SQL

Introduction:

SQL → Structured Query Language (or)
Sequential Query Language.

* SQL was introduced in the year 1970 by a company called IBM [International Business Management]
* Raymond Boyce is Father of SQL.
* EF Codd is Co-Father of SQL.
* SQL was previously known as SEQUEL
  [Simple English Query Executable Language]
  Bcoz, if didn't have ANSI Standard.
* After getting ANSI standard it was known as SQL.
* Oracle is the current Owner of SQL.

---

* SQL Versions

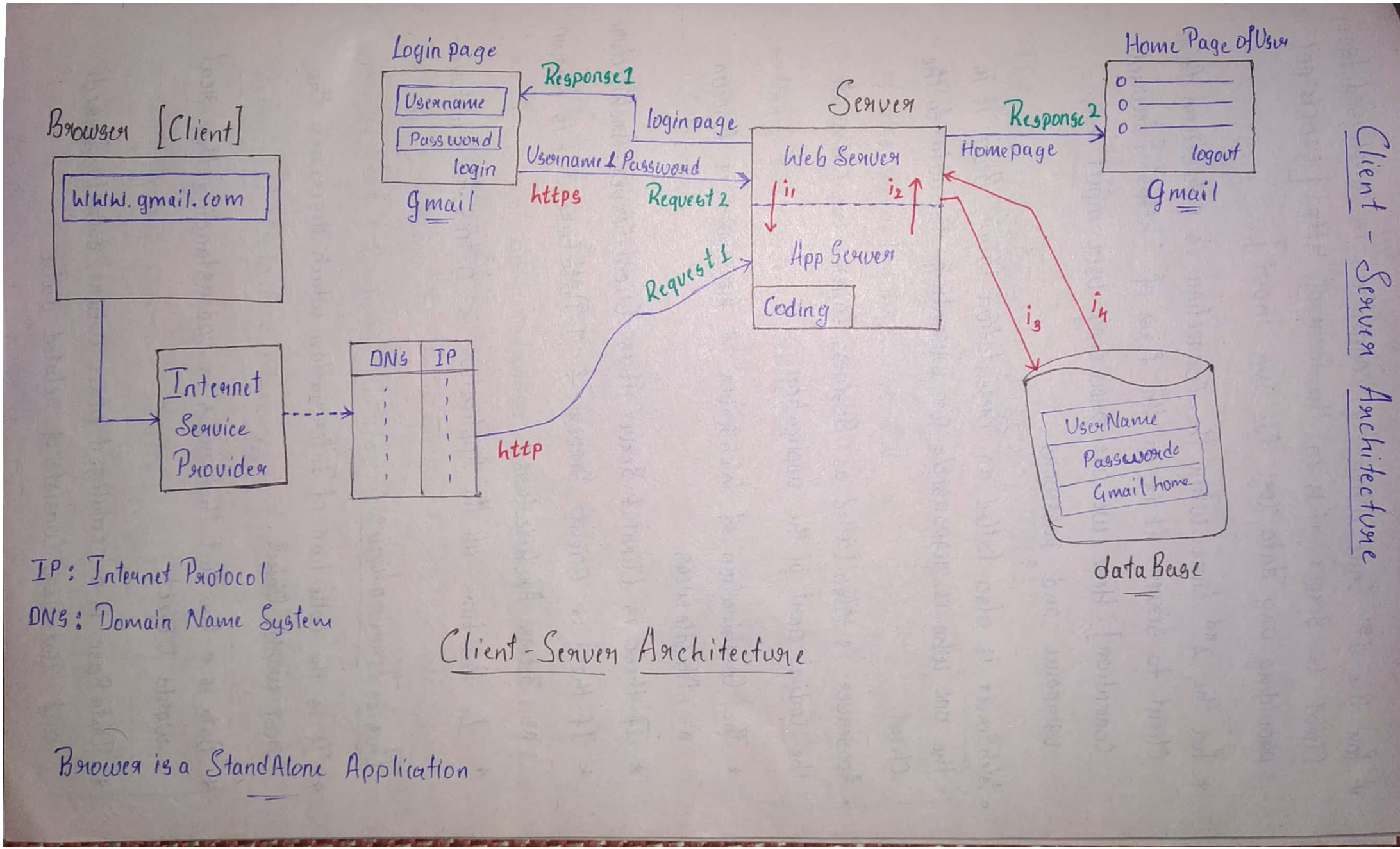
1i, 2i, ..., 9i } i → internet based

10g & 11g
g → Grid / Tables

12c, 18c, 19c } c → cloud

* 19c is the latest version of SQL.
* We use 10g in our Computer.

[StandAlone Application → No need of Internet to access it.
Web Application → needs Internet to access an Application]



* for the 1st time when the connection is established from Client to Server it is in the form of 'http' [User is not providing any data (or) No user input].

* for the 2nd time when the connection is established from Client to server it is in the form of 'https' [Secured Connection]. Here we are providing user input i.e, username and password.

- WebServer is also called as Presentation layer, Bcoz. it is the one who is responsible for presenting the data to the Client.
- AppServer is also called as Business layer, we store all the coding part of the application.

* The Combination of WebServer & AppServer is known as Middle wear.
* If their is Client & Server it is Client-Server Architecture.
* If their is Client, Server & 1 Database it is known as 3 tier Architecture.
* In Real time all the Application is n-tier Architecture.

Basic Terminologies

* It is the collection of Information which discribes the real world Object.

(or)

* Data is a Raw fact that helps in understanding the real world Object.

---

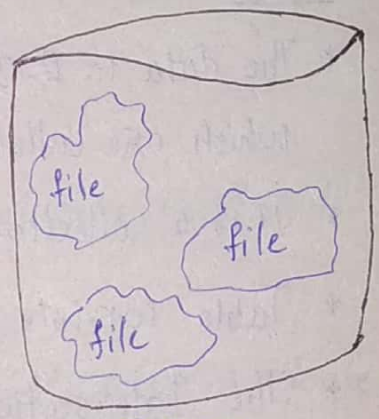
* DataBase is a centralized place where data is Stored and DataBase Consists of related data

---

* If the data is stored in database, it should allow database users to perform following actions.
  - → Add
  - → update
  - → delete
  - → access

DataBase Management System [DBMS]

* DBMS is a Software [collection of Programs] which use SQL as a Language.
* Using SQL Query is developed inorder to interact with the database.

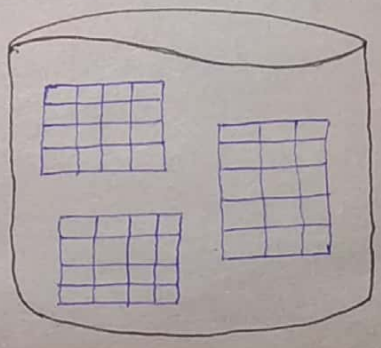


DBMS

* The data that is stored in the DBMS is in the format of files. [files with different Extensions]
* Since, we can't establish relationship between 1 file to another files.
* To Overcome the drawback of DBMS we go for RDBMS [Relational Database Management System]

Relational Database Management System [R-DBMS]

* R-DBMS is a Software which is nothing but providing relation-
* ship b/w 1 file to anothe file.



* R-DBMS uses SQL as language → using SQL Query is developed in order to interact with database.
* In R-DBMS the data is stored in the form of tables Since, the data is stored in the form of tables we can establish relationship from 1 table to another table.
* Table is also refered as entity (or) Objects.

Tables

* The data in R-DBMS is stored in database objects which are called tables.
* It is a collection of related entries.
* Table consists of Rows and Columns.
* The Intersection point of Rows and Column is known as cell.
* The data is stored inside the cell.
* Columns are also reffered as Attributes (or) fields.
* Rows are also refered as Records (or) tuples.

Note: ✱ we can establish Relationship b/w 1 table to another table by using special members known as primary key & foreign key.

Data Integrity

* Checking the correctness and Accuracy of the data in database is known as data integrity.
* It Ensures only valid data is stored into the database.

→ How ? when ? data Integrity is achived.

- Data Integrity can be achived by following the concept of R-DBMS and assigning the columns with datatypes and Constraints.

---

* The following Categories of data Integrity exist with each R-DBMS
  - Entity Integrity → There are no duplicate rows in a table.
  - Domain Integrity → Enforces valid entries for a given column by restricting the type, the format or the range of values.
  - Referential Integrity → Rows can't be deleted, which are used by other records
  - User-defined Integrity → Enforces some Specific Business rules that do not fall into entity, domain (or) Referential integrity

---

Example for R-DBMS Software

MySQL, Oracle, MongoDB, Maria DB, DB2

Datatypes

* It represents the type of data that a column can store.
* A column can be assigned with maximum of 1 datatype.
* Datatypes are mandatory for the columns
* There are n number of datatypes. but, with respect to SQL only 4 datatypes
  - > Char
  - > varchar
  - > number
  - > date

① char datatype

* If a column is assigned with char datatype it accepts following values.
* The default size of char datatype is 1.

Note: * we can restrict the limit of characters that a column can store by specifying size for the datatype.

Size ↑

char (2)

C1
1
AB
10A ✗
A.B ✗
AB

a-z
A-Z
0-9
All Special Charaters
} characters

② varchar datatype

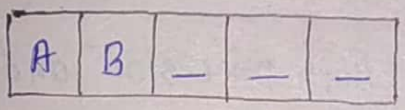
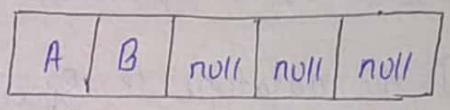
* If a column is assigned with varchar datatype it accepts following values
* varchar doesn't have default size.

Syntax: datatype (size)

varchar (3)

C2
100
AB$
0
1AB

Difference Between char and varchar

char	varchar
* Empty blocks are considered as space  char(5)	* Empty block are considered as null.  varchar(5)
* Space Consumes Memory	* null doesn't Consumes Memory.
* Default Size is 1	* No Concept of default size.
* Fixed Allocation of Memory.	* Dynamic Allocation of Memory
* Char can store upto 2000 characters	* Varchar can store upto 4000 characters.

Note: * in Oracle 9i varchar2 was introduced which can store upto 10,000 characters.

* from Oracle 10g both varchar and varchar2 can store upto 10,000 characters

③ number datatype

* If a column is assigned with number datatype it can store only +ve digits

Syntax: number (precision, scale)

Mandatory ↑ Optional ↗

Ex: (4,2) 99.99 | (4,1) 999.9 | (4,3) 9,999

(4,4) ⇒ it is not possible

number (3)

C_3
00
1A ✗
010
999
999 ✗

Only nubers (0-9)

* Number datatypes accepts 2 Arguments namely
  → Precision
  → Scale
* precision is a mandatory Argument
* Scale is a optional Argument.
* If the datatype is Specified with both precision and Scale the it can accepts Integer & decimal values

Note: * Always precision must be Greater than scale.

* default Size of number datatype is 1.
* No duplication is allowed.

④ date datatype

* If a column is assigned with date datatype it can accept following values
* It can write in 2 ways namely

DD - MON - YY

(or)

DD - MON - YYYY

date
C4
24-OCT-19
25-06-19 X
24/OCT/19 X
24-OCT-2019

Note: * Datatype should be Specified without Size [Only for date]

Primary key	Not-null	Not-null	Not-null	Not-null	Not-null	Unique Not-null	Not-null	Not-null	Check Not-null Unique
number(3)	varchar(30)	number(2)	varchar(50)	Date	char(1)	varchar(20)	number(5,2)	varchar(4)	number(10)
id	name	age	college	DOB [Date of Birth]	Gender	email_id	per	branch	phone_no
101	Nandini	19	Prerana Engineering College	09-NOV-2000	F	nandini1433@gmail.com	99.14	EEE	9004002341
102	Umadevi	18	Sai Vidya Institute of Technology	04-APR-2001	F	umadevi309@gmail.com	100.00	ECE	9441430812

Constraints

* They are the rules of validation that are assigned on the column.
* Constraints are not mandatory but highly recomended
* A column can be assigned with mulliple constraints
* We have 5 types of Constraints in SQL

① NOT null
② unique
③ check
④ primary key
⑤ Foreign key

⊛ null

* null is neither space nor Zero the empty cell in a table is called as null.
* In Oracle SQL one null is not equal to another null all the null's are unique.

$$null \neq null$$

* If we perform any Airthematic Operation with null the result is null.

$$null + 1 = null$$

A	null	B
C	D	null

① NOT null

* if a column is assigned with not null constraints it can't be empty at any point of time
* The column can't except null values

NOT null
number(3)

C_1	
100	
900	
null	X
AB	X

② unique

* If a column is assigned with unique constraints then it can't accept duplicate values

Unique number (3)

C_2
100
200
null
null

③ Check

* Check are the extra rules of validation that are given by Customer based on the requirements.
* They are also refered as domain Constraints

Check number (2)

age	
16	✗
18	
45	

Check (age ≥ 18)

④ primary key

* It is a combination of NOT null and unique constraints.
* It is used to identify the record uniquely
* A table can have maximum of 1 column delared as primary key
* A table which consists of primary key is known as master Table (or) parent Table.
* Primary key is not mandatory but heigh recomndable.

NOT null & unique number (3)

C_4	
100	
99	
99	✗
null	✗

Candidate Key's

* The Column that are assigned with not null & unique constraints are eligible to become primary key.

Alternate Key's

* The Column that are eligible to become primary key but not declared is known as Alternate Key's

How to choose primary key in Column?

→ The Column that is assigned with not null & unique constraints & assigned with numbers datatype which is lesser in size is selected as primary key.

SQL Commands [in next page]

⑧ **ed** → ed command is used to edit the previously executed query.

```
>> ed
```

---

⑤ **foreign key**

* It is used to establish relationship from 1 table to another.
* The Column that is declared as foreign key can accept null values and duplicate values.
* The table which consists of foreign key is known as child table.
* A table can have more than 1 foreign key.
* It is also Refered as referential integrity constraints.

Note: * In order to declare a column as foreign key the same column should be declared as primary key in another table.

Example :

Primary Key

Student_id	Student_name	branch	college
11	Ajith	ECE	AIT
12	Dhanush	EEE	SVIT
13	Suraj	CS	RNSIT
14	Abhi	MCA	BMSIT

Primary Key

Foreign Key

Event_id	event_name	event_sponser	Student_id
101	Sunburn	KF	12
102	Tommorowland	DSP	11
103	Hampi Utsav	SRK	14
104	Flag host	BSY	13

SQL Commands

* Set : it is used to set the no of charactors or lines or no of lines per page.

Ex: SQL > set lines 120 pages 120

Note: The value of line size and page size should be > 120 [Greater than 120]

* show user : It is used to display the name of current user who has logged in. >> show user

Query to display list of tables in the database

★ SQL Commands :-

⊛ Set → Set command is used to set the number of characters per line and the number of lines per page.

>> set lines 120 pages 120

⊛ connect → Connect command is used to switch from one user to another user.

>> connect

⊛ cl scr → clear command is used to clear Screen.

>> cl scr

⊛ / (forward slash) → It is used to execute previously executed queries.

>> /

⊛ save → Same command is used to store previously executed queries.

>>> save E:/sql44.txt
Created file E:/sql44.txt.

⊛ Spool → Spool command is used to store previously executed query along with the output.

>>> spool E:/sql44.txt

>>> spool off

★ SQL Sub Languages

① Data Query Language [DQL] → • Select

② Data Definition Language [DDL] →
- create
- alter
- drop
- rename
- truncate

③ Data Manipulation Language [DML] →
- insert
- update
- delete

④ Transcation Control Language [TCL] →
- commit
- rollback
- ~~rollback~~
- save point

⑤ Data control Language [DCL] →
- grant
- Revoke

---

① Data Query Language

(i) Select : it is used to read the data from the database.

* Select clauses has 3 Capablities
  - <1> projection
  - <2> selection
  - <3> joins

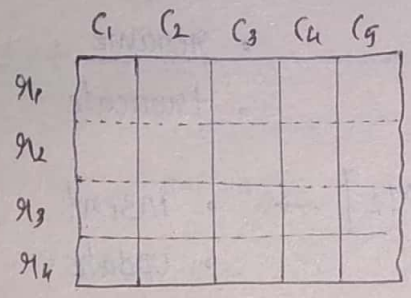
DQL → It is used to display Specified data from the database.

