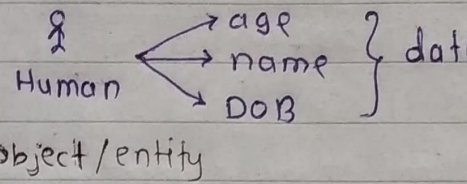
SQL Structured Query Language

* Data :- Data is a row fact which discribe the attribute of an entity. Piece of a information

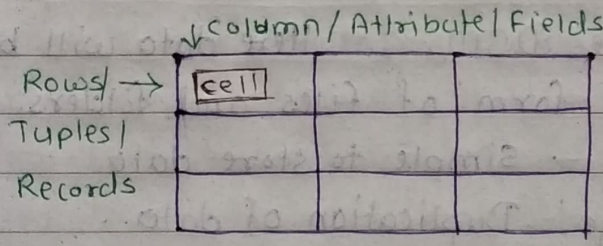


Data ⇒ Row facts
Information ⇒ meaningfull data / processed data

* Information :- Processed data is called information
eg. Qspider is a slw training institute

Data is classified into two types ⇒
① Structured data ② Unstructured data

① Structured Data :- Data which is arranged in a structured manner / way.
eg. Data stored in a table.



② Unstructured Data :- Data which is not arranged in a specific structure.
eg. Data store in PDF, TextFile, word file.

* Database :- Database is a plare where we can store the data in organised way for data retrival. (centralized Location).

Page:
Date: / /

* DBMS :- Database Management System.
It is a system which is used to interact with the database using query language for data manipulation.

CRUD operation

C → Create
R → Retrive / Read
U → Update
D → Delete

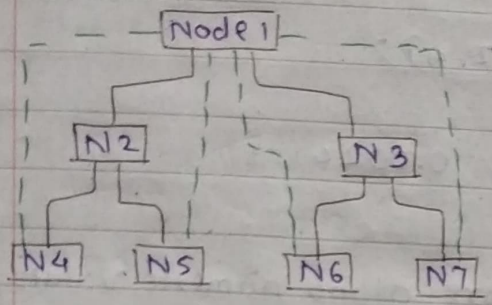
* Data Manupulation :-
Create , Insert , update, delete

DBMS is classified into four types :-
① Flat File DBMS
② Heirarical DBMS
③ Network DBMS
④ RDBMS

① Flat file DBMS :- Here data will be stored in the form of files and folders.
Advantage - Simple to store data
Disadvantage - Duplication of data.
eg. Data store on drive of a computer.

② Hirarical DBMS :- Here data will be stored in the form of parent-child relation or in a tree structure.
Advantage - Data duplication not allowed
Disadvantage - Retrival time consuming.

③ Network DBMS :- Here data will be stored in a n/w structure. All the nodes (data) are interlinked with another data.



Advantages - Retrival data easy
Disadvantage - Complex

④ RDBMS :- Here data will be stored in the form of rows and columns. i.e. in a table structure.

- In RDBMS to perform some operations it should be done by using their relations.

* Data Intigrity :- It is used to get accuracy, correctness and consistancy of the column of the table and it can be achieved by datatype and constrints.

1) Datatypes :- It restricts particular type of data to be store in the column of a table.

Types ⇒ (A) Number — int, float

(B) char

(C) Varchar

(D) Date

(A) Number :- used to store numric values/data.

eg. 1) ID Number (2); -99 to 99

We can store integers as well as float values.

case 2) ID Number (4,2);

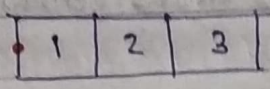
| 9 | 9 | 9 | 9 |

99.99

Decimal

Page:
Date: / /

case 3) ID Number (3,3);



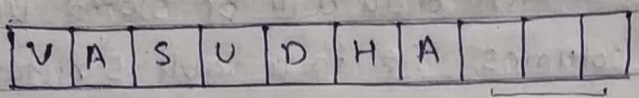
0.123

case 4) ID Number (2,4);

0.0 0 2 4 0.0024

(B) Char :- Used to store alpha numeric values/data

eg. Name char (10);



