name LIKE 'E_a%';

S_ID   S_NAME          S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DE589  Eraser                   5         45      54.38 Robin
```

LOGICAL OPERATORS

IS NULL Operator

The IS NULL operator is used to display all the rows for columns that do not have a value.

Example:

Display the details of the stationery for which the tax is not mentioned.

```
SQL> select *from Stationery where S_Tax IS NULL;

S_ID   S_NAME              S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                     15        120            Jessy
RT987  Ink Eraser               7         65            Stephen
```

NOT OPERATOR

It is used to negate conditions.

Example:

Display the details of the stationery whose name is not ‘Pen’.

```
SQL> select *from Stationery where NOT S_Name = 'Pen';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
RT987  Ink Eraser               7         65            Stephen
MT987  Scale                   20         70      38.64 Jessy
```

NOT OPERATOR

Not Operator with BETWEEN:

Display the details of the stationery whose quantity does not ranges from 50 and 60.

```
SQL> select *from Stationery where S_Quantity NOT BETWEEN 50 AND 60;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                     15        120            Jessy
DE589  Eraser                   5         45      54.38 Robin
RT987  Ink Eraser               7         65            Stephen
MT987  Scale                   20         70      38.64 Jessy

SQL> select *from Stationery where NOT S_Quantity BETWEEN 50 AND 60;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                     15        120            Jessy
DE589  Eraser                   5         45      54.38 Robin
RT987  Ink Eraser               7         65            Stephen
MT987  Scale                   20         70      38.64 Jessy
```

(**NOT** can be given near **column name** or near **BETWEEN** operator)

NOT OPERATOR

Not Operator with IN:

Display the details of the Stationery whose S_ID are not AS154, MT987 and DE589.

```
SQL> select *from Stationery where S_ID NOT IN ('AS154','MT987','DE589');

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                     15        120            Jessy
RT987  Ink Eraser               7         65            Stephen

SQL> select *from Stationery where NOT S_ID IN ('AS154','MT987','DE589');

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DW215  Pen                     15        120            Jessy
RT987  Ink Eraser               7         65            Stephen
```

(**NOT** can be given near **column name** or near **IN** operator)

NOT OPERATOR

Not Operator with LIKE:

Display the details of the stationery whose name does not starts with ‘P’.

```
SQL> select *from Stationery where S_Name NOT LIKE 'P%';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DE589  Eraser                  5         45      54.38 Robin
RT987  Ink Eraser              7         65            Stephen
MT987  Scale                  20         70      38.64 Jessy

SQL> select *from Stationery where NOT S_Name LIKE 'P%';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
DE589  Eraser                  5         45      54.38 Robin
RT987  Ink Eraser              7         65            Stephen
MT987  Scale                  20         70      38.64 Jessy
```

(**NOT** can be given near **column name** or near **LIKE** operator)

NOT OPERATOR

Not Operator with LIKE:

Display the details of the stationery whose name does not contain 5 letters.

```
SQL> select *from Stationery where NOT S_Name LIKE '_____';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ -------------- --------- ---------- ---------- ----------
AS154  Pencil                10         50      42.56 Robin
DW215  Pen                   15        120            Jessy
DE589  Eraser                 5         45      54.38 Robin
RT987  Ink Eraser             7         65            Stephen

SQL> select *from Stationery where S_Name NOT LIKE '_____';

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ -------------- --------- ---------- ---------- ----------
AS154  Pencil                10         50      42.56 Robin
DW215  Pen                   15        120            Jessy
DE589  Eraser                 5         45      54.38 Robin
RT987  Ink Eraser             7         65            Stephen
```

NOT OPERATOR

Not Operator with IS NULL:

Display the details of the Stationery which has tax.

```
SQL> select *from Stationery where S_Tax IS NOT NULL;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
MT987  Scale                   20         70      38.64 Jessy

SQL> select *from Stationery where NOT S_Tax IS NULL;

S_ID   S_NAME           S_PRICE S_QUANTITY      S_TAX SUPPLIER
------ --------------- ---------- ---------- ---------- ----------
AS154  Pencil                  10         50      42.56 Robin
DE589  Eraser                   5         45      54.38 Robin
MT987  Scale                   20         70      38.64 Jessy
```

(**NOT** can be given near **column name** or near **NULL**)

COCATENATION OPERATOR

It is used to join two string values or expressions in a SELECT query.

|| is the symbol used for concatenating or joining two strings.

Example:

Display S_ID and S_Name in a single column.

```
SQL> select S_ID||S_Name from Stationery;

S_ID||S_NAME
--------------------
AS154Pencil
DW215Pen
DE589Eraser
RT987Ink Eraser
MT987Scale
```

COCATENATION OPERATOR

Display all the rows in Stationery in the following format as “Price Details”.

“Price of Pen is Rs.15”

```
SQL> select 'Price of ' || S_Name || ' is Rs.' || S_Price AS Price_Details from
Stationery;

PRICE_DETAILS
--------------------------------------------------------------------------------
Price of Pencil is Rs.10
Price of Pen is Rs.15
Price of Eraser is Rs.5
Price of Ink Eraser is Rs.7
Price of Scale is Rs.20
```

DISTINCT

The DISTINCT clause is used in a SELECT statement to filter duplicate rows in the result set. It ensures that rows returned are unique for the column or columns specified in the SELECT clause.

In Simple,

The DISTINCT clause is used to return only distinct (different) values.

Example:

Display the distinct values in Supplier Name Column of Stationery Relation.

```
SQL> select DISTINCT(Supplier) from Stationery;

SUPPLIER
----------
Jessy
Robin
Stephen
```

DISTINCT

Example:

Display the distinct values in Supplier Name and S_ID Columns of Stationery Relation.

```
SQL> select DISTINCT S_Name,S_ID from Stationery;

S_NAME          S_ID
--------------- ------
Scale           MT987
Ink Eraser      RT987
Pen             DW215
Pencil          AS154
Eraser          DE589
```