Select → It is ~~used~~ the statement that is used to fetch (or) other def^{n} retrerive data from the database.

<1> projection :

* Selecting the data from the required table by selecting all the column (or) required column is known as projection
* In projection we can't restrict rows default all the rows are selected.

Ex:



Select C_1 from tablename ;

$\Rightarrow C_1 : R_1, R_2, R_3, R_4$

Select C_2, C_1 from tablename ;

$\Rightarrow C_1, C_2 : R_1, R_2, R_3, R_4$

Syntax :-

② Select * / distinct column(s) / alias-name expression

① from tablename ;

Ex: Student

id	name	branch	percent
11	Anu	CS	94
12	Nandu	EEE	91
13	Uma	ECE	87

SQL > select name, branch
from student ;

name	branch
Anu	CS
Nandu	EEE
Uma	ECE

from clause

* From is a clause that is used to Specify the tablename.
* from clause loads the table into the compiler.
* 'Select' is a statement in order to select the statement from the table.

Note: * SQL Keywords are not Case Sensitive.

* from clause always executes 1st

Note: * Star (*) indicates all the Columns.

Q) Display Studentname along with percent for all Students?

```
SQL > select name, percent
      from Student;
```

Q) Display name, branch, percent for all Students?

```
SQL > select name, branch, percent
      from Student;
```

Q) Display id and percent for all Students?

```
SQL > select id, percent
      from Student;
```

Q) Display all columns for all Students?

```
SQL > select *
      from Student;
```

Syntax:

```
Select * / { distinct [Columns] } / Expression [Allias]
from table_name;
```

Literals

* A literal can be a number, character, special character (or) string (or) date data, which can be included in the select clause.
* Whenever the literals are included in the select clause the same literals will be displayed as column name and a result for each record once.
* Character, special character, string or date literals must be enclosed with single quotes.

Different types of literals

① Numeric → 75125, 3568, 12714, 33,333 etc
② Character → 'A', '%', '9', 'z', '(' → Parantheses
③ String → Hello
④ Boolean → True, False, Null etc
⑤ Date and time → '26-11-2002', '2012-10, 29'

Dual ⇒ has dummy column contains only one capacity

Ex:

```
Select 1
from dept;
```

1
1
1
1
1
1

```
SQL> Select '_'
     from dept;
```

'_'
1
2
4
7
9

```
SQL> Select 'hello'
     from dept;

'HELLO'
-------
hello
hello
hello
hello
hello
```

Operation Precedence [+ - x % /]

* multiplication and division takes priority over Addition and substraction.
* ( ) → indicates highest priority.

Ex:

```
SQL > Select 1*3+4
      from dept;

1*3+4
-----
10
10
10
```

```
SQL > Select (2+3)*3
      from dept;

(2+3)*3
-------
15
15
15
```

Aliasing

* Aliasing meaning Renameing
* In SQL using the concept of aliasing either we can rename the column name or we can rename the table-name.

Column Aliasing

* Renaming the Column name known as Column aliasing which can be done in 3 ways (or) methods.

a) Using space blw oldname and new column name

Syntax:-

② Select oldcolumnname newcolumnname
① from tablename;

Ex: Select empno ename
from emp;

b) Using 'as' keyword blw oldcolumnname and the new column name.

Syntax:-

② Select oldcolumnname as newcolumnname
① from tablename;

Ex:
Select comn as salary
from emp;

c) " " 'If the newcolumnname contains the space' blw the quotes. In this method " " preserve the case.

Syntax:-

② Select oldcolumnname "new_columnname"
① from tablename;

Ex:
Select comn "monthly comn"
from emp;

Q) Display empno, ename as employeename, job as disgnation and sal as monthly salary for all employee?

SQL> select empno, ename as employeename, job as designation, sal as monthlysalary
from emp;

Q) Display ename, empno, sal as monthlysalary and annual salary for all the employees?

SQL> select empno, ename, sal monthlysalary, sal*12 annual salary
from emp;

Q) Display ename, job, sal as monthlysalary, comm as monthly commission and annual salary along with the commission of 175?

SQL> select ename, job, sal monthlysalary, comm monthlycomm, sal*12+175 annualsalary
from emp;

Q) Display ename, job, sal as montlysalary, annualsalary with quartly comm of 220.

SQL> select ename, job, sal monthlysal, (sal*12)+(4*220) annual salary
from emp;

distinct

* distinct keyword is used to return only the unique records from the column of the table. (or)

(or)

* distinct keyword is used to eliminate the duplicate record in the column of the table.

* distinct keyword can't be used more than ones in the select clause.

Ex: select distinct deptno, distinct job from emp; } → Error

Ex: select distinct deptno
from emp;

Ex: select distinct deptno, job
from emp;

Syntax:-

```
Select distinct column[s]
from tablename;
```

Note: * Whenever distinct keyword is present in Query the changes are reflected only in the output not in table.

* If a single column is specified in the distinct keyword then it results unique values of the specified column only.

* If the multiple column's are specified in the distinct keyword then it result in the unique combination of specified columns.

Concatation Operator (||)

* It is used to concatanate Columns (or) Columns with String.
* The operator is represented by double vertical language.

Ex:
```
Select Ename||' '||job
from emp;
```

Q) Allen is a salesman

```
SQL> Select ename||' is a '||job
     from emp;
```

Q) Allen is a salesman and his salary is 800.

```
SQL> Select ename||' is a '||job||' and his salary is '
                                  ||sal
     from emp;
```

Q) Allen is a salesman and his salary is 800 and working in deptno 80.

```
SQL> Select ename||' is a '||job||' and his salary is '
                ||sal||' and working in deptno '||deptno
     from emp;
```

Operators

* They are the special symbols which are used to perform the operations on the operands.
* Operands are nothing but the data or value on which the operation will be performed.
* Operations are classified into several catagories.
  - (a) Arithmatic operators : +, -, *, /, %
  - (b) Logical Operators : AND, OR, NOT
  - (c) Relational Operators : <, >, <=, >=, =, !=
  - (d) Special operators : IN, LIKE, IS, BETWEEN

<2> Selection:

* Fetching the data from the required table by selecting all the columns or required columns along with the required no of rows.
* In selection we restrict the rows.

Syntax:

(3) Select * / { distinct [columns] } / { Expression [Alias] }
(1) from tablename
(2) where condition[s] ;

where clause

* where clause restricts the number of records to be displayed and returns the record only for specific condition.
* where clause sets the condition for each and every record in the table.
* If the condition is 'True' it selects the record.
*

* If the condition is False, it ignores the record and checks the condition for next record in the table.

Ex:
```
Select *
from emp
where deptno = 10;
```

Ex:
```
Select *
from emp
where job = 'SALESMAN';
```

Syntax:

where LHS Relational Operator RHS

Note: * RHS value is case Sensitive.

* RHS value should be enclosed within Single quote for String values.

Q) display emp details who are earning salary less than or equal to 3000?

```
SQL> Select *
     from emp
     where sal <= 3000;
```

Q) display emp details who are working as manager?

```
SQL> Select *
     from emp
     where job = 'MANAGER';
```

Q) display ename, job along with salary for the employee who is working in deptno 30?

```
SQL> Select ename, job, sal
     from emp
     where deptno = 30;
```

Q) display emp details who joined company on 3rd dec 81?

```
SQL> Select *
     from emp
     where hiredate = '03-DEC-81';
```

Note:
- join on =
- from >=
- before <
- after >
- on <=

Q) display unique salary only for the employee who are working as salesman?

```
SQL> Select distinct sal
     from emp
     where job = 'SALESMAN';
```

Q) display unique deptnumber for all the employee?

```
SQL> Select distinct deptno
     from emp;
```

Q) display unique employee details for all the employee?

```
SQL> Select distinct *
     from emp;
```

Q) display unique empname and salary for all the emp?

```
SQL> Select distinct ename, sal
     from emp;
```

Q) display unique job along with deptno only for the employee who belongs to deptno 10?

```
SQL> Select distinct job, deptno
     from emp
     where deptno = 10;
```

Q) display unique job, managerno only for the employee who is not belonging to dept no 10 ?

```
SQL> Select distinct job, mgr
     from emp
     where deptno <> 10;
```

<> (or) != Not belongs to Symbol

Q) display unique commision value for all employee ?

```
SQL> Select distinct comm
     from emp;
```

Comm
100 → null
1400
500
300
0

Note: ✱ Exceptional Case :- distinct class doesn't treat nulls as unique values

@ Logical Operators

* Logical AND → if both the condition is True it returns the record from the table.

* Logical OR → if either the condition is True it returns the record from the table.

* Logical NOT → if the condition is True it returns false.

Q) display all the employees who doesn't belongs to deptno 30 and the salary is more than 1600 ?

```
SQL> Select *
     from emp
     where deptno!=30 and sal > 1600;
```

Q) Display all the employees details whose job is Salesman and working in deptno 30 ?

```
SQL> Select *
     from emp
     where job = 'SALESMAN' AND deptno = 30 ?
```

Q) display all the emp details who are from 10 and 20 deptno?

```
SQL> Select *
     from emp
     where deptno = 10 AND deptno = 20;
```

Q) display all the emp details whoses managers numbers is not as 7698 and job is manager ?

```
SQL> Select *
     from emp
     where MGR != 7698 AND job = 'MANAGER';
```

Q) display all the emp details who joined before '02-Sep-81' and not working in deptno 20 ?

```
SQL> Select *
     from emp
     where hiredate < '02-SEP-81' AND deptno <> 20;
```

Q) display all the employees who are getting some commsion or working in deptno 20?

```
SQL> Select *
     from emp
     where comm >= 0 OR deptno = 20;
```

Q) display all the employee's whose job is manager or Salary >= 2300 or working in deptno 20?

```
SQL> Select *
     from emp
     where job='MANAGER' OR sal>=2300 OR deptno=20;
```

Q) display all the employee for working in deptno 20 or 30 and joined after '25-FEB-81'?

```
SQL> Select *
     from emp
     where (deptno = 20 OR deptno = 30) AND
                hiredate > '25-FEB-81';
```

(b) Special Operators

* IN ⟶ It is used to evaluate multiple values of same column

Ex: deptno IN 10; ✓
deptno IN 10,20,30; ✗
deptno IN (10,20,30); ✓

* Is ⟶ It is used to evaluate only the null records peresent in the columns of table.

Ex: columnname Is null.

* LIKE ⟶ It is used for pattern matching.

2 types of LIKE there are

'%' → it is used to match more than one character,
[char / number / special Character / String]

'_' → It is used to match only one character.
[char/number/special character/ ~~Blank~~]

⊛ BETWEEN → It is used to evaluate range of values b/w
lower Range Value AND Highest Range Value
↳ Mandatory

NOTE: * If the BETWEEN operator is used, then AND Logical operator is mandatory.

Order by clause

* It is used to sort the data either in ascending or descending order.
* By default, the data will be sorted in ascending order.

ASC : It sorts the data in Ascending Order.
DESC : It sorts the data in Descending order.

Syntax:

③ select * / {distinct column[s]} / {Expression alias}
① from table_name
② where condition[s]
④ order by columnname Sortingtechniqve;

* Always the orderby clause must be in the last line of the SQL Statement / Query.

Note: * We should Specify only one column in order by clause.
* By default, the data is sorted in Ascending Order.

Note: ★ null has highest priority only in descending Order. so if there any null values it will show that first and then show the other values.

Ex:

① Select comm
from emp
Order by comm DESC nulls last;

② Select comm
from emp
order by comm Asc nulls first;

③ Select comm
from emp
Order by comm DESC;

Note: ★ It is possible to pass more than one column in the order by clause to sort the data but always the first priority to sort the data will be given to first Columnname specified.

★ If the duplicate values or the same values present in the first column, then 2nd priority will be given to second Columnname specified to sort the data.

★ data can be sorted using literals in the order by clause.

Q> display unique sal for all the employee's and sort the salary in increasing order?

```
SQL> select distinct sal
     from emp
     order by sal asc;
```

Q> display unique salaries only for the employee's who belongs to deptno 30 and sort the salary in increasing?

```
SQL> select distinct sal
     from emp
     where deptno = 30
     order by sal asc;
```

Q> display employee details only for the employee's who belongs to deptno 20 and sort the ename in desending?

```
SQL> select *
     from emp
     where deptno = 20
     order by ename DESC;
```

Q> display employee details only for the employee who belongs to manager no 7698 and sort the employee name in Ascending order?

```
SQL> select *
     from emp
     where mgr = 7698
     order by ename ;
```

Note: * By default order by clause sorts the column name in ascending order if we do not specify Ascending / Descending Order.

31/10/2019

Expression

* It is a Combination of operands and operators

Ex: 10 + 20

→ 10 & 20 are operands

→ + is operator.

* Based on the operands specified in the expression it is classified into 2 types.

① Column type

② Literal type.

① Column type

* If the column name is specified as input in the expression then it is a column type Expression.

Ex: Sal + 1000

② Literal type

* If any of the direct value is specified as input in the expression then it is a Literal type expression.

* Based on the literal specified in the expression it is classified into 3 types.

⊛ number literal

⊛ charater literal

⊛ Date literal.

⊛ number

* If a number is specified as a input in the expression then it is a number literal type expression.

Ex: 100 + 100

⊛ character

* If any of the character is specified as a input in the expression then it is a character literal type expression

Ex: 'A' + 100

⊛ date

* If any of the date value is specified as input in the expression then it is a date literal type expression.

Ex: '3

Q) display employee name, salary and annual salary for all employee's ?

SQL> select ename, sal, sal * 12
from emp;

Q) display employee name, salary, 50% increment on the actual salary for all the employee's ?

SQL> select ename, sal, sal + sal * 0.5
from emp;

Q) display employee name, salary, 500 bonus on actual salary for all the employee's ?

SQL> select ename, sal, sal + 500
from emp;

Q) display employee name, salary, 3 years annual salary for all the employee's ?

SQL> select ename, sal, sal * 12 * 3
from emp;

Q) display ename, salary, quaternal salary of Annual for all the employee's?

SQL> Select ename, sal, sal * 3
from emp;

Q) display ename, salary, 25% decrement on the actual salary?

SQL> Select ename, sal, sal - sal * 0.25
from emp;

Q) display ename, salary, 75% decrement on the Anual salary?

SQL> Select ename, sal, sal * 12 (1 - 0.75)
from emp;

Aliasing

* It is a alternative name mainly given to expression, columns, tables.
* Aliasing is not mandatory. It is prefired for user convinence.

Syntax:

Select colum_name alias_name, expression alias_name

(or)

Select columnname as aliasname, expression as alias_name

* If alias name contains Special character then it should be enclosed within double quotes.
* Alias name is a user-defined name but it can't be keywords of SQL.

Q> display Employeename, salary, Annual Salary for all the employee's ?

```
SQL> select ename, sal, sal*12 Annualsal
     from emp;
```

Q> display employee name, salary, Annual salary only for the employee's who's annual salary is greater than 15000 ?

```
SQL> select ename, sal, sal*12 Annualsal
     from emp
     where 12*sal > 15000;
```

Q> display employee name, salary, 50% increment on actual salary only for the emp belongs to deptno = 30 ?

```
SQL> select ename, sal, sal + sal*0.5 as "sal50%"
     from emp
     where deptno = 30;
```

Q> display employeename, 1000 bonus on the annual salary only for the emp who are working as salesman ?

```
SQL> select ename, sal*12 + 1000 as Anualsalbonus
     from emp
     where job = 'SALESMAN';
```

Q> display employeename, sal, annualsal only for the employee who belongs to deptno 30 & sort the annualsal in desending?

```
SQL> select ename, sal, sal*12 as annualsal
     from emp
     where deptno = 30
     orderby annualsal DESC;
```

* Alias name is inactive in where clause and active in order by clause, Bcoz where clause executes before select statement and order by clause executes after select statement.