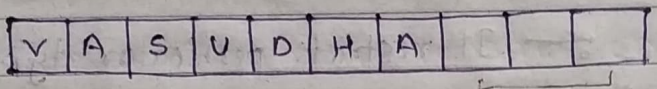
wasted memory block

Size of char is 2000

(C) Varchar :- Varchar is used to store alpha numerical values/data.

eg. Name varchar (10);

we can reuse these memory block



Reuse/Free memory space

Size of varchar is 4000

⇒ Difference between Char & varchar

Char	Varchar
1) It's fixed length memory allocation	1) It's variable length memory allocation
2) Unused space will be wasted	2) Unused spaces will be release/free
3) In unused spaces blank space will be store	3) In unused space null will be stored.
4) Size 2000	4) Size 4000 (varchar & varchar2 are same)

Page
Date

(D) Date :- By default date formate is -

DD-MON-YY

* Constrains :-

Constrains are the conditions we apply on a column of a table.

following are the types of constrains -

1) Not null
2) Unique
3) Primary key
4) Foreign key
5) check
6) Default

Only on a colum name we can apply not an a table name.

① Not Null :-

1) Null ⇒

① Null is nothing, neither zero nor blank space.
② It will not occupy any memory space.
③ Null represents unknown value.
④ Any arithmetic operation performed with null it results in null only.
eg. 100 + null = null.
⑤ Two nulls are never same

ID	Name
1	A
2	
3	

← Two nulls never same

- Not null will insure some value should be present in a column of a table.
- values may be duplicate
- we can apply not null constrain on more than one columns of data.

Syntax :- column_name datatype Not null;

eg. ID number(2) Not null;

	ID	Name
Not Null →	1	A
	2	B
	3	C
Not allowed →	Null	D

Not null constraints will not allow null value to be inserted in a column of a table but can accept duplicate value.

② Unique key :-

- It will not allow duplicate value in a column of a table
- Unique column can take multiple null. but not duplicate.
- We can apply unique constriant on more than one column. (distingt value)

eg.

ID number(2) unique;

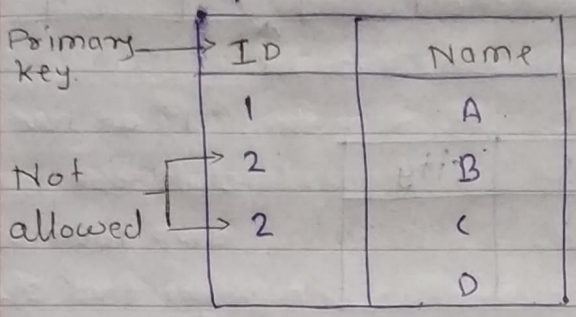
	ID	Name
unique →	1	A
	2	B
	4	C
	5	D
Duplicate not allowed		E
	5	F

Two nulls are allow because two null never same

③ Primary key :-

- It is a combination of unique + not null
- We can apply primary only on a single column of a table.
- Primary key is used to uniquely identify the records.
- Creation of primary key is not mandatory but it highly recomonded to create primary key.

eg. ID number (2) primary key;