* Order of Execution

③ Select

① from

② where

④ orderby

Q) Display ename, salary, comm, 60% inc and 50% dec for all the employee?

SQL> select ename, sal, comm, sal + sal * 0.6 as "60% I"
sal - sal * 0.5 as "50% D"
from emp;

Operators

01/11/2019

// * A Query can be Specified with multiple condition by using the concept of Operators.

* Operators are the special symbols / keywords that is used to perform specific Operation.

* In SQL Operators are classified into 5 types

Arithmetic Operators : +, -, *, /, %

Rational Operators : >, <, >=, <=, =, <> [!=]

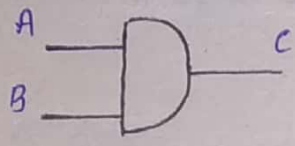
Logical Operators: AND, OR, NOT

Special Operators: IN, IS, LIKE, BETWEEN

Set Operators: UNION, UNIONALL, INTERSECT, MINUS

Logical Operators

⊛ Logical AND



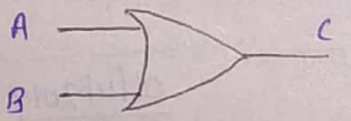
A	B	C
F	F	F
F	T	F
T	F	F
T	T	T

Syntax:

```
Select   * / Columns
from     tablename
where    Cond^n 1  AND  Cond^n 2
                   AND  Cond^n 3 ;
```

* If all the Condition are True then final result is True.
* If any one Condition fails then the final result is False.
* If all the Condition are False then final result is False.

⊛ Logical OR



A	B	C
F	F	F
F	T	T
T	F	T
T	T	T

Syntax:

```
Select   * / columns
from     tablename
where    Cond^n 1  OR  Cond^n 2
                   OR  Cond^n 3 ;
```

* If all the Condition are True then final result is True.
* If any one Condition is True then final result is True.
* If all the Condition are False then the final result is False.

Q) display employee details who are working as salesman and belongs to deptno 30?

```
SQL> select *
     from emp
     where job = 'SALESMAN' AND deptno = 30;
```

Q) display emp details who are working as manager and reporting to manager no 7839?

```
SQL> select *
     from emp
     where job = 'MANAGER' AND mgr = 7839;
```

Q) display emp details who are working as salesman and manager no 7698 and belongs to deptno 30?

```
SQL> select *
     from emp
     where job = 'SALESMAN' AND mgr = 7698 AND deptno = 30;
```

Q) display emp details who are working as Clerk or manager?

```
SQL> select *
     from emp
     where job = 'CLERK' AND job = 'MANAGER';
```

Q) display emp details empname, job, mgrno, only for the emp who joined company in the year 81?

```
SQL> select ename, job, mgr
     from emp
     where hiredate >= '01-JAN-81' AND hiredate <= '31-DEC-81';
```

Q) display employee details Those who joined company in the year 81 or 82?

```
SQL> select *
     from emp
     where (hiredate >= '01-JAN-81' AND hiredate <= '31-DEC-81')
           OR (hiredate >= '01-JAN-82' AND hiredate <= '31-DEC-82');
```

Q> display employee details who are working as clerk or manager and belongs to deptno 30?

```
SQL> Select *
     from emp
     where (job = "CLERK" OR job = "MANAGER") AND deptno = 30;
```

Note: * Logical AND take highest precedence.

Q> display employee details who are working as either clerk or manager or analyst and belongs to deptno 20?

```
SQL> Select *
     from emp
     where (job = 'CLERK' OR job = 'MANAGER' OR
            job = 'ANALYST') AND deptno = 20;
```

Special Operators

⊛ IN

* It is a multivalued operator which takes single LHS values and multiple RHS values.
* The RHS value of IN operator should be enclosed with in the paranthesis.

Syntax:

```
Select  * / Column [s]
from    tablename
where   LHS IN RHS;
```

Ex: where 20 in (10, 20, 30)

20 = 10 F
20 = 20 T
20 ≠ 30 F
} True → working as OR Operator

where 120 in (10, 20, 30)

120 = 10 F
120 = 20 F
120 = 30 F
} False

* IN Operator works similar to OR Operator.
* Generally IN Operator selects the record only if the LHS value present in RHS.
* Generally IN Operator is prefered when we have same LHS with diff RHS.

Note: IN Operator is used to Optmise the Query.

Q) display emp details who are working as either clerk (or) manager (or) salesman (or) Analyst ?

```
SQL> Select *
     from emp
     where job IN ('CLERK', 'MANAGER', 'SALESMAN', 'ANALYST');
```

Q) display emp details who are reporting to mgrno 7698, or 7839, or 7788 or 7821 ?

```
SQL> Select *
     from emp
     where mgr IN (7698, 7839, 7788, 7782);
```

Q) display emp details who are earning salary as 3000, 5000 800, 1250, 4000 ?

```
SQL> Select *
     from emp
     where sal IN (3000, 5000, 4000, 800, 1250);
```

⊛ BETWEEN

* It is prefered when the cond^n is in the form of range.

Syntax:

```
Select   * / Column(s)
from     tablename
where    LHS BETWEEN RHS;
```

* The RHS of BETWEEN operator is in range. The range consists of 2 values Lowerlimit value AND Upperlimit value.
* Range should be Specified in the following format

LHS BETWEEN Lowerlimit value AND Upperlimit value.

Note: * Always lowerlimit value should be lesser than the upperlimit value.

Ex:

where 170 BETWEEN 100 AND 500

170 >= 100 T
170 <= 500 T
} True → working as AND Operator.

where 1170 BETWEEN 100 AND 500

T 1170 >= 100
F 1170 <= 500
} False → AND Operator

* BETWEEN operator works on inclusive values [all the values in the given range including lowerlimit value and the upper limit value]
* BETWEEN Operator works same as AND operator.

* Generally BETWEEN Operator in prefered when the cond^n is w.r.t range format.

Q> display emp details who are earning salary in the range of 1000 to 4000?

```
SQL> select *
     from emp
     where sal BETWEEN 1000 AND 4000;
```

Q> display emp details only if the emp no ranges from 7400 to 7600?

```
SQL> select *
     from emp
     where empno BETWEEN 7400 AND 7600;
```

Q> display emp details who are earning Commision in the range of 100 to 1000?

```
SQL> select *
     from emp
     where comm BETWEEN 100 AND 1000;
```

Q> display emp details who joined company from the year 81 and 82?

```
SQL> select *
     from emp
     where hiredate BETWEEN 01-JAN-81 AND 31-DEC-82;
```

⊛ LIKE

* It is used to Specify the pattern or wild cards.

Syntax:

```
Select  * / column[s]
from    tablename
where   LHS  LIKE  'pattern';
```

* The RHS value of LIKE Operator should be enclosed within 'Single quotes'.
* It takes only 1 Pattern at a time.
* If we want to Specify multiple patterns then we should Specify multiple times LIKE Operator.
* Pattern is a Combination of Regular Expression.

➤ Regular Expression

* The combination of Ordinary characters and Meta Characters are known as Regular Expression.

→ Ordinary ~~Expression~~ Characters

* All the Characters except modulus (%) and the underscore (_) are known as Ordinary Characters

→ Meta Characters

* modulus (%) and Underscore (_) are known as meta Characters, Bcoz they have Special ablity w.r.t SQL.

- Underscore (_):
  * It represents exactly 1 Charcacter.
  * That Character can be any character apart from modulus and underscore.

- **modulus (%):**
  - * It represents '0' or 'n' number of characters.
  - * That character can be any character apart from modulus and underscore.

Ex:

'____'	'A___'	'A___Y'	'%a'	'%AA%'
'COOL'	'Arun'	'AUNTY'	'Rama'	'AADHAR'
'JAVA'	'ASHA'	'ANGRY'	'Ravana'	'AAHED'
'ABHI'	'Amma'	'ARRAY'	'Krishna'	'BAZAAR'

* If a pattern is specified along with character position then it is pattern matching ——→ Ex: 'A___'
* If a pattern is specified without a character position then it is wild card such ——→ Ex: '%AA%'

Note: * Pattern is case-sensitive.

Q) Display emp details whose name has character 'A'?

```
SQL> Select *
     from emp
     where ename LIKE '%A%';
```

Q) Display emp details whose name has exactly 4 characters?

```
SQL> Select *
     from emp
     where ename LIKE '____';
```

Q) display emp details whose name ends with character 's'?

```
SQL> Select * from emp
     where ename LIKE '%s';
```

Q) display emp details whose name has exactly 4 or 3 char and belongs to deptno 30?

```
SQL> Select *
     from emp
     where (ename LIKE '____' OR ename LIKE '___')
           AND deptno = 30;
```

Q) display emp details whose name contains character 'A', 'E' and earning salary in the range of 1000 - 4000?

```
SQL> Select *
     from emp
     where ename LIKE '%A%E%' AND Sal BETWEEN
           1000 AND 4000;
```

Q) display emp details whose name contains character 'E' at last but 1 position & working as Clerk or Salesman?

```
SQL> Select *
     from emp
     where ename LIKE '%E_' AND Job IN ('CLERK',
           'SALESMAN');
```

Q) display emp details whose job contains substring 'MAN' and earning salary in the range of 2000 - 5000?

```
SQL> Select *
     from emp
     where job LIKE '%MAN%' AND Sal BETWEEN
           2000 AND 5000;
```

Q) display emp details whose name contains character 'A' last but 2nd Position?

```
SQL> Select * from emp
     where ename LIKE '%A__';
```

Q> display emp details who joined company in the year 81 and working as salesman, manager or president and salary in the range of 500 - 4000?

```
SQL> Select *
     from emp
     where hiredate LIKE '%81' AND job IN ('SALESMAN',
     'MANAGER', 'PRESIDENT') AND Sal BETWEEN
     (500 AND 4000);
```

Q> display emp details who joined company in the month of April and december?

```
SQL> Select *
     from emp
     where hiredate LIKE '%APR%' OR hiredate LIKE '%DEC%';
```

Q> display emp details who joined company in the year 81 who belongs to deptno 20 and 30 and earning salary in the range of 1000 to 5000 and working as either clerk, manager or salesman?

```
SQL> Select *
     from emp
     where hiredate LIKE '%81' AND deptno IN (20, 30)
           AND sal BETWEEN 1000 AND 5000 AND Job
           IN ('CLERK', 'MANAGER', 'SALESMAN');
```

Q> display emp details whose name ends with vowels?

```
SQL> Select *
     from emp
     where ename LIKE '%A' OR ename LIKE '%E' OR
           ename LIKE '%O' OR ename LIKE '%I' OR
           ename LIKE '%U';
```

Q> display emp details who are not earning any Commission?

```
SQL> Select *
     from emp
     where comm IS null;
```

⊛ IS

* It is used to Compare null values.
* Apart from null it doesn't work with any other values.

Syntax:

```
Select * / Column[s];
from tablename
where LHS IS null;
```

Q> display emp details who is not reporting to any manager?

```
SQL> Select *
     from emp
     where mgr IS null;
```

Concatination in SQL

* In SQL in order to do Concatination for string and the columns we use of [||] Pipelines Operators.

Ex:

```
SQL> Select 'dash' || ' ' || ename || ' ' || 'Good Afternoon'
     from emp;
```

```
SQL> Select 'Hey ' || ename || ' your salary is ' || sal
     from emp;
```

```
SQL> Select ename || ' salary is ' || sal || ' and he joined company
     on ' || hiredate from emp;
```

05/11/19

⊛ Logical NOT

* It can't be used independently like any other operators.
* It should be used with a Combination of Specical Operator.
* Combination of NOT Operator as follows.
  - i) IS NOT
  - ii) NOT IN
  - iii) NOT BETWEEN
  - iv) NOT LIKE.

i) NOT IN

* It Selects the record only if the LHS value is not present in the RHS value.

ii) NOT BETWEEN

* It Selects the record only if the LHS value is not in the range of RHS.

iii) NOT LIKE

* It Selects the pattern only if the LHS is not matching the RHS Pattern.

iv) IS NOT

* It Selects the value only if the LHS value is not null.

Q) display emp details who are not working as Clerk Salesman?

```
SQL> Select *
     from emp
     where job NOT IN ('CLERK', 'SALESMAN');
```

Q) display emp details who didn't joined Company in the year 81?

```
SQL> Select *
     from emp
     where hiredate NOT LIKE '%81';
```

Q) display emp details who doesn't belong to deptno (10 and 20) and earning salary in the range 1000 - 3000?

```
SQL> Select *
     from emp
     where deptno NOT IN (10,20) AND Sal BETWEEN
                                  1000 AND 3000;
```

Q) display emp details whose name doesn't contains 6 characters and not earning any commission?

```
SQL> Select *
     from emp
     where ename NOT LIKE '______' AND comm IS null;
```

Q) display emp details who's earning some value as commission?

```
SQL> Select *
     from emp
     where comm IS NOT null;
```

Q) display emp details only for the employee who are having mgr no?

```
SQL> Select *
     from emp
     where mgr IS NOT null;
```

Q) display emp details who's name ends with the consonants?

```
SQL> Select *
     from emp
     where ename NOT LIKE '%a' OR
           ename NOT LIKE '%e' OR
           ename NOT LIKE '%i' OR
           ename NOTLIKE '%o' OR
           ename NOTLIKE '%u';
```

Set Operators

* It is used to combine multiple Query output into a Single output.

① UNION

* It Combines multiple query output into a Single output by giving all values without duplicates.

Ex: $S_1 = \{1, 2, 3, 4\}$ $S_2 = \{3, 4, 5, 6\}$

$S_1 \cup S_2 = \{1, 2, 3, 4, 5, 6\}$

② UNION ALL

* It Combines multiple query output into a Singl output by giving all values with duplicates.

Ex: $S_1 = \{1, 2, 3, 4\}$ $S_2 = \{3, 4, 5, 6\}$

$S_1 \cup S_2 = \{1, 2, 3, 4, 5, 6, 3, 4\}$

③ INTERSECT

* It Combines multiple query output into a Single output by giving common values.

Ex: $S_1 = \{1, 2, 3, 4\}$ $S_2 = \{3, 4, 5, 6\}$

$S_1 \cap S_2 = \{3, 4\}$

④ MINUS

* It Combines multiple query output into a Single output by resulting in uncommon values in 1st Operand.

Ex: $S_1 = \{1, 2, 3, 4\}$ $S_2 = \{3, 4, 5, 6\}$

$S_1 - S_2 = \{1, 2\}$ $S_2 - S_1 = \{5, 6\}$

Ex:

```
SQL> Select ename
     from emp
     where deptno = 30
     UNION
     Select ename
     from emp
     where job = 'SALESMAN';
```

Output:

ALLEN
WARD
MARTIN
BLAKE
TURNER
JAMES

Ex:

```
SQL> Select ename
     from emp
     where deptno = 30
     UNION ALL
     Select ename
     from emp
     where job = 'SALESMAN';
```

Output:

ALLEN
WARD
MARTIN
BLAKE
TURNER
JAMES
ALLEN
WARD
MARTIN
TURNER

Ex:

```
SQL> Select ename from emp
     where deptno = 30
     INTERSECT
     Select ename from emp
     where job = 'SALESMAN';
```

Output:

ALLEN
WARD
MARTIN
TURNER

Ex:

```
SQL> Select ename from emp
     where deptno = 30
     MINUS
     Select ename from emp
     where job = 'SALESMAN';
```

Output:

BLAKE
JAMES

Functions

* Functions are the set of statements that are written inorder to perform Specific Operations.
* Functions are called / invoked by their function-name.

Syntax:

function_name (arg)

* In SQL functions are classified into 2 types
  1. User-defined functions
  2. Built-in / Pre-defined function.

1. User-defined function

* The function which are written by the user is known as user-defind function.
* It can be modified based on the requirement changes.
* In order to write user-defined function we need a language called PLSQL [Programming Language SQL] along with SQL.

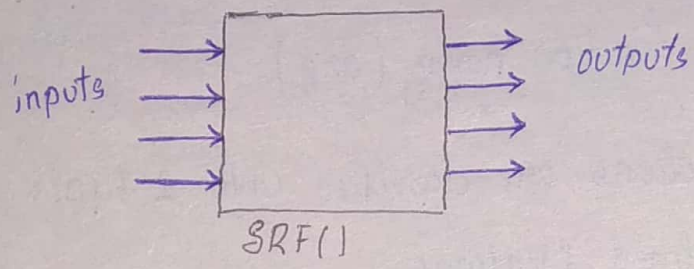
2. Built-in function / Pre-defined function

* The function which are provided by the compiler during the installation of the software is known as built-in function
* They are classified into 2 types.
  1. Single Row function.
  2. multi Row function.

Note: * Built-in function can't be modified.

① Single Row function:

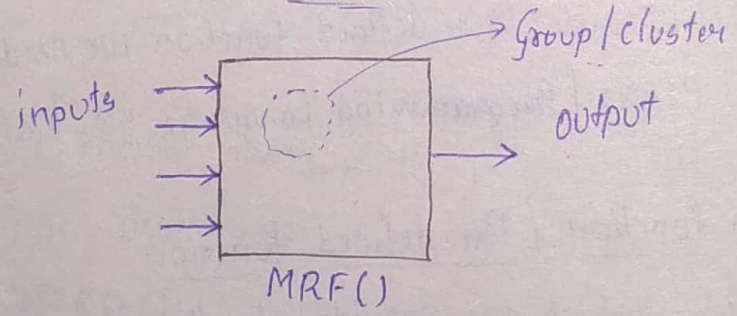
* It takes one input and gives single output.
* If it takes 'n' no of inputs and it given 'n' no of outputs.
* The output count is directly proportional to the input count.



06/11/2019

② Multi Row function:

* It takes n no of inputs and gives only single output.
* In multi row functions groups are created where the input is taken.
* By default it creates Single Group. hence Single output value.
* It is also refered as Group functions / Aggregate functions.



* The multi row functions avilable in SQL are

⊛ MIN (column_name)

→ It gives minimum value for the Specified input.

⊛ MAX (column_name)

→ It gives maximum value for the Specified input.

⊛ AVG (column_name)

→ It gives Average value for the Specified input.

* SUM (Column-name)

→ It gives Summation value for the Specified input.

* COUNT (Column-name)

→ It gives the Count of records for the Specified input.

Note: * All the Multi Row Function takes Single Argument.

Q) display maximum salary among the employee?

```
SQL> Select MAX(sal) from emp;
```

Q) display minimum salary among the employee?

```
SQL> Select MIN(sal) from emp;
```

Q) display maximum salary, minimum salary, Average salary for all the employee?

```
SQL> Select MAX(sal), MIN(sal), AVG(sal) from emp;
```

Q) display maximum salary, totalSalary only for the employee who are working as Salesman?

```
SQL> Select MAX(sal), SUM(sal) from emp
     where job = 'SALESMAN';
```

Q) display lowest salary, highest salary only for the employee who is not earning any comm & belongs to either deptono 10,20?

```
SQL> Select MAX(sal), MIN(sal)
     from emp
     where comm IS null AND deptno (10,20);
```

Q) display the no of employee who are working as salesman?

```
SQL> Select COUNT(*) from emp
     where job = 'SALESMAN';
```

Q) display Avg Salary, total salary, only for the employee whose name contains exactly 5 char and earning b/w 1000-4000?

SQL> Select AVG(sal), SUM(sal) from emp
where ename LIKE '_____' AND Sal BETWEEN 1000 AND 4000;

Q) display lowest salary, highest salary only for the employee who joined the company in the year 81 & employee job contains exactly 7 characters & belongs to deptno 10, 20, 30?

SQL> Select MAX(sal), MIN(sal) from emp
where hiredate LIKE '%81' AND job LIKE '_______'
AND deptno IN (10, 20, 30);

Q) display the no of emp along with avg sal only for the employee's who are working in deptno 30?

SQL> Select COUNT(sal), AVG(sal) from emp
where deptno = 30;

Q) display no of emp along with total sal only for the emp who's name has exactly 4 or 5 characters & joined in the company in the year 81?

SQL> Select COUNT(sal), SUM(sal) from emp
where (ename LIKE '____' OR ename LIKE '_____')
AND hiredate LIKE '%81';

Q) display no of employee who's job contains vowels?

SQL> Select COUNT(*) from emp
where job LIKE '%A%' OR job LIKE '%E%' OR
job LIKE '%I%' OR job LIKE '%O%' OR
job LIKE '%U%';

Q) display employee name & maximum sal?

SQL> MAX(sal), ename from emp; ✕ Wrong Query

Error : Not a Single-Group Group function.

* In Order to achive the Combination of columnname & multi row function we go for Group by clause
* Using Group by clause we can create multiple groups in multi row function.

group by clause

07/11/2019

* It is used to Create multiple output values in the multi row function
* It Creates multiple Groups in the Multi Row Function.

Syntax:

⑤ Select group by Expression.
① from table-name
② where condition[s]
③ group by ref_column[s]
④ order by ref-column[s] Asc/Desc;

* The Combination of Columnname and Multi Row function is a Group by Expression.
* Whenever Group by clause is Specified in the Query the input for this Select Statement is in the form of groups.
* Whenever a multiple columns is Specified in the Group by clause then it Creates Sub Groups along with main Groups.
* Whenever a Single column is Specified in the Group by Clause then it Creates main Groups.

* If multiple ref-columns is specified in the Group by clause then it checks for the unique combination of values.

Ex:

Std-id	Student name	branch	per
11	Dinga	CS	60
12	Guldu	IS	70
13	Sundra	ME	90
14	Sundai	IS	75
15	pavan	CS	60
16	ranjith	IS	75

Q) display branch wise no of Students?

```
SQL> Select branch, count(*)
     from Student
     group by branch;
```

o/p:

branch	count(*)
CS	2
IS	3
ME	1

Q) display per wise no of Students?

```
SQL> Select per, count(*)
     from Student
     group by per;
```

o/p:

Per	count(*)
60	2
90	1
70	1
75	2

Q) display job wise no of Employee's?

```
SQL> Select job, count(*)
     from emp
     group by job;
```

Q) display jobwise no of employee's along with maximum salary for all the emp?

```
SQL> Select job, MAX(sal), COUNT(*)
     from emp
     group by job;
```

Q) display deptno wise no of employee's?

```
SQL> Select deptno, COUNT(*)
     from emp
     group by deptno;
```

Q) display hiredate wise no of employee's except manager?

```
SQL> Select hiredata, COUNT(*)
     from emp
     where job != 'MANAGER'
     group by hiredate;
```

Q) display managerno wise only for the employee whose name contains 4 or 5 characters?

```
SQL> Select mgr, COUNT(*) from emp
     where ename LIKE '____' OR ename LIKE '_____'
     group by mgr;
```

Q) display salwise employee only for the emp whose is not earning any commision?

```
SQL> Select sal, COUNT(*)
     from emp  -> where COMM IS null
     group by sal;
```

Rules of Group by Clause:

* The Columns present in select statement should be present in Group by clause.
* The Columns Specified in Group by clause need not be present in the select statement.
* we can't Specify Multi Row Function in where clause.

Q) display branchwise no of Students for each per?

```
SQL> Select  branch, per, COUNT(*)
     from    Student
     group by branch, per;
```

Q) display jobwise no of employee for each deptno?

```
SQL> Select  job, deptno, COUNT(*)
     from    emp
     group by job, deptno;
```

Q) display mgrwise no of employee for each deptno?

```
SQL> Select  mgr, deptno, COUNT(*)
     from    emp
     group by mgr, deptno;
```

Q) display mgr wise employee ?

```
SQL> Select mgr, COUNT(*)
     from emp
     group by mgr;
```

Q) display deptnowise employee along with the maximum sal for each job!

```
SQL> Select deptno, MAX(sal), COUNT(*), job
     from emp
     group by deptno, job;
```

Q) display hiredate wise no of employee for each job along with total salary except clerk!

```
SQL> Select hiredate, job, SUM(sal), COUNT(*)
     from emp
     where job <> 'clerk'
     Group by hiredate, job;
```

Q) display commision wise no of employee for each manager no!

```
SQL> Select comm, mgr, COUNT(*)
     from emp
     group by comm, mgr;
```

Q) display deptno wise no of employee for each hiredate only for the emp who joined company in the year 81?

```
SQL> Select deptno, hiredate, COUNT(*)
     from emp
     where hiredate LIKE '%81'
     group by deptno, hiredate;
```

Q) display deptnowise no of employee where atleast 3 employee are working ?

```
SQL> Select deptno, COUNT(*)
     from emp
     group by deptno
     having COUNT(*) >= 3;
```

NOTE: * In Order to Specify Multi Row Function as a Condition we go for having clause.

08/11/19

having clause

* It is used to Specify Multi Row Function as a Condition.
* It Filters only Group data.

Syntax:

⑤ Select group by expression
① from table_name
② where condition[s]
③ group by ref-column[s]
④ having condition[s]
⑥ Order by ref-column[s] ASC/DESC;

* It is not independent, it is dependent on group by clause.
* having clause filters the record only after grouping.
* we can Specify Multi Row Function as a Condition / Column Cond".

Q> display branchwise no of Students where atleast 2 Students are Studing?

```
SQL> Select branch, COUNT(*)
     from Student
     group by branch
     having COUNT(*) >= 2;
```

Q> display jobwise no of employee where at least 3 employee are working in respective job?

```
SQL> Select job, COUNT(*)
     from emp
     group by job
     having COUNT(*) >= 3;
```

Q> display mgrwise employee's whose minimum salary is Greater than 1000?

```
SQL> Select mgr, COUNT(*)
     from emp
     group by mgr
     having MIN(sal) > 1000;
```

Q> display jobwise no of Employee except Clerk where atleast 2 employee's are working?

```
SQL> Select job, COUNT(*)
     from emp
     where job <> 'CLERK'
     group by job
     having COUNT(*) >= 2;
```

```
SQL> Select job, COUNT(*)
     from emp
     group by job
     having job <> 'CLERK'
     AND COUNT(*) >= 2;
```

Q) display deptno wise employee for each job except deptno 10 only if the highest salary is > 2000?

```
SQL> Select deptno, COUNT(*), job
     from emp
     group by deptno, job
     having deptno<> 10 AND MAX(sal) > 2000;
```

Q) display hiredate wise no of employee for each deptno except president where atleast 2 emp are working?

```
SQL> Select hiredate, deptno, COUNT(*)
     from emp
     Where job <> 'PRESIDENT'
     group by hiredate, deptno
     having COUNT(*) >= 2;
```

Q) display salary wise no of employee for each job only for the emp who is earning salary in the range of 1000-5000 where atmost 2 emp are working?

```
SQL> Select sal, job, COUNT(*)
     from emp
     where sal BETWEEN 1000 AND 5000
     group by sal, job
     having COUNT(*) <= 2;
```

(or)

```
SQL> Select sal, job, COUNT(*)
     from emp
     group by sal, job
     having sal BETWEEN 1000 AND 5000 AND COUNT(*)<=2;
```

Q) display mgrno wise emp for each job along with the maximum salary only for the employee who is working in deptno 20 & 30 where atmost 3 employee are working?

```
SQL> Select  mgr, job, MAX(sal), COUNT(*)
     from    emp
     where   deptono IN (20,30)
     group by mgr, job
     having  COUNT(*) <= 3;
```

Q) display deptno wise no of emp for each mgr no along with the lowest salary only for the employee who joined in year 81 and where maximum salary should be less than 4000?

```
SQL> Select deptno, mgr, MIN(sal), COUNT(*)
     from emp
     where hiredate LIKE '%81'
     group by deptno, mgr
     having MAX(sal) < 4000;
```

Rules of having clause:

* The Multi Row Function Specified in having clause need not be present in Select Statement.
* In Order to Specify Column Condition in having clause the Same column should be present in groupby clause.

Note: * Prefer where clause for Column Condition.
* Prefer having clause for Multi Row Function has a Condition.

Q> display hiredate wise no of employee for each job only for the emp who is not earning any comm and minimum salary > 1000?

```
Select hiredate, job, COUNT(*)
from emp
where comm is null
group by hiredate, job
having MIN(sal) > 1000;
```

Q> display deptno wise no of employee for each salary only for the employee who are working as salesman, mgr, clerk, where lowest salary greater than 2000 and atleast 1 emp should be working!

```
Select deptno, sal, COUNT(*)
from emp
where job IN('SALESMAN', 'MANAGER', 'CLERK')
group by deptno, sal
having MIN(sal) > 2000 AND COUNT(*) >= 1;
```

Q> display deptno wise no of emp for each job only for the emp who's name containing only '4' or '5' characters and lowest salary should be Greater than 1000?

```
Select deptno, job, COUNT(*)
from emp
where ename LIKE '____' OR ename LIKE '_____'
group by deptno, job
having MIN(sal) > 1000;
```

★ Difference between where clause & having clause

Inwhere clause	having clause
* Independent	* dependent on Group by clause
* filter all the records	* filter only Grouped data.

Q> display commwise no of emp only for the emp who is earning comm and lowest salary less than 3000?

```
Select comm, COUNT(*)
from emp
group by comm
having comm IS NOT null AND MIN(sal) < 3000;
```

Sub Query

* A query with in a query is known as Sub Query
* A query written inside another query is known as SubQuery.

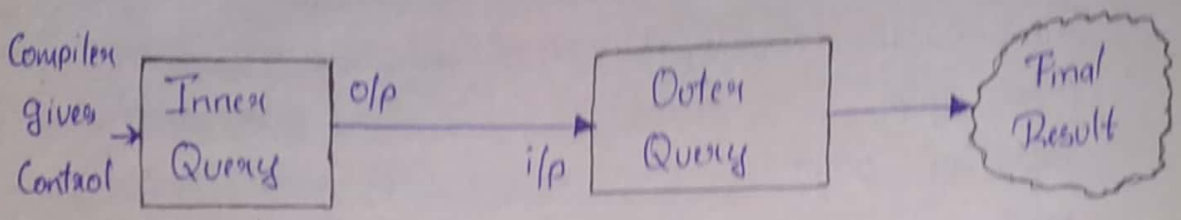
Syntax:

```
Select */Column[s]
from table_name
where (Select */Column[s],
       from table_name
       where Condition[s]);
```

* The Query that is present towards the outer most part is known as Outer query.
* The query that is present towards inner most part is known as Inner query.
* The Combination of Outer query and Inner query is known as Sub Query.
* We can write query in the Nested format hence it is known as Nested Sub Query.

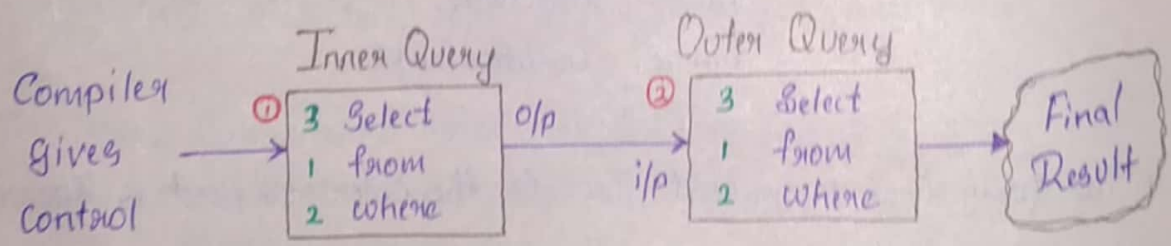
Note: * In Oracle we can Nest upto 255 inner Queries.

Sequence of Execution



* The Compiler gives control to the inner query.
* Inner Query executes based on the Condition Specified.
* The Inner Query o/p is given as an i/p to Outer Query.
* The Outer Query executes based on the Condition Specified and provides the final result.

Note: * Inner Query is Independent.
* Outer Query is Dependent on Inner Query o/p.



* Based on the **output** given by Inner Query the sub Queries are classified into 2 types.
  ① Single Row [Sub Query]
  ② Multi Row Sub Query

① Single Row Sub Query

* If Inner Query gives exactly one o/p then it is known as Single Row Sub Query.