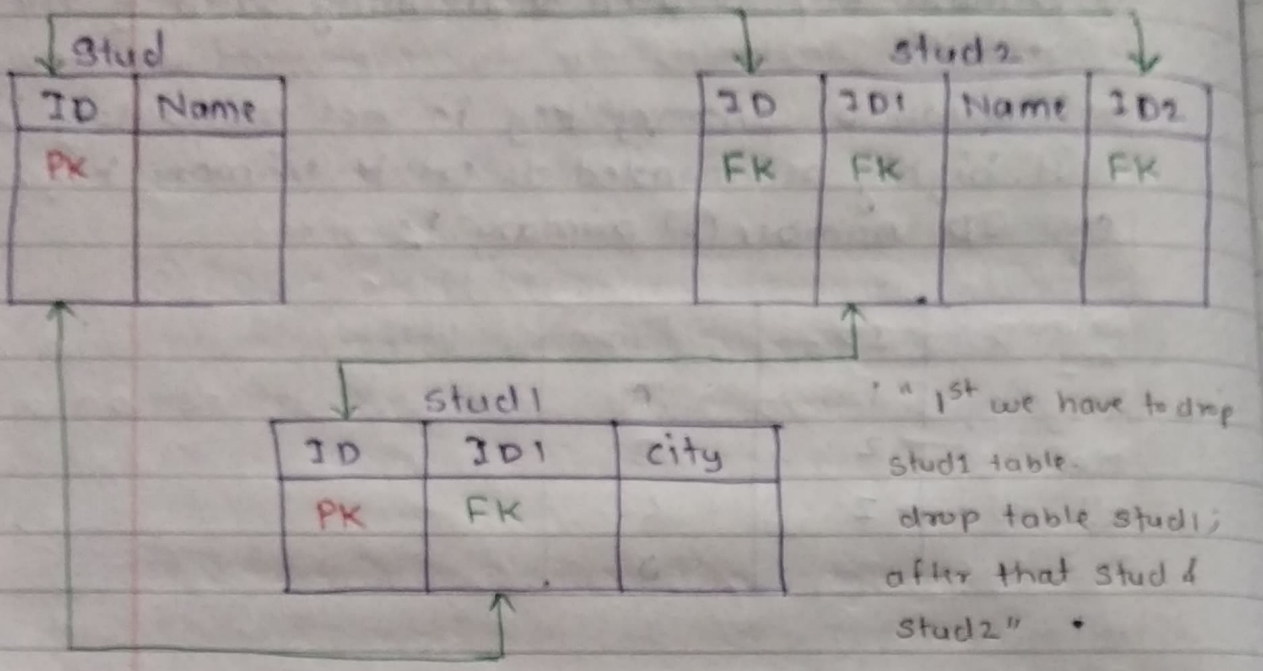
④ Foreign Key :-

- It is used to create relation between two tables.
- Foreign key is also called as referencibily intigrety constraints
- Foreign key is created on child table.
- To create a foreign key master table should have primary key defined on the common columns of a master table.
- We can have more than one foreign key
- Foreign key is a combination of Null & duplicate.

```
create table stud (ID number (2) primary key,
                   Name varchar (10));

create table stud1 (ID number (2) primary key,
    ID1 number (2) references stud (ID),
    city varchar (10));
```

create table stud2 (ID number(2) references stud(ID), ID1 number(2) references stud1(ID), Name varchar(10), ID2 number(2) references stud(ID));



⑤ check :-

- check is a user defined constraint. we can create for extra validation.
- It will not allow invalid data to be get inserted which not satisfied check constraint.

eg. create table stud(ID number(2), marks number (3) check (marks >=35 AND & 99 marks <=99));

stud

ID	Marks
1	35
2	99
3	75
4	34

← marks >=35 & <=99

← Not allow -

⑥ Default :-

- It is a user define constrain. If user will not provide any value for a column then default value can be consider / set.
- If user want to provide certain value user can provide the default value.

eg. create table stud (ID number (2), city varchar (10) default ('PUNE'));

ID	city
1	Mumbai
2	PUNE ← By default it will add

Simple Commands ⇒

1] set linesize 120;
set pagesize 20;

2] select * from tab; // check all tables / number of tables in database

3] shift delete // clear screen.

4] desc emp; // describe table // check table details // all constrains, datatype // display table structure.

5] create table demo(ID number (2), Name varchar (10));

create ↓ Duplicate Table

create table demo1 As select * from demo;

6] ed To modify any query
/ Enter.

Page:
Date: / /

SQL Structured Query Language (1970 → IBM)

1) SQL is used to perform manupulation operation on data which is stored in structured formate
2) On unstructured data we can not apply SQL
3) SQL is a query language is not a programming language.

SQL Statements ⇒

① DDL - Data Defination Language
- (A) create
- (B) alter
- (C) drop
- (D) Truncate
- (E) Rename

② DML - Data Manupulation Language
- (A) insert
- (B) update
- (C) delete

③ DCL - Data control language
- (A) Grant
- (B) Revoke

④ DQL - Data Query Language
- (A) select

⑤ TCL - Transaction Control Language
- (A) Commit
- (B) Rollback
- (C) Save point

① DDL ⇒

Ⓐ Create ⇒ It is used to create a table, views in a database.