② Multi Row Sub Query

* If Inner Query gives more than one o/p then it is known as Multi Row Sub Query.

Note: * In case of Single Row Sub Query we can use any Relational Operator and IN operator for Comparision.

* Incase of Multi Row Sub Query we should use IN operator for Comparision.

Case 1 : To find Unknown / Undetermined values.

Q) display emp details whose is earning same salary as scott?

```
SQL> Select *
     from emp
     where sal = (Select sal
                  from emp
                  where ename = 'SCOTT');
```

Q) display emp details who are working same job as smith?

```
SQL> Select *
     from emp
     where job IN (Select job
                   from emp
                   where ename = 'SMITH');
```

(OR)

```
SQL> Select *
     from emp
     where job = (Select job
                  from emp
                  where ename = 'SMITH');
```

Q) display emp details who are working in the same deptno as allen?

```
SQL> Select *
     from emp
     where deptno = (Select deptno
                     from emp
                     where ename = 'ALLEN');
```

Q) display emp details who are earning Comm more than Scott's Commission?

```
SQL> Select *
     from emp
     where comm > (Select comm
                   from emp
                   where ename = 'SCOTT');
```

Q) display emp details who joined Company after Turner date joining?

```
SQL> Select *
     from emp
     where hiredate > (Select hiredate
                       from emp
                       where ename = 'TURNER');
```

Q) display emp details who are working in the same job as jones and emp belongs to the same deptno as Adams?

```
SQL> Select *
     from emp
     where job = (Select job
                  from emp
                  where ename = 'JONES') AND
           deptno = (Select deptno
                     from emp
                     where ename = 'ADAMS');
```

Q) display emp details who are earning salary more than Blake?

```
SQL> Select *
     from emp
     where sal > (Select sal
                  from emp
                  where ename = 'BLAKE');
```

Q) display emp details who joined Company after martin's hiredate and emp who are earning salary more than adams salary?

```
SQL> Select *
     from emp
     where hiredate > (Select hiredate
                       from emp
                       where ename = 'MARTIN') AND
           sal > (Select sal
                  from emp
                  where ename = 'ADAMS');
```

Q) display emp details who is not working in same job as scott and earning same value as commission and emp belongs to the same mgr no of Jones?

```
SQL> Select *
     from emp
     where job <> (Select job
                   from emp
                   where ename = 'SCOTT') AND
           comm IS NOT null AND
           mgr IN (Select mgr
                   from emp
                   where ename = 'JONES');
```

Q> display emp details who are earning salary more than Smith's salary and sal is in the range of 1000 - 5000 and emp working is in the same deptno as Ford ?

```
SQL> Select *
     from emp
     where sal BETWEEN 1000 AND 5000 AND
           sal > (Select sal
                  from emp
                  where ename = 'SMITH') AND
           deptno = (Select deptno
                     from emp
                     where ename = 'FORD');
```

Q> display emp details who are working in the same job as turner and joined the company in the year 81 and emp who are earning sal more than James salary?

```
SQL> Select *
     from emp
     where hiredate LIKE '%81' AND
           job IN (Select job
                   from emp
                   where ename = 'TURNER') AND
           sal > (Select sal
                  from emp
                  where ename = 'JAMES');
```

Q> display emp details who joined company after Allen date of joining and emp who's sal is less than King sal & emp who are working in the same job as jones?

```
SQL> Select *
     from emp
```

```
where hiredate > (Select hiredate
                  from emp
                  where ename = 'ALLEN') AND

      Sal > (Select sal
             from emp
             where ename = 'KING') AND

      job = (Select job
             from emp
             where ename = 'JONES');
```

Q) display emp details who are working in the same job as the Smith, jones, allen!

```
SQL> Select *
     from emp
     where job IN (select job
                   from emp
                   where ename IN ('SMITH', 'JONES', 'ALLEN'));
```

Note: * If we can't distinguish between Single Row Sub Query and Multi Row Sub Query we should use IN Operator for Comparision.

Case 2: 12/11/2019

Mapping multiple tables by using common columns between the tables.

---

dept Table

Select * from dept;

Deptno	DName	Loc
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Bostom

Q) display employee details only for the employee who are working in New York Location?

```
SQL>Select * 
    from emp
    where deptno IN (Select deptno
                     from dept
                     where loc = 'NEW YORK');
```

Q) display employee details only for the employee who are working in research dept?

```
SQL> Select *
     from emp
     where deptno IN (Select deptno
                      from dept
                      where dname = 'RESEARCH');
```

Q) display department details for the employee SCOTT?

```
SQL> Select *
     from dept
     where deptno IN(Select deptno
                     from emp
                     where ename = 'SCOTT');
```

Q) display department details only for the employee either working as Salesman or Manager?

```
SQL> Select *
     from dept
     where deptno IN(Select deptno
                     from emp
                     where job IN('SALESMAN','MANAGER'
```

Q) display department details only for the employee who are working as clerk and belongs to Dallas location?

```
SQL> Select *
     from dept
     where deptno IN ( Select deptno
                       from emp
                       where job = 'CLERK' ) AND
           Loc = 'DALLAS';
```

```
SQL> Select *
     from dept
     where Loc = 'DALLAS' AND
           deptno IN ( Select deptno
                       from emp
                       where job = 'CLERK' );
```

Q) display department details only for the employee who joined the company in the year 81 and earning salary in the range of 1000 - 3000?

```
SQL> Select *
     from dept
     where hiredate IN ( Select hiredate
                         from emp
                         where hiredate LIKE '%81' AND
                               Sal BETWEEN 1000 AND 3000 );
```

Q) display department details only for the employee who belongs to manager no 7839 and working in dept accounting?

```
SQL> Select *
     from dept
     where dname = 'ACCOUNTING' AND
           mgr IN ( Select mgr
                    from emp
                    where mgr = 7839 );
```

Q) display dept details only for the emp who is working in the same job as Smith and dept name should contain char 'A'?

```
SQL> Select *
     from dept
     where deptno IN ( Select deptno from emp
                       where job IN ( Select job
                                      from emp
                                      where ename = 'SMITH' ))
           AND dname LIKE '%A%';
```

Q) display department details only for the employee who are working salary less than King's salary and emp name should contains exactly 5 characters?

```
SQL> Select *
     from dept
     where deptno IN ( Select deptno
                       from emp
                       where ename = 'KING' AND
                       or    where sal IN ( Select sal
                                            from emp
                                            where ename = 'KING
           AND ename LIKE '_____' );
```

Q> display department details only for the emp who joined the company after allen's date of joining and working in chicago location!

```
SQL> Select *
     from dept
     where deptno (Select deptno
                   from emp
                   where hiredate > (Select hiredate
                                     from emp
                                     where ename = 'ALLEN'))

          AND loc = 'CHICAGO';
```

Q> display employee details who are earning salary more than Allen salary and working in New York location?

```
SQL> Select *
     from emp
     where sal > (Select sal
                  from emp
                  where ename = 'ALLEN') AND

           deptno IN (Select deptno
                      from dept
                      where loc = 'NEW YORK');
```

Q) display employee details only for the employee who are working in the same job as james and employee belongs to Sales dept!

```
SQL> Select *
     from emp
     where job IN ( Select job
                    from emp
                    where ename = 'JAMES' ) AND
           deptno IN ( Select deptno
                       from dept
                       where dname = 'SALES' );
```

Q) display employee details only for the employee who joined company after Turner date of joining and ~~dept~~ emp who is earning salary less than Scott salary and belongs to Sales dept!

```
SQL> Select *
     from emp
     where hiredate > ( Select hiredate
                        from emp
                        where ename = 'TURNER' ) AND
           sal < ( Select sal
                   from emp
                   where ename = 'SCOTT' ) AND
           deptno IN ( Select deptno
                       from dept
                       where dname = 'SALES' );
```

Q) display employee details only for the employee who is working in the same job as MILLER and employee earning salary less than KING salary and employee joined company after WARD date of joining?

```
SQL> Select *
     from emp
     where job IN (Select job
                   from emp
                   where ename = 'MILLER') AND
           sal < (Select sal
                  from emp
                  where ename = 'KING') AND
           hiredate > (Select hiredate
                       from emp
                       where ename = 'WARD');
```

Q) display department details where atleast 2 employee's are working?

```
SQL> Select *
     from dept
     where deptno IN (Select deptno
                      from emp
                      group by deptno
                      having COUNT(*) >= 2);
```

Q) display employee details only for the employee who are working in the same job as james and employee belongs to Sales dept!

```
SQL> Select *
     from emp
     where job IN ( Select job
                    from emp
                    where ename = 'JAMES' ) AND
           deptno IN ( Select deptno
                       from dept
                       where dname = 'SALES' );
```

Q) display employee details only for the employee who joined company after Turner date of joining and ~~dept~~ emp who is earning salary less than Scott salary and belongs to Sales dept!

```
SQL> Select *
     from emp
     where hiredate > ( Select hiredate
                        from emp
                        where ename = 'TURNER' ) AND
           sal < ( Select sal
                   from emp
                   where ename = 'SCOTT' ) AND
           deptno IN ( Select deptno
                       from dept
                       where dname = 'SALES' );
```

Q) display employee details only for the employee who is working in the same job as MILLER and employee earning salary ~~between~~ less than KING salary and employee joined company after ward date of joining?

```
SQL> Select *
     from emp
     where job IN (Select job
                   from emp
                   where ename = 'MILLER') AND
           sal < (Select sal
                  from emp
                  where ename = 'KING') AND
           hiredate > (Select hiredate
                       from emp
                       where ename = 'WARD');
```

Q) display department details where atleast 2 employee's are working?

```
SQL> Select *
     from dept
     where deptno IN (Select deptno
                      from emp
                      group by deptno
                      having COUNT(*) >= 2);
```

13/11/19

Case 3:

To Find n^{th} Minimum and n^{th} Maximum Mapping Same table.

Q) Display third maximum salary?

```
SQL> Select MAX(sal)
     from emp
     where sal < (Select MAX(sal)
                  from emp
                  where sal < (Select MAX(sal)
                               from emp));
```

Lesser than

Lesser than

Q) display 3rd minimum salary?

```
SQL> Select MIN(sal)
     from emp
     where sal > (Select MIN(sal)
                  from emp
                  where sal > (Select MIN(sal)
                               from emp));
```

Q) display 5th minimum salary?

```
SQL> Select MIN(sal)
     from emp
     where sal > (Select MIN(sal)
                  from emp
                  where sal > (Select MIN(sal)
                               from emp
                               where sal > (Select MIN(sal)
                                            from emp
                                            where sal > (Select MIN(sal)
                                                         from emp))));
```

Q) display 7th Maximum Salary? → Same as Above format

Q> display 4th recent hiredate date?

```
SQL> Select MAX(hiredate)
     from emp
     where hiredate < (Select MAX(hiredate)
                       from emp
                       where hiredate < (Select MAX(hiredate)
                                         from emp
                                         where hiredate < (Select
                                                           MAX(hiredate)
                                                           from emp)));
```

Q> display 3rd Oldest hiredate date?

```
SQL> Select MAX(hiredate)
     from emp
     where hiredate > (Select MAX(hiredate)        -- Greaterthan
                       from emp
                       where hiredate > (Select MAX(hiredate)   -- Greaterthan
                                         from emp));
```

Q> display scott's manager's details?

```
SQL> Select * from emp
     where empno IN (Select mgr from emp
                     where ename = 'SCOTT');
```

Q> display adam's manager's details?

```
SQL> Select *
     from emp
     where empno IN (Select mgr
                     from emp
                     where ename = 'ADAMS');
```

Q) display Scott's manager's manager details?

```
SQL> Select *
     from emp
     where empno IN (select mgr
                     from emp
                     where empno IN (select mgr
                                     from emp
                                     where ename = 'SCOTT'));
```

Q) display miller's manager's manager details?

```
SQL> Select *
     from emp
     where empno IN (Select mgr
                     from emp
                     where empno IN (Select mgr
                                     from emp
                                     where ename = 'MILLER'));
```

Q) display Smith's manager's manager's manager details?

```
SQL> Select *
     from emp
     where empno IN (Select mgr
                     from emp
                     where empno IN (Select mgr
                                     from emp
                                     where empno IN (Select mgr
                                                     from emp
                                                     where
                                     ename = 'SMITH')));
```

Q) display Ford's manager's manager's manager details?

→ Same as Above Query.

Q) display employee details who are reporting to King?

```
SQL> Select *
     from emp
     where mgr IN (Select empno
                   from emp
                   where ename = 'KING');
```

Q) display employee details who are reporting to blake?

```
SQL> Select *
     from emp
     where mgr IN (Select empno
                   from emp
                   where ename = 'BLAKE');
```

Q) display department details for 2nd maximum salary?

```
SQL> Select *
     from dept
     where deptno IN (Select deptno
                      from emp
                      where sal = (Select MAX(sal)
                                   from emp
                                   where sal < (Select MAX(sal)
                                                from emp)));
```

Q) display departmet details for 1st minimum salary?

```
SQL> Select *
     from dept
     where deptno IN (Select deptno
                      from emp
                      where sal = (Select MIN(sal)
                                   from emp));
```

★ Drawbacks of SubQueries

* Since we are writing queries in the nested format the length of the query increases & preformance decreases.
* we can map multiple tables bot can fetch data from Single table.

15/11/2019

② Data Defination Language

1) ~~Create~~

⊛ CREATE

* Create Stmts is used to create table in database.

Syntax:

```
CREATE TABLE table_name
( column1 datatype (size) constraint[s],
  column2 datatype (size) constraint[s],
  ------
  ------
  ------
  columnN datatype (size) constraint[s]);
```

Q> Creating Student table

```
SQL> CREATE TABLE Student (
        std_id     varchar2 (3)   primary key,
        sname      varchar2 (30)  not null,
        branch     varchar2 (3)   not null,
        percentage number (4,2)   not null);
```

Q> Creating Event table

```
SQL> CREATE TABLE Event
     (event_id     number(3)     primary key,
      event_name   varchar(50)   not null,
      event_sponser varchar(30)  not null,
      std_id       number(3)     REFERENCES
                   Student(std_id));
```

Q> Creating Customer table

```
SQL> CREATE TABLE Customer
     (cust_id    varchar(10)   primary key,
      cust_name  varchar(30)   not null,
      cust_add   varchar(30)   not null,
      cust_phno  number(10)    not null);
```

Q> Creating Product table

```
SQL> CREATE TABLE Product
     (prod_id    varchar(10)   primary key,
      prod_name  varchar(30)   not null,
      price      number(7,2)   not null
      cust_id    varchar(10)   REFERENCES
                 Customer(cust_id));
```

Q> Creating payment table

```
SQL> CREATE TABLE Payment
     (pay_id     varchar(10)   primary key,
      pay_type   varchar(30)   not null,
      prod_id    varchar(10)   REFERENCES
                 Product(prod_id));
```

* Creating same copy of the existing table.

Syntax:

```
CREATE TABLE table_name
AS
SELECT * FROM table_name;
```

* The Above Syntax is used to create identical copy of the existing table.

⊛ ALTER

* Alter Statement is used to modify the table w.r.t columns.

Syntax: (i) for Adding a column

```
ALTER TABLE table_name
ADD column_name datatype constraint;
```

(ii) for Renaming a column

```
ALTER TABLE table_name
RENAME COLUMN old_column_name TO
                    new_column_name;
```

(iii) for Deleting a column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Ex:

```
SQL> CREATE TABLE Student1
     (std_id   number(3)  primary key,
      sname    varchar(30)  not null);
```

→ Altering the above Student1 table

⇒ ADD

```
SQL> ALTER TABLE Student1
     ADD branch varchar(3) not null; // table-Altered
```

⇒ RENAME

```
SQL> ALTER TABLE Student1
     RENAME COLUMN Stdame TO Std_name; // table-Altered
```

⇒ DROP [delete]

```
SQL> ALTER TABLE Student1
     DROP COLUMN branch; // table-Altered
```

⊛ RENAME

* Rename statement is used to rename table

Syntax:

```
RENAME old_table_name
TO new_table_name;
```

⊛ TRUNCATE

* It is used to remove all records of a table at one shot.
* It makes table empty.
* It will not remove table database instead it removes records from table.