* How to create table? ⇒

Create table Table_Name;

create table Demo (ID number(2) primary key, Name varchar(10) Not null, Contact varchar(10) unique, City varchar(10) default ('PUNE'), marks number(2) check (marks >35 AND marks <=99));

ID	Name	Contact	City	Marks
PK	Not Null	Uniaue	Default	check

* How to create duplicate table? ⇒

create table Demo1 as select * from Demo;

ID	Name	Contact	City	Marks
	Not Null			

- In duplicate table all the records, all the columns and only Not null constraint will get copy.
- After creating a duplicate tables both tables are independant.

(5) Alter :- Alter is used to add column, to rename a column, to modify a structure of a column, To delete a column.

Case 1 : ADD a column ⇒

Syntax ⇒

```
alter table table-name add col-name
    datatype(size);
```

eg.

1> create table stud (id number(2));

2> desc table stud;

Name	Null?	Type
ID		number

3> alter table stud add name varchar(10);

4> desc table stud;

Name	Null?	Type
ID		number
Name		varchar

Case 2 : Modify ⇒ Is used to modify the structure of a table, we can change the datatype, also size of a column

Syntax ⇒ To modify size.

```
alter table table-name modify (col-name
    datatype (size));
```

Syntax ⇒ To modify datatype

```
alter table table-name modify (col-name
    newdatatype (size));
```

" column should be empty to modify column datatype "

Page:
Date: / /

Case 3 : Rename a column ⇒

Syntax ⇒

```
alter table table_name rename column
    old_col_name to new_col_name;
```

eg.

```
alter table stud rename column ID to ID1;
```

Case 4 : To drop a column ⇒

Syntax ⇒

```
alter table table_name drop column col_name;
```

Case 5: Add a constraint ⇒ Table should be empty.

Syntax ⇒

Primary Key

```
alter table table_name add constraint
    constraint_name constraint (column_name);
```

eg.

```
alter table stud add constraint pk primary key (id);
```

Syntax ⇒

Foreign Key

```
alter table table_name add constraint constraint_
 _name constraint (column_name) references
table_name (column_name);
```

eg.

demo

ID (PK)	Name

stud

ID (FK)	marks

1> alter table stud add constraint FK Foreign key (ID)
references demo (ID);

Unique ⇒

2> alter table stud add constraint UK unique (~~Name~~ Colume);

- ~~alter~~

c) Drop ⇒ Drop is used to drop the structure of the table and add records of table.

Syntax ⇒

```
drop table table_name ;
```

eg.

```
drop table stud1 ;
```

SQL> show recyclebin ;

all the tables after drop will moves to recyclebin.

To restore table from recyclebin ⇒

Syntax ⇒

```
Flashback table table_name to before drop;
```

To delete the table from recyclebin ⇒

Syntax ⇒

```
purge table table_name ;
```

"permanatly delete a table"

D) Truncate ⇒ Is used to delete or clear only recordss of the table., Structure of table remains same.

Syntax ⇒

```
truncate table table_name;
```

E) Rename ⇒ Is used to rename the table name.

Syntax ⇒

```
rename old_table_name to new_table_name;
```

eg.

```
rename stud to stud1;
```

Page:
Date: 3/3/21

② DML : Data Manupulation language

a) insert :- insert is used to insert the records at new line.

Case1 : insert record in all the columns.

```
insert into table-name values (value1, value2);
```

Case 2 :

insert into specific column

```
insert into tablename (column-name) values (value1);
```

b) Update :- Update is used to update the records

Case1 : Update all records.

```
update table-name set column-name = value;
```

Case 2 : Update specific records

```
update table-name set column-name = value where condition;
```

Eg.

1) update stud set city = 'PUNE' where ID=1 OR ID=2 ;
2) update stud set city = 'PUNE' where ID<=2;
3) update stud set city = 'PUNE' where ID=1 OR Name = 'B';

1) create table stud (ID number(2), Name varchar(10));
2) insert into stud values (1, 'A');
3) insert into stud values (2, 'B');
4) alter table stud add city varchar(10);
5) update stud set city = 'PUNE';
6) select * from stud;

ID	Name	City
1	A	PUNE
2	B	PUNE

stud

← alter

update

c> delete : Delete is used to delete all the records or specific records from table.

Case1 : To delete all records.

```
1> delete from table-name;   OR
2> delete table-name;
```

Case2 : To delete specific records

```
1> delete from table-name where condition;   OR
2> delete table-name where condition;
```

4/3/21 ③* TCL - Transaction Control Language

1) Commit
2) Rollback
3) Savepoint

① Commit -

Commit is used to save DML changes permanantly.

Syntax -

```
commit;
```

② Rollback - Is used to undo DML changes.

Syntax -

```
Rollback;
```

eg. 1) > create
> insert
> update
> delete
} All statements will be Rollbacked
> Rollback

eg 2) > create
> insert → DML - insert will be saved automatically because of alter (DDL)
> Alter → DDL Also save
> update } Rollbacked.
> delete }
> Rollback

Note ① DDL statements are auto commited, we need not to commit them.

② After commit their is no use of Rollback.

③ Savepoint - It act's as a break point.
- ~~Satem~~ Statements before savepoint nighter commited noe rollbacked.

Syntax:

```
Savepoint savepoint_name;
```

eg ①
> create
> insert
> update
> Savepoint x;
> delete
> Rollback to x; ← only delete will be rollbacked
> Rollback; ← All DML will be rollbacked.

No use of savepoint.

eg. ② > create
> insert
> Savepoint a;
> alter
> update
> Rollback to a; ← error
> Rollback;

error → Savepoint 'x' never established.

Savepoint is save records for temporary.

Page:
Date: / /

(4) DQL - Data Query language -

Select - Is used to display the records

Syntax -

```
Select * from table name;
select column-name from table name;
Selet column-name from table name where cond^n;
```

5/3/21 * Dual :-

1) Dual is a dummy table which is used to perform some operations which are not present on any existing table.
2) By default it have one row & one column.

* ALIA's :

1) ALIA's is a additional name provided to the columns.
2) ALIA's are applicable for single execution.
3) We have to provide ALIA's name after column name.

eg. > Select sal+comm As total_sal from emp;

TOTAL_SAL
1900
1700
—
2650

4) 'AS' keyword is not mandatory

eg > select sal+comm total_sal from emp;

TOTAL_SAL
1900
1700
—
2650

eg. Select sal+comm "TOTAL SAL" from emp;

TOTAL SAL
1900
1700
—

eg. Select sal + comm "Total Sal" from emp;

Total Sal
1900
1700
-
⋮

eg. select empname, sal + comm from emp where sal+comm > 1500;

emp-name	sal + comm
A	1900
-	⋮
-	
⋮	

* SQL Operators :-

1) Arithmetic Operator - -, +, *, /
2) Logical - AND, OR, NOT
3) Relational - <, >, ≤, ≥, !=, <>
4) Special - IS, IN, Between, like
5) Concat - ||

① Arithmetic Operator -

a) Write a query to Display (WAQTD) incremented sal of all the emp by RS 100.
b) WAQTD decrement the commition of all the emp by RS 50.
c) Display anual sal of all the emp.
d) Display quardarly sal of all the emp's

a) ⇒ SQL > Select empname, sal+100 "Incre sal" from emp;
b) SQL > select comm-50 "Decremental sal." from emp;
c) SQL > select empname, sal*12 from emp;
d) SQL > select ename, sal, (sal*12)/4 "Quat.sal" from emp;
e) increment the sal of all emp by 10%

sal + sal*0.1
sal*11/10

e> Select ename, sal, * sal + (sal* 10/100) "10%." ~~emp~~ from emp;

② Relational Operator -

a) WAQTD emp who's sal is greter than 1250.
b) WAQTD emp who's sal is less than 3000
c) WAQTD emp who are not working in dept 30.
d) Display all the salesman (JOB)
e) Display all the emp who earn at least 1250.
d) Display all the emp who earn at most 3000.
f) Display all the emp who are not cleark.

a) ⇒ Select ename from emp where sal > 1250;
b) ⇒ select ename from emp where sal < 3000;
c) ⇒ Select ename from emp where deptno != 30;
d) ⇒ Select ename, job from emp where job = 'salesman';
e) ⇒ Select ename from emp where sal >= 1250;
d) ⇒ Select ename from emp where sal <= 3000;
e) ⇒ select ename from emp where job != 'clerk';
(job <> 'clerk';)