Syntax:

```
TRUNCATE TABLE table_name;
```

⊛ DROP

* Drop statement is used to remove table from the database.

Syntax:

```
DROP TABLE table_name;
```

Ex:

⇒ TRUNCATE

```
SQL> TRUNCATE TABLE Student1; // table - truncated
```

⇒ DROP

```
SQL> DROP TABLE Student1; // table - droped.
```

---

③ Data Manipulation Language 18/11/2019

⊛ INSERT

* It is used to add records into the table.

Syntax:

```
INSERT INTO table_name values (V1, V2, V3 ...);

(or)

INSERT INTO table_name (Column1, Column2, ...)
        values (V1, V2, V3 ...);
```

⊛ UPDATE

* It is used to update existing data in table with newer data.

Syntax:

```
UPDATE table_name
SET column_name = value
WHERE Condition[s];

(or)

UPDATE table_name
SET column_name = value, column_name = value
WHERE Condition[s];
```

⊛ DELETE

* It is used to remove records from the table.

Syntax:

```
DELETE FROM table_name
WHERE condition[s];
```

Eg:

Q) Inserting a record in employee table?

```
SQL> INSERT INTO emp values (123, 'ABC', 'DANCER'
,144, '01-APR-97', 1000, 100, 10);
```

(or)

```
SQL> INSERT INTO emp (empno, ename, job, mgr,
hiredate, sal, com, deptno) values (123, 'ABC',
'DANCER', 144, '01-APR-97', 1000, 100, 10);
```

Q) Update a Record in employee table?

```
SQL > UPDATE emp
      SET ename = 'RAMYA'
      WHERE empno = 123;
```

(or)

```
UPDATE emp
SET ename = 'RAMYA', sal = '1500'
WHERE empno = 123;
```

Q) delete an Record from employee table?

```
SQL > DELETE FROM emp
      WHERE empno = 123;
```

Joins

* Joins are the special possibility of the select statement where multiple table information can be fetched simultaneosly
* Joins are classified into 5 types.
  1. Cartesian Join / Cross Join
  2. Inner Join / Equi Join
  3. Outer Join
  4. Non-Equi Join
  5. Self Join

1. Cartesian Join :

* It is also known as Cross Join.
* It is a Cross Product of Multiple table.
* Each and Every value gets multiplied with the other value.

Ex:

$A = \{10, 20, 30\}$ $B = \{*, \$, @\}$

$A \circledast B = \{10*, 20*, 30*, 10\$, 20\$, 30\$, 10@, 20@, 30@\}$

Employee

emp_no	emp_name	deptno
1	A	10
2	B	10
3	C	20
4	D	30

Department

deptno	dname
10	Research
20	Accounting
30	Sales
40	Marketing

SQL> SELECT emp_name, dname FROM emp, dept;

A	Research	Valid	C	Research	Invalid
A	Accounting	Invalid	C	Accounting	Valid
A	Sales	Invalid	C	Sales	Invalid
A	Marketing	Invalid	C	Marketing	Invalid
B	Research	Valid	D	Research	Invalid
B	Accounting	Invalid	D	Accounting	Invalid
B	Sales	Invalid	D	Sales	Valid
B	Marketing	Invalid	D	Marketing	Invalid

Note: * In the above Query the count of Invalid record is more than valid record. hence, In the Industry we don't prefer Cartesian join.

19/11/2019

② Inner Join:

* Joining multiple table by utilizing common column's between tables is known as Inner Join / Equi Join.
* Joining multiple table by specifing equals to operator (=) in the joining condition is known as Inner Join / Equi Join.
* Inner join gives only valid records hence it is prefereable.
* In Order to apply Inner join the table should have relationship.
* In Order to join Table1 and Table2 the joining condition is

Tablename1.commoncolumn = Tablename2.commoncolumn

Syntax:

```
SELECT  */ column[s]
FROM    T1, T2
WHERE   Join Condition[s];
```

Q> display employee name along with the department name for all the employee?

```
SQL> Select  ename, dname
     from    emp, dept
     where   emp.deptno = dept.deptno ;
```

Output:

ename	dname
A	research
B	research
C	Accounting
D	Sales

It Gives only Valid output

Q> display employee name along with the department name for the employee scott?

```
SQL> Select  ename, dname
     from    emp, dept
     where   ename = 'SCOTT'  AND
             emp.deptno = dept.deptno ;
```

o/p:

ename	dname
SCOTT	Research

Q> display ename salary along with departname, location only for the employee whose salary is in the range of 500 - 4000?

```
SQL> Select  ename, sal, dname, loc
     from    emp, dept
     where   emp.deptno = dept.deptno  AND
             ~~ename~~ ~~****~~ ~~**~~
             Sal BETWEEN 500 AND 4000;
```

Q> display employee name along with department name only for the employee whose name has 5 characters?

```
SQL> Select  ename, dname
     from    emp, dept
     where   emp.deptno = dept.deptno  AND
             ename LIKE '_____';
```

Q> display employee name salary along with the loc only for the employee who are working in chicago location?

```
SQL> Select ename, sal, dname, loc
     from  emp, dept
     where loc = 'CHICAGO' AND emp.deptno = dept.deptno;
```

Q> display employee name, hiredate along with the department name, location only for the employee who are working in the deptno 30?

```
SQL> Select ename, hiredate, dname, loc
     from  emp, dept
     where  emp.deptno = dept.deptno  AND
            emp.deptno = 30 ;
```

Q> display employee name, department name only for the employee who are working as manager or salesman and working in chicago location?

```
SQL> Select  ename, dname,
     from    emp, dept
     where   emp.deptno = dept.deptno  AND  loc = 'CHICAGO'
             AND job IN ('SALESMAN', 'MANAGER');
```

Q> display employee name, hiredate, department name, location only for the employee who joined the company in the month of FEB or SEP or DEC and location should contain char 'A'?

```
SQL> Select ename, hiredate, dname, loc
     from emp, dept
     where emp.deptno = dept.deptno AND loc LIKE '%A%'
           AND ( hiredate LIKE '%FEB%' OR
                 hiredate LIKE '%SEP%' OR
                 hiredate LIKE '%DEC%' );
```

Oracle Syntax	ANSII Syntax
* Join 2 Tables Syntax Select *|column[s] from T_1, T_2 where T_1.cc = T_2.cc;	* Join 2 Tables Syntax Select *|column[s] from T_1 Inner join T_2 on T_1.cc = T_2.cc;
* Join 3 Tables Syntax Select *|column[s] from T_1, T_2, T_3 where T_1.cc = T_2.cc and T_2.cc = T_3.cc;	* Join 3 Tables Syntax Select *|column[s] from T_1 inner join T_2 on T_1.cc = T_2.cc inner join T_3 on T_2.cc = T_3.cc;
* In from clause we can Specify 'n' no of Tables from T_1, T_2, T_3, ... T_n	* In from clause we can Specify max of 2 Tables from T_1 inner join T_2

Q> display employee name, hiredate, along with the deptment name only for the employee who joined the company in year 81?

```
SQL> Select ename, hiredate, dname
     from emp inner join dept
     on emp.deptno = dept.deptno AND
        hiredate LIKE '%81';
```

Q> display country name, region name for all the Countries?

```
SQL> Select country_name, region_name
     from Countries inner join regions
     on Countries.region_id = regions.region_id;
```

Q> display Countryname, regionname only for those regions whose region name has char 'A'?

```
SQL> Select country_name, region_name
     from countries inner join regions
     on Countries.region_id = regions.region_id AND
        region_name LIKE '%A%';
```

Q> display regionname, countryname, along with the city for all the country?

```
SQL> Select country_name, region_name, city
     from Countries inner join regions
     on Countries.region_id = regions.region_id
     inner join locations on Countries.country_id =
        locations.country_id;
```

Q> display regionname, country name, city, street_Address for all the country?

```
SQL> Select country_name, region_name, city, street_Address
     from countries inner join regions
     on countries.region_id = regions.region_id
     inner join locations on
     countries.country_id = locations.country_id;
```

Q> display regionname, countryname, city only for the cities whose name starts with char 'B'?

```
SQL> Select country_name, region_name, city
     from countries inner join regions
     on countries.region_id = regions.region_id
     inner join locations on city LIKE 'A%' AND
     countries.country_id = locations.country_id;
```

20/11/2019

③ Outer Join:

* It is a combination of inner join and invalid records.

(or)

* Joining multiple tables by utilizing common column's b/w tables which results in both valid and invalid records.
* valid records from all the tables and Invalid records only from the Specified table.

④ Left Outer Join

* It gives valid record from both the tables and Invalid record only from left table

Oracle Syntax:

```
Select   */column[s]
from     T1, T2
where    T1.cc = T2.cc(+);
```

ANSII Syntax:

```
Select   */column[s]
from     T1 left outer join T2
on       T1.cc = T2.cc;
```

Optional

⊛ Right Outer Join

* It gives valid records from both the tables and invalid records only from right table.

Oracle Syntax:

```
Select   */column[s]
from     T1, T2
where    T1.cc(+) = T2.cc;
```

ANSII Syntax:

```
Select   */column[s]
from     T1 Right outer join T2
on       T1.cc = T2.cc;
```

optional

⊛ Full Outer Join:

* It gives valid records from both the tables and invalid records from both left and right table.

* It gives valid and invalid records from both the tables.

Oracle Syntax:

```
select  * / column[s]
from    T1, T2
where   T1.cc = T2.cc (+)
UNION
select  * / column[s]
from    T1, T2
where   T1.cc (+) = T2.cc;
```

ANSII Syntax:

```
select  * / column[s]
from    T1 full outer join T2
on      T1.cc = T2.cc;
```

optional

Note:

* LOJ = Inner join + invalid (left)
  ROJ = Inner join + invalid (right)
  FOJ = Inner join + invalid (left + right)

* A table is specified as left or Right based on the joining condition.
* Whatever the sequence that is specified in the joining condition the table is desides as left or Right.

Q> display valid records from employee and department table and invalid records from only department table?

```
SQL> select  *
     from    emp, dept
     where   emp.deptno (+) = dept.deptno;
```

Q> display employee name, departmentname along with valid record from both the table and invalid record from employee table?

Oracle

```
SQL> Select  ename, dname
     from    emp, dept
     where   emp.deptno = dept.deptno (+);
```

ANSII

```
SQL> Select  ename , dname
     from    emp left outer join dept
      on     emp.deptno = dept.deptno ;
```

Q> display employee name, department name along with valid and invalid records from both the table?

Oracle

```
SQL> Select  ename , dname
     from    emp , dept
     where   emp.deptno = dept.deptno (+) UNION
     Select  ename , dname
     from    emp , dept
     where   emp.deptno (+) = dept.deptno ;
```

ANSII

```
SQL> Select  ename , dname
     from    emp full outer join dept
      on     emp.deptno = dept.deptno ;
```

Q> display region name , country name and invalid records only from regions table and valid records from both table?

Oracle

```
SQL> Select  country_name , region_name
     from    countries , regions
     where   regions.region_id = countries.region_id (+);
```

ANSII

```
SQL> Select  country_name , region_name
     from    countries right join regions
     on      regions.region_id = countries.region_id ;
```

Q) display region_name, country name, & invalid records only from countries table and valid records from both table?

Oracle

```
SQL> Select country_name, region_name
     from countries, regions
     where regions.region_id (+) = Countries.region_id;
```

ANSII

```
SQL> Select country_name, region_name
     from countries left join regions
     on regions.region_id = Countries.region_id;
```

Q) display region_name, country name, along with valid and invalid records from both the tables?

ANSII

```
SQL> Select country_name, region_name
     from countries full join regions
     on regions.region_id = countries.region_id;
```

Oracle

```
SQL> Select country_name, region_name
     from countries, regions
     where regions.region_id (+) = Countries.region_id UNION
     Select country_name, region_name
     from countries, regions
     where regions.region_id = Countries.region_id (+);
```

22/11/2019

(14) Non-Equi Joins:

* Joining multiple tables which are not having relationship is known as Non-Equi Joins.
* Joining multiple tables without Specifing '=' operator in joining cond".

Rules

* The table should not have relationship.
* It should consists of relavent data [the column names can be different but datatye and size should be same]

Q) display employee name, salary along with grade for all the employee?

```
SQL> Select ename, sal, grade
     from emp, salgrade
     where sal BETWEEN losal AND hisal;
```

Q) display employee name, salary, grade only for the employee who are not working as manager?

```
SQL> Select ename, sal, grade
     from emp, salgrade
     where job NOT IN 'MANAGER' AND sal BETWEEN
     losal AND hisal;
```

Q) display employee name, salary, grade, hiredate, only for the employee whose grade is >= 3 [greater than equal to 3] ?

```
SQL> Select ename, sal, grade, hiredate
     from emp, salgrade
     where grade >= 3 AND sal BETWEEN losal AND hisal;
```

Q) display employee name, job, salary, grade only for the employee who is not earning any comm and belongs to either manager (or) Clerk (or) salesman?

```
SQL> Select ename, job, sal, grade
     from emp, salgrade
     where comm Is null AND sal BETWEEN losal
     AND hisal AND job IN ('MANAGER', 'CLERK'
     'SALESMAN');
```

Q) display employee name, salary, grade, deptname for all the employee?

```
SQL> Select ename, sal, dname, grade
     from emp, salgrade, dept
     where emp.deptno = dept.deptno AND sal BETWEEN
           losal AND hisal;
```

Q) display employee name, salary, dept name, grade only for the employee who are working in chicago location?

```
SQL> Select ename, sal, dname, grade
     from emp, salgrade, dept
     where emp.deptno = dept.deptno AND loc = 'CHICAGO'
           AND sal BETWEEN losal AND hisal;
```

Q) display employee name, salary, grade, deptno, dept name only for the emp who salary is less than 4000?

```
SQL> Select ename, sal, grade, dname, emp.deptno
     from emp, dept, salgrade
     where emp.deptno = dept.deptno AND sal < 4000
           AND sal BETWEEN losal AND hisal;
```

Q) display employee name, salary, grade, location only for the employee who are working in sales department?

```
SQL> Select ename, sal, grade, loc
     from emp, dept, salgrade
     where emp.deptno = dept.deptno AND sal BETWEEN
           losal AND hisal AND dname = 'SALES';
```

Q) display deptname, salary, grade only for the employee whose deptname has character 'A'?

```
SQL> Select dname, sal, grade
     from emp, dept, salgrade
     where emp.deptno = dept.deptno AND dname LIKE '%A%'
      AND sal BETWEEN losal AND hisal;
```

⑤ Self Join:

* Joining the same table to itself is known as self join.

(or)

* Referring the same table to itself is known as self join.
* Except self join all other joins are with respect to multiple tables but self joins is with respect to single table.
* In self join we have 2 cases

Case 1:- Same table with different columns

Syntax:

```
Select  */column[s]
from    T1
where   T1.C1 = T1.C2 ;
```

Case 2:- Same table with same columns

Syntax:

```
Select  */column[s]
from    T1
where   T1.C1 = T1.C1 ;
```

Note: * As for the Rules Join minimum 2 tables should be specified in from clause.

Q) display employee name along with manager name?

SQL> Select A.ename As empname, B.ename As mgrname
from emp A, emp B
where A.mgr = B.empno;

Table Diagram:

emp A

empno	emp name	mgr
1	A	4
2	B	3
3	C	2
4	D	1

emp B

empno	empname	mgr
1	A	4
2	B	3
3	C	2
4	D	1

Output:

empname	mgrname
A	D
B	C
C	B
D	A

Q) display employeename, employee salary along with manager name and manager salary for all the employee?