③ Logical Operator -

a) Display all the employees who are working as a clerk or manager.
b) DATE who are salesmans and earning sal > 1250.
c) DATE who are from dept 10 & 20.
d) DATE who are not analist
e) DATE who are analist & earning sal > 2500 or < 5000

a ⇒ Select * from emp where job = 'CLERK' OR job = 'MANAGER';
b ⇒ Select * from emp where job = 'salesman' AND sal > 1250;
c ⇒ Select * from emp where deptno = 10 OR deptno = 20;
d ⇒ Select * from emp where not job = 'ANALIST';

e⇒ select * from emp where job = 'Analist' AND sal>2500 or sal <5000;

4)* Special Operator :- is, In, Between, like

i) Is : - It is used to compare null.
We cannot compare any other value with is.

Syntax:

```
select */col_name from table_name
where col_name is null / is not null;
```

Q. Write a query to display employees who are not earning any commision.
⇒ select * from emp where commision is null;

Q WAQTD who are earning some commision.
⇒ select * from emp where commision is not null;

ii) In : Instead of using multiple or (||) we can use single in operator.
In operator works as OR (||) operator.
If we want single LHS with multiple RHS we use in operator.

Syntax:

```
select * / col_name from table_name where
col_name in (value1, value2---);
```

Q WAQTD the emp who are working as a clerk or analyst.
⇒ select * from emp where job in ('clerk', 'analyst');

Q WAQTD the emp who are not working as a cleark or analyst.

⇒ select * from emp where job not in ('cleark', 'analyst');

iii) Between : Between operator selects the value within a given range.

Syntax :

```
Select * / col-name from table-name
where col-name between value1 and
value2 ;
```

Q WAQTD the emp whose salary between 1250 and 3000.

⇒ select * from emp where sal between 1250 and 3000;

Q WAQTD emp whose sal not between 1250 and 3000

⇒ select * from emp where sal not between 1250 and 3000;

Q WAQTD emp whose name in between 'A' to 's';

⇒ select * ~~emp~~ from emp where ename between 'A' and 's';

iv) like : Like operator is used for pattern matching. we use two wildcard characters for pattern matching (like spe char)

Following are the two like char

a) % = It matches with 0, 1, or n number of characters.

b) _ = It matches with single character

escape : escape char are used in pattern string to indicate that any wildcard char that occurs after escape chare, in a pattern string should be treated as a regular char.

Q WAPTD the emp whose name starts with A

⇒ select * from emp where ename like 'A%'

pattern string

Note i) Name ends with A = ename like '%A'

ii) Name starts with A = ename like 'A%A' and ends with A

iii) Whose name contains A anywhere in the name = ename like '%A%'

iv) Whose name contains exactly 4 char = ename like '____'

v) Name contains A at 2nd position = ename like '_A%'

vi) Name contains A at 2nd last position = ename like '%A_'

vii) Whose name contains 2 A = ename like '%A%A%'

Page:
Date: / /

Q 1 list all the emp who don't have any reporting manager?

⇒ select * from emp where mgr is null;

Q 2 = list all the salesmans in dept 30 and having salary grater than 1500.

⇒ select * from emp where job='salesman' and Deptno=30 and sal > 1500;

Q 3 list all the emp whose name starts with 'S' or 'A'

⇒ select * from emp where ename like 'S%' or ename like 'A%';

Q 4 list all the emp whose name does not starts with 'ES' or 'R'

⇒ select * from emp where ename not like 'ES%' or ename not like 'R%';

Q 5 list all the emp with anual salary except those who are working in Dept 30.

⇒ ~~Select * from emp~~

select sal*12 As Anucal_Sal from emp where dept is not 30;

Q 6 Display all the emp who are joined after year 81.

⇒ Select * from emp where hiredate >= '31-Dec-1981';

Q 7 Display all the emp who are joined in Feb.

⇒ Select * from emp where hiredate like '%FEB%';

Q 8 list the emp who are not working as a manager & clerk in dept 10 & 20 with the salary in a range of 1000 to 3000.