SQL> Select A.ename As empname, A.sal As empsal,
B.ename As mgrname, B.sal As mgrsal
from emp A, emp B
where A.mgr = B.empno;

Q> display employee name, manager name for all the employee scott?

```
SQL> Select A.ename AS empname, B.ename AS mgrname
     from emp A, emp B
     where A.mgr = B.empno AND A.ename = 'SCOTT';
```

Q> display employee name, employee date of joining, manager name manager date of joining for all the employee?

```
SQL> Select A.ename AS empname, A.hiredate AS emphiredate
          B.ename AS mgrname, B.hiredate AS mgrhiredate
     from emp A, emp B
     where A.mgr = B.empno;
```

25/11/19

Case 2 :-

2019

Q> display employee name who are earning same salary?

```
SQL> Select A.ename AS emp1, B.ename AS emp2
     from emp A, emp B
     where A.sal = B.sal AND A.ename <> B.ename;
```

with duplicates

without duplicates

emp A

empno	empname	sal
1	A	100
2	B	200
3	C	300
4	D	100

emp B

empno	ename	sal
1	A	100
2	B	200
3	C	300
4	D	100

o/p with duplicates

emp1	emp2
A	A
A	D
B	B
C	C
D	A
D	D

o/p without duplicates

emp1	emp2
A	D
D	A

Q> display employeename who joined company on same date?

SQL> Select A.ename AS emp1 , B.ename AS emp2
from emp A , emp B
where A.hiredate = B.hiredate AND A.ename <> B.ename;

Q> display employeename who are working in same job?

SQL> Select A.ename AS emp1, B.ename AS emp2
from emp A, emp B
where A.job = B.job AND A.ename <> B.ename;

Q> display employee name who are working in the same deptno?

SQL> select A.ename AS emp1, B.ename AS emp2
from emp A, emp B
where A.deptno = B.deptno AND A.ename <> B.ename;

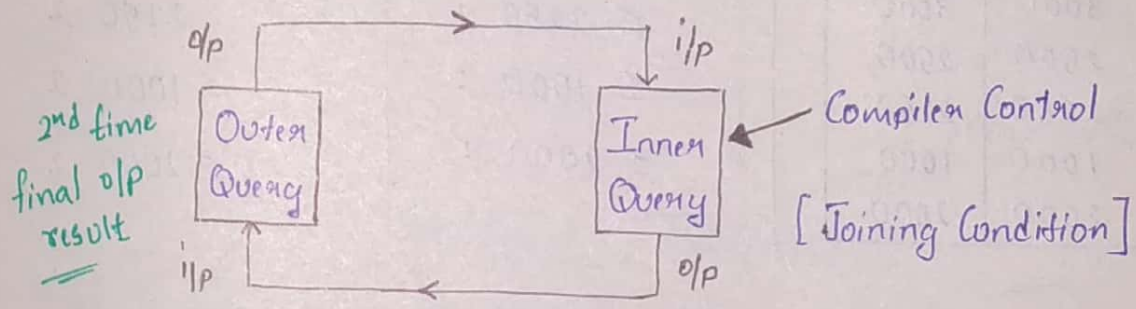
Q> display employeename who are reporting to same manager number?

SQL> Select A.ename AS emp1 , B.ename AS emp2
from emp A, emp B
where A.~~deptno~~ mgr = B.mgr AND A.ename <> B.ename;

★ Co-Related Subqueries :-

* It is advance type of subquery which is the combination of subqueries and joins.
* Co-relation is present b/w outer query and inner query by specifying join condition in the inner query.
* In co-related subqueries both outer query and inner query are inter depended.

  i.e., Inner query depends on outer query for input and Outer query depends on Inner query for input.



* The compiler gives control to the inner query because of the joining condition the control is given back to the outer query.
* The outer query executes partially i.e, excluding where where clause.
* The outer query output is given as a input to the inner query.
* Inner query executes completely and inner query output is given as input to the outer query.
* For the 2nd time outer query executes completely i.e., including where clause and provides final result.

Q) Write a query to display 4th maximum salary using the co-related subquery?

```
SQL> Select distinct sal from emp A
     where 4 = ( Select count (distinct sal)
                 from emp B
                 where A.sal ≤ B.sal );
```

1st time → outer query
2nd time → inner query
[final Result]

Output:

emp A	emp B
5000	5000
3000	3000
2500	2500
2250	2250
1000	1000
2000	2000

```
5000 ≤ 5000  ✓ 1
     ≤ 3000  1
     ≤ 2500  1
     ≤ 2250  1
     ≤ 1000  1
     ≤ 2000  1
```

```
3000 ≤ 5000  ✓ 1
3000 ≤ 3000  ✗ 2
     ≤ 2500  2
     ≤ 2250  2
     ≤ 1000  2
     ≤ 2000  2
```

```
2500 ≤ 5000  ✓ 1
2500 ≤ 3000  ✗ 2
2500 ≤ 2500  ✗ 3
     ≤ 2250  3
     ≤ 1000  3
     ≤ 2000  3
```

```
2250 ≤ 5000  ✓ 1
2250 ≤ 3000  ✗ 2
2250 ≤ 2500  ✗ 3
2250 ≤ 2250  ✓ 4
     ≤ 1000  4
     ≤ 2000  4
```

final Result
4th max Salary is
2250

Q) Write a query to display 3rd minimum salary?

```
SQL> Select distinct sal
     from emp A
     where 3 = ( Select count (distinct sal)
                 from emp B
                 where A.sal ≥ B.sal );
```

1st time → outer query
2nd time → inner query
[final Result]

O/P:

1000 ≥ 1000 ✓1	5000 ≥ 1000 ✓1	2000 ≥ 1000
≥ 5000 1	≥ 5000 ✓2	≥ 5000
≥ 2000 1	≥ 2000	≥ 2000
≥ 7000 1	≥ 7000	≥ 7000
≥ 9000 1	≥ 9000	≥ 9000
≥ 2500 1	≥ 2500	≥ 2500
7000 ≥ 1000	9000 ≥ 1000	2500 ≥ 1000
≥ 5000	≥ 5000	≥ 5000
≥ 2000	≥ 2000	≥ 2000
≥ 7000	≥ 7000	≥ 7000
≥ 9000	≥ 9000	≥ 9000
≥ 2500	≥ 2500	≥ 2500 ✓3

final Result is 2500

Q> display 10th maximum salary using C-o-releated subquery!

```
SQL> select distinct sal
     from emp A
     where 10 = ( select count ( distinct sal)
                  from emp B
                  where A.sal ≤ B.sal);
```

Q> display 10th maximum hiredate using Co-related subquery?

```
SQL> select distinct hiredate
     from emp A
     where 10 = ( select count ( distinct hiredate)
                  from emp B
                  where A.hiredate ≤ B.hiredate );
```

Q) display 5th minimum hiredate using Co-related subquery?

```
SQL> select distinct hiredate
     from emp A
     where 5 = ( select count ( distinct hiredate)
                 from emp B
                 where A.hiredate ≥ B.hiredate);
```

Q) display 90th maximum salary using Co-related subquery?

```
SQL> select distinct Sal
     from
     where 90 = ( select count ( distinct sal
                  from
                  where A.sal
```

Note: * To find nth maximum and nth minimum value we go for Co-related subquery.

26/11/19

Pseudo Columns

* They are false columns present in each and every table in the database.
* They are reflected in output only when the pseudo column names are specified in the select statement.
* Pseudo columns avilable in SQL are rownum, rowid.

~~ex display~~ Syntax:

```
select * / columns
from Table_name
where Condition[s] (Pseudo Columns);
```

ROWID	ROWNUM
* It is a 18 bit unique address	* It is a unique Sequential no.
* It is a static	* It is Dynamic.
* It is Generated by the insert Statement.	* It is Generated by the Select Statement.
* It defines the physical address of the record in the table [Database name, Table name, Column name]	* It gives the count of records in the table

Q> display 1st record in the employee table?

```
SQL> Select *
     from emp
     where rownum = 1;
```

Q> display last record in the employee table?

```
SQL> Select *
     from emp
     where rowid = ( Select MAX(rowid)
                     from emp );
```

Q> write a query to display 1st record of employee table?

```
SQL> Select *
     from emp
     where rowid = ( Select MIN(rowid)
                     from emp );
```

Q> display 6th record of the employee table?

```
SQL> Select  *
     from    emp  A
     where   6 = (Select  COUNT (rowid)
                  from    emp  B
                  where   A.rowid >= B.rowid) ;
```

Q> display 3rd record of the employee table?

```
SQL> Select  *
     from    emp  A
     where   3 = (Select  COUNT (rowid)
                  from    emp  B
                  where   A.rowid >= B.rowid) ;
```

Q> display 10th record of the employee table?

```
SQL> Select  *
     from    emp  A
     where   10 = (Select  COUNT (rowid)
                   from    emp  B
                   where   A.rowid >= B.rowid) ;
```

Q> display 3rd maximum record of the Employee table?

```
SQL> Select  *
     from    emp  A
     where   3 = (Select  COUNT (rowid)
                  from    emp  B
                  where   A.rowid <= B.rowid) ;
```

MINIMUM [>=] , MAXIMUM [<=]

Q> display 6th maximum record of the employee table?

```
SQL > Select *
      from emp A
      where 6 = ( Select COUNT (rowid)
                  from emp B
                  where A.rowid <= B.rowid );
```

Q> display 7th minimum / 7th record of the employee table?

```
SQL > Select *
      from emp A
      where 7 = ( Select COUNT (rowid)
                  from emp B
                  where A.rowid >= B.rowid );
```

Q> display 1st 3 records of the employee table?

```
SQL > Select *
      from emp
      where rownum IN (1, 2, 3);
```

Q> display 2nd, 4th, 6th record of employee table?

```
SQL > Select *
      from emp A
      where ( Select COUNT (rowid)
              from emp B
              where A.rowid >= B.rowid ) IN (2, 4, 6);
```

Q) display 3rd, 6th, 9th, 12th record of employee table?

```
SQL> Select *
     from emp A
     where (select COUNT(rowid)
            from emp B
            where A.rowid >= B.rowid) IN (3,6,9,12);
```

Q) display last 3 records of the employee table?

```
SQL> Select *
     from emp A
     where (select COUNT(rowid)
            from emp B
            where A.rowid <= B.rowid) IN (1,2,3);
```

Q) display last 10 records of the employee table?

```
SQL> Select *
     from emp A
     where (select COUNT(rowid)
            from emp B
            where A.rowid <= B.rowid) BETWEEN 1 AND 10;
```

Q) display 3 to 8 records of the employee table?

```
SQL> Select *
     from emp A
     where (select COUNT(rowid)
            from emp B
            where A.rowid >= B.rowid) BETWEEN 3 AND 8;
```

Note: * Whenever the table information is not revealed then we go for Pseudo Columns.

Single Row Functions

27/11/2019

⊛ Character Functions:

* It works wrt column or values which is of the Character type.
* Character functions are classified into
  a) Case manipulation function.
  b) Character manipulation function.

a) Case Manipulation function

① UPPER

* UPPER function converts all letters in the specified string to uppercase. If there are characters in the string that are not letters, they are unaffected by this function.

Syntax:

```
UPPER(arg1)
```

arg1 → column name / String value

② LOWER

* LOWER function converts all letters in the specified string to lowercase.

Syntax:

```
LOWER(arg1)
```

arg1 → Column name / String value.

Eg:

~~OS SQL>~~

★ Dual

⇒ Dual is a dummy table that is used to handle direct values which are not present in the table.

Eg:

i> SQL> Select UPPER('jeevan') from dual;

output :

UPPER('jeevan')
JEEVAN

ii> SQL> Select UPPER('chandu'), UPPER('nandu'), UPPER('anu') from dual;

output:

UPPER('chandu')	UPPER('nandu')	UPPER('anu')
CHANDU	NANDU	ANU

iii> SQL> Select LOWER(ename) from emp;

output:

LOWER(ename)
allen
smith
ford
martin
:

iv> SQL> Select LOWER('ANU'), LOWER('UMA') from dual;

output:

LOWER('ANU')	LOWER('UMA')
anu	uma

Note: * we can use Single Row Function in where clause.

Eg: SQL> Select * from emp
where LOWER(ename) = 'ALLEN';

③ INITCAP

* INITCAP function sets the first character in each word to Upper case & the rest to lowercase.

Syntax:

INITCAP(arg1)

arg1 → column name / string value

Eg:

SQL> Select INITCAP('abhi ram'), INITCAP('uma')
from dual;

output:

INITCAP('abhi ram')	INITCAP('uma')
Abhi Ram	Uma

⑤ Character Manipulation function:

④ CONCAT

* CONCAT function allows you to concatinate two columns / strings together.

Syntax:

CONCAT(arg1, arg2)

arg1, arg2 → column name / string value.

Eg:

i> SQL> select CONCAT('Abhi', 'Ram') from dual;

output:

CONCAT('Abhi', 'Ram')
Abhi Ram

ii> SQL> Select CONCAT(CONCAT('JATREE', 'Abhi'), 'Ram')
from dual;

Output: CONCAT(CONCAT('JATREE', 'Abhi'), 'Ram')

JATREE Abhi Ram

iii> SQL> Select CONCAT(CONCAT('A', 'B'), CONCAT('C', 'D'))
from dual;

Output: CONCAT(CONCAT('A', 'B'), CONCAT('C', 'D'))

ABCD

⑤ SUBSTR

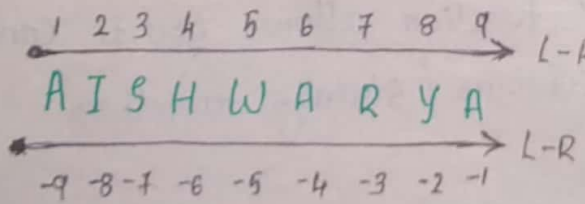
* SUBSTR function allows you to extract a part of a string / substring from a string.

Syntax:

SUBSTR(arg1, arg2, arg3)

arg1 → String / columns
arg2 → ~~String / columns~~ Start Position
arg3 → length / no of characters

Eg:



SUBSTR('AISHWARYA', 2, 3) // ISH

SUBSTR('AISHWARYA', 1, 4) // AISH

SUBSTR('AISHWARYA', 5, 4) // WARY

SUBSTR('AISHWARYA', 6, 4) // ARYA

SUBSTR('AISHWARYA', 8, 4) // YA

SUBSTR('AISHWARYA', -1, 1) // A

SUBSTR('AISHWARYA', -4, 3) // ARY

SUBSTR('AISHWARYA', -2, 2) // YA

* Arg3 is optional. It is the number of characters to extract if this parameter is omitted the SUBSTR function will return the entire string.
* Where arg2 and arg3 are both integers.

i> SQL> Select SUBSTR('AISHWARYA', -2, 3) from dual;

Output:

SUBSTR
YA

ii> SQL> Select SUBSTR('AISHWARYA', 3) from dual;

Output:

SUBSTR
SHWARYA

iii> SQL> Select SUBSTR('AISHWARYA', -3) from dual;

Output:

SUBSTR
RYA

⑥ Length

* length is used to find the length of given String

Syntax:

length(arg1)

arg1 → column / string

Eg:

i> SQL> Select length('AISHWARYA') from dual;

length
9

ii> SQL> Select ename, from emp
where length(ename) = 5;

⑦ INSTR

* INSTR function returns the index position of a character / substring in a string.

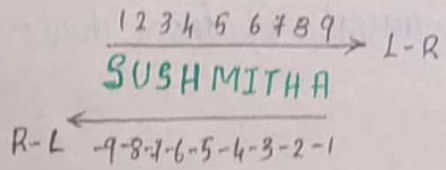
Syntax:

INSTR(arg1, arg2, arg3, arg4)

arg1 → column / string
arg2 → search char / string
arg3 → start position
arg4 → Occurance.

* Arg3, Arg4 is optional, if they are omitted by default they are taken as 1.
* Arg3, Arg4 are integers.

Eg:



i> SQL> Select INSTR('SUSHMITHA', 'H', 2, 1) from dual;

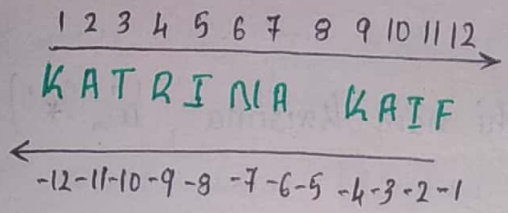
Output:

INSTR
4

ii> SQL> Select INSTR('SUSHMITHA', 'A', 1, 2) from dual;

Output:

INSTR
0



* INSTR('KATRINA KAIF', 'k', 1, 2) // 9

INSTR('KATRINA KAIF', 'A', 3, 2) // 10

INSTR('KATRINA KAIF', 'A', 1, 4) // 0

Note: * When we give -ve values it checks from R → L

* INSTR('KATRINA KAIF', 'k', -1, 2) // 1

INSTR('KATRINA KAIF', 'I', -4, 2) // 0

INSTR('KATRINA KAIF', 'k', -1) // 9 → 1st Occurance by default chck

⑧ REPLACE:

* REPLACE function replace a sequence of characters in a string with another set of characters.

Syntax:

REPLACE(Arg1, arg2, arg3)

arg1 → column / string

arg2 → search char / string

arg3 → replacement char / string.

* arg3 is optional, It is not specified by default it is taken as NULL.

Eg:

i> SQL> Select REPLACE('abhi ram krishna', 'a', '*') from dual;

output:

REPLACE
*bhi r*m krishn*

ii> SQL> Select REPLACE('abhi ram', 'ram', $) from dual;

output:

REPLACE
abhi $

iii> SQL> Select REPLACE('AISHWARYA', 'A') from dual;

output:

REPLACE
ISHWRY

28/11/2019

④ REVERSE

* REVERSE function reverses a given input.

Syntax:

REVERSE (arg1)

arg1 → Column / String

Eg:

Q> display employee's who's name is palindrome?

SQL> Select [redacted], *
from emp
where ename = REVERSE(ename);

output:

ENAME	JOB	MGR	HIREDATE
NAYAN	DANCE	1421	21-FEB-87

★ Q> display emp names and count the characters 'A' in all the employee name?

ename

SQL> Select, LENGTH (ename) - LENGTH (REPLACE (ename, 'A')), ↑
from emp;

Output:

ENAME	LENGTH(ENAME) - LENGTH(REPLACE(ENAME, 'A'))
ALLEX	1
FORD	0
MARTIN	1
TURNER	0
BLAKE	1
ADAMS	2
SMITH	0
;	;

★ Q> display last char of employee name as * (star)?

SQL> Select ename, REPLACE(ename

⑧ Number functions: It works w.r.t column/value which is of number type.

① ROUND

* ROUND is used to round the value to specific decimal.

Syntax:

ROUND(arg1)

arg1 → column / direct value.

Eg: (i) SQL> Select ROUND(9.6) from dual;

Output:

ROUND(9.6)
10

(ii) SQL> Select ROUND(9.4) from dual;

Output:

ROUND(9.4)
9

(iii) SQL> Select ROUND(9.5) from dual;

Output:

ROUND(9.5)
10

② TRUNC

* Truncate the values to specified decimal.

Syntax:

TRUNC(arg1)

arg1 → column / direct value.

Eg: (i) >Select TRUNC(9.5) from dual;

Output:

TRUNC(9.5)
9

(ii) SQL> Select TRUNC(23.45) from dual;

Output:

TRUNC(23.45)
23

(iii) SQL> Select TRUNC(999.2) from dual;

Output:

TRUNC(999.2)
999

③ MOD

* Returns remainder of division.

Syntax:

MOD(m,n)

m → Column / direct value

n → Column / direct value

Eg:

(i) SQL> Select MOD(10,2) output:
from dual;

MOD(10,2)
0

(ii) SQL> Select MOD(7,2) output:
from dual;

MOD(7,2)
1

(iii) SQL> Select MOD(99,4) output:
from dual;

MOD(99,4)
3

④ SQRT

* SQRT function returns the square root of a number.

Syntax:

SQRT(n) n → Column / direct value

Eg:

(i) SQL> select SQRT(169) output:
from dual;

SQRT(169)
13

(ii) SQL> Select SQRT(625) output:
from dual;

SQRT(625)
25

(iii) SQL> Select SQRT(9801) output:
from dual;

SQRT(9801)
99

⑤ POWER

* POWER functions returns m raised to the n^{th} power.

Syntax:

POWER(m,n)

m → column / direct value

n → column / direct value

Ex:

(i) SQL> Select POWER(9,3) from dual;

output:

POWER(9,3)
729

(ii) SQL> Select POWER(3,4) from dual;

output:

POWER(3,4)
81

⑥ Date Functions:

* It works w.r.t value / column which is of data type.

→ Oracle database stores dates in an internal numeric format:

Century, Year, Month, day, Hours, Minites, Seconds

→ The default date display format is DD-MON-YY

→ Sysdate: It is a function that returns date.

→ Systimestamp: It is a function that returns date, time including seconds & miliseconds and time Zone [GMT]

Eg:

(i) SQL> select SYSDATE from dual;

output:

SYSDATE
28-NOV-19

(ii) SQL> Select SysTIMESTAMP from dual;

output:

SYSTIMESTAMP
28-NOV-19
03.43.05.957000 PM
+05:30

① EXTRACT

* EXTRACT function extracts only date/month/year value from the input.

Syntax:

EXTRACT (DAY from input date)

EXTRACT (MONTH from input date)

EXTRACT (YEAR from input date)

Eg:

(i) SQL> Select extract (DAY from sysdate) from dual;

output:

EXTRACT
28

(ii) SQL> Select EXTRACT (MONTH from SYSDATE) from dual;

output:

EXTRACT
NOV

(iii) SQL> Select EXTRACT (YEAR from SYSDATE) from dual;

output:

EXTRACT
19

(iv) SQL> Select EXTRACT (YEAR from HIREDATE) from emp;

output:

EXTRACT
97
87

② ADD_MONTHS

* ADD_MONTHS function adds specified number of months to input date.

Syntax:

ADD_MONTHS(input date , n)

n → no of months to be added

Eg: (i) SQL> Select ADD_MONTHS ('28-FEB-97', 12) from dual;

Output:

ADD_MONTHS
28-FEB-98

(ii) SQL> Select ADD_MONTHS(SYSDATE, 24) from dual;

Output:

ADD_MONTHS
28-NOV-21

③ MONTHS_BETWEEN

* Calculates no of months between two given dates.

Syntax:

MONTHS_BETWEEN(date1, date2) ---> (date1 > date2)

Eg:

(i) SQL> Select MONTHS_BETWEEN ('01-DEC-2020', '14-FEB-2019') from dual;

Output:

MONTHS_BETWEEN
21.5806

TRUNC

(ii) SQL> Select TRUNC(MONTHS_BETWEEN ('01-DEC-2020', '14-FEB-2019')) from dual;

Output:

MONTHS_BETWEEN
21

* Q> write a query to find the experience of all the employee till current date?

SQL> Select ename, TRUNC(MONTHS_BETWEEN('SYSDATE, HIREDATE)/12) from emp;

Output:

ENAME	TRUNC(MONTHS_BETWEEN(sysdate, hiredate)/12)
ALLEN	24
FORD	38
BLAKE	32
MARTIN	38
TURNER	35
ADAMS	38

④ LAST_DAY

* LAST_DAY function displays last date value for specified month in input date.

Syntax:

LAST_DAY (input date)

Eg: (i) SQL> Select LAST_DAY ('01-FEB-19') from dual;

Output:

LAST_DAY
28

(ii) SQL> Select LAST_DAY ('01-FEB-12') from dual;

Output:

LAST_DAY
29

⑤ NEXT_DAY

* NEXT_DAY function displays next date value for specified day and month in input date.

Syntax:

NEXT_DAY (input date , 'DAY')

Eg:

(i) SQL> Select NEXT_DAY (sysdate , 'SUNDAY') from dual;

output:

NEXT_DAY
01-DEC-19

(ii) SQL> Select NEXT_DAY (SYSDATE , 'THURSDAY') from dual;

output:

NEXT_DAY
05-DEC-19

(iii) SQL> Select NEXT_DAY (SYSDATE , 'FRIDAY') from dual;

output:

NEXT_DAY
29-NOV-19

[*] Arithmetic with date :-

* Add or Substrate a number to or from a date for a Resultant date value.

Eg:

(i) SQL> Select SYSDATE -12 from dual; ⟶

SYSDATE -12
16-NOV-19

(ii) SQL> Select SYSDATE - 365 from dual; ⟶

SYSDATE -365
27-NOV-18

(iii) SQL> Select 10 + 20 from dual; // 30

(iv) SQL> Select '10' + 20 from dual; // 30

↳ implicit Conversion from String to number

29/11/2019

⊛ Conversion Functions:

* It is used to convert one date type into another date type.
* There are 2 types of Conversion functions.

a) Implicit Conversion Function

* Type of Conversion without using functions.

Ex: Select 10+'20' from dual; // 30

b) Explicit Conversion Function

* Type of Conversion by using function.

Ex: Select 10 + ASCII('A') from dual; // 75

① TO_CHAR

* Converts input date value to character type.

Syntax:

TO_CHAR(date, 'FORMAT MODEL')

* The format model: It must enclose with single quotation mark
* Elements of Date format:

YYYY → Full Year in number
YEAR → Year Spelled out.
MM → Two digit value of month
MON → Three letter abbreviation of month
MONTH → Full Name of month
DY → Three letter abbreviation of day.
DAY → Full name of day.
DD → Numeric day of month.

Eg:

(i) SQL> Select 10 + '20' from dual; Output:

10 + '20'
30

(ii) SQL> Select 10 + A from dual; → 'A' Invalid Identifier

(iii) SQL> Select 10 + ASCII ('A') from dual;

Output:

10 + ASCII ('A')
75

(iv) SQL> Select 10 + ASCII ('a') from dual;

Output:

10 + ASCII ('a')
107

Ex: (i) SQL> Select TO_CHAR (SYSDATE, 'DD MM YYYY') from dual;

Output:

TO_CHAR (SYSDATE, 'DD MM YYYY')
29 11 2019

(ii) SQL> Select TO_CHAR (SYSDATE, 'DAY MON YYYY') from dual;

Output:

TO_CHAR (SYSDATE, 'DAY MON YYYY')
FRIDAY NOV 2019

(iii) SQL> Select TO_CHAR (SYSDATE, 'DY MONTH YEAR') from dual;

Output:

TO_CHAR (SYSDATE, 'DY MONTH YEAR')
FRI NOVEMBER TWENTY NINTEEN

⊛ General Function :

* It works w.r.t value / column irrespective of datatype.

① NVL

* Converts a null to actual values

Syntax:

NVL(arg1, arg2)

arg1 → column [containing null value]
arg2 → direct value / column [without null value]

② NVL2

* Converts a null to actual values

Syntax:

NVL2(arg1, arg2, arg3)

arg1 → column [containing null value]
arg2 → direct value / column [with / without null value]
arg3 → direct value / column [without null value]

* If arg1 is null it returns arg3
* If arg1 is not null then returns arg2

NVL()
* If arg1 is null it returns arg2
* If arg1 is not null then it returns itself.

Eg:

(i) SQL> Select comm, NVL(comm, 1000) from emp;

Output:

COMM	NVL(COMM, 1000)
1500	1500
	1000
	1000
300	300
600	600
	1000
0	0
;	;

(ii) SQL> Select comm, NVL(comm, sal) from emp;

Output:

COMM	NVL(COMM, SAL)
1500	1500
	600
	800
300	300
600	600
	2850
0	0

(iii) SQL> Select comm, NVL2(comm, comm, sal) from emp;

Output:

COMM	NVL2(COMM, COMM, SAL)
1500	1500
	600
	800
300	300
600	600
	2850
0	0

* User Input

* dynamically accepting values with the help of '&' (Ampasand)

```
SQL> Select * from emp
     where deptno = &n;
```

Output: Enter value for n : 30

where deptno = &n → old

where deptno = 30 → new

EMPNO	ENAME	JOB	HIREDATE	MGR	DEPTNO

---

④ Transaction Control Language

⊛ ROLLBACK

* It Undo's the previous changes.

⊛ COMMIT

* It is used to Save the changes.

⊛ SAVEPOINT

* It is used to create a check point in the SQL. so that we can rollback to a Specific state [Check point].

Syntax:

```
SAVEPOINT NAME;
```

Ex:

(i)>COMMIT ;

(ii)>ROLLBACK ;

(iii)>SAVEPOINT A;

(iv) SQL> SAVEPOINT B;

SQL> DELETE FROM emp
WHERE empno = 9999;

SQL> ROLLBACK TO SAVEPOINT B;

ANY (or) ALL

* It is a multi-valued operator which takes single LHS & multiple RHS value.
* It is prefered when the condition is w.r.t less than (or) Greater than.

Q> display employee details who are earning salary more than any one of manager?

```
SQL> Select * from emp
     where sal > (select MIN(sal)
                  from emp
                  where job = 'MANAGER');
```

By Using Any in the above query

```
SQL> Select * from emp
     where sal > ANY (select sal from emp
                      where job = 'MANAGER');
```

Q> display employee details who is earning salary more than all the managers?

```
SQL> Select * from emp
     where sal > (select MAX(sal) from emp
                  where job = 'MANAGER');
```

By Using ALL in the above query

```
SQL> Select * from emp
     where sal > ALL (select sal
                      from emp
                      where job = 'MANAGER');
```

---

⑤ Data Control Language

⊛ GRANT

* It is used to provide access permission from 1 user to another user.

Syntax:

```
GRANT PERMISSION ON TABLE_NAME
TO USER_NAME;
```

Note: * We can grant multiple permission at a time but only for 1 user at a time.

⊛ REVOKE

* It is used to take back permission that has been granted.

Syntax:

```
REVOKE PERMISSION ON TABLE_NAME
FROM USER_NAME;
```

Note: ★ The revoke has to be done where the permissions are granted.

★ Same USER has to grant and revoke permissions.

Eg:

```
SQL> GRANT SELECT, UPDATE ON EMP TO HR;
Grant Succeeded
SQL> CONNECT HR;
Enter user password : tiger ⇒ CONNECTED
SQL> SELECT * FROM SCOTT.EMP;
```

→ Printing Emp table in HR User

```
SQL> CONNECT SCOTT;
Enter user password : tiger ⇒ CONNECTED
SQL> REVOKE SELECT, UPDATE ON EMP FROM HR;
revoke Succeeded
```

Eg:

```
SQL> CONNECT HR;
SQL> GRANT SELECT ON REGIONS TO SCOTT;
SQL> CONNECT SCOTT;
SQL> SELECT * FROM HR.REGIONS;
SQL> CONNECT HR;
SQL> REVOKE SELECT ON REGIONS FROM SCOTT;
```

Normalization

30/11/2019

* It is the process of reducing redundancy [repeated / duplicated data] and dependancy in the database.

(or)

* Simplifying complex into a simple smaller table is known as Normalization.
* E.F Codd is the father of Normalization.
* A table is decided as simple as complex not based on the no of columns present in the table.

* Normalization forms

1NF
2NF
3NF
4NF
5NF
6NF

→ 3.5NF [BCNF = Boyce Code Normalization Form]

Bike-id	Name	Gender	Ph-no	Bike-Name
11	Suraj	M	9119114314	Pulsar 220 Apache 180
12	Sachin	M	8664435295	Royal Enfield TVS Excel
13	Ravi	M	7643814992	FZ 25 Duke 150
14	Anu	F	9434791986	Fasino Dio
15	Ajith	M	9047377200	R15 JAWA

* The above table is complex table, because a single cell contains multiple values.
* The Extra values that is present inside the cell should be added as new row.

⊛ 1NF

* Each and Every cell in a table should contain Single value.

⊛ 2NF

* Rules for 2NF
  - → It should obey 1st Normal form [1NF].
  - → Any of the column should be declared as primary key.
  - → There should be complete dependency.
  - → There should not exists partial dependency.

Note: ★ **Complete Dependency**: All the non-primary key columns should be dependent on primary key columns.

★ **Partial Dependency**: If the non-primary key columns are dependent on non-primary key column then it is partial dependency.

⊛ 3NF

* Rules for 3NF

→ The table should obey 1NF, 2NF.

→ There should be no transitive dependency.

Note: * Pictorial Representation of tables is known as 'Schema'.