⇒ Select * from emp where job not in ('Manager','clerk') and Deptno in((0,20) And sal between 1000 And 3000;

Q9 List all the emp whose salary not in a range of 1000 to 2000 and working in dept 10,20 or 30 except all salesmans.

⇒ Select * from emp where sal not between 1000 and 2000 and deptno in(10,20,30) and job not in('salesman');

Q 10 Display all the emp whose job starts with man..

⇒ Select * from emp where job like 'MAN%.';

Q11 List the emp who are hired after 82 and before 87

⇒ Select * from emp

Q12 Display the emp whose commision is greater than sal

⇒ Select * from emp where comm >sal;

Q13 Display the emp whose annual sal ends with 0.

⇒ Select ename, sal, sal*12 'ANN.SAL" From emp where sal*12 like '%0';

Q.14 Display the emp whose name starts with vowels

⇒ Select * from emp where ename like 'A%.' or ename like 'E%.' or ename like 'I%.' or ename like 'O%.' or ename like 'U%.';

Q.15 Display the emp who are hired in nov & Dec

⇒ Select x from emp where Hiredate like '%.nov%.' or hiredate like '%.Dec%.';

Page:
Date: / /

Q 16 Display the emp whose Job contain manager.

⇒ select * from emp where job ~~is 'manager'~~ like '%MAN%';

Q.17 Display all the emp who are having reporting manager in dept 10 also g with 10% high in the salary.

⇒ ~~Select * from emp where~~ select ename, sal, sal+sal*0.1 "Hike 10%" from emp where deptno = 10 and mgr is not null;

Q 18 Display the emp whose name having exactly two L

⇒ select * from emp where ename like '%L%L%';

Q.19 Display the name and desgnation of all the emp having reporting manager and also their names starts with 'S'.

⇒ select ename, job from emp where mgr is not null and ename like '%S';

Q.20 Display all the emp who are having reporting manager in dept 10.

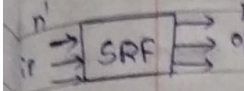
⇒ select * from emp where deptno = 10 and mgr is not null;

Page
Date 09/03/21

* SQL Functions :-

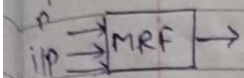
SQL functions are categorised into two types -

① Single Row function -



Here we are providing n number of inputs, & for every input we will get seprate output.

② Multi Row function -



To multi row function we are providing n number of inputs and we will get only one output for multiple inputs.

* Multi Row Function - (Agregate function / Group function)

1) Multi row functions are.

min()
max()
avg()
sum()
count()

② Min() :- It returns minimum value or lowest character of given column. Syntax.

```
Select min(col-name) from Table-name;
```

eg.

```
> Select min(sal), min(hiredate), min(ename) from emp;
```

MIN(SAL)	MIN(HireDate)	min(ENAME)
800	17-DEC-80	ADAMS

↑ ASCII value

```
> Select min(Sal + comm) from emp;
```

min(sal + comm)
1500

Page:
Date:

(b) Max() It returns maximum value or largest character

Syntax:

```
select max(col-name), max(col-name) from
table_name;
```

eg.

```
> select max(SAL), max(Hiredate), max(Ename), max(comm)
  from emp;
```

max(SAL)	max(Hiredate)	max(Ename)	max(comm)
5000			

10/3/21

* Average :- It returns average value of specified colum

Syntax:

```
select avg(column_name) from Table_name;
```

eg.

```
select avg(sal) from emp;
select avg(mgr) from emp;
select avg(sal + comm) from emp;
```

* Sum : It returns summation of all the values in given column.

Syntax:

```
select sum(column_name) from Table_name;
```

eg.

```
select sum(sal) from emp;
```

* Count : It returns number of rows in a given column.

Note Count function will not count null values.

Syntax:

```
select count(column_name) from table_name;
```

eg.

1> Select count(empno) from emp;

→ 14

2) select count(comm) from emp;
→ 4

3) select count(*) from emp;
→ 14

4) select count(*) - count(comm) from emp;
→ 10

5) select count(*) from emp; where comm is null;
→ 10

* distinct : It is a keyword. It is used to select unique record from the column.

Syntax :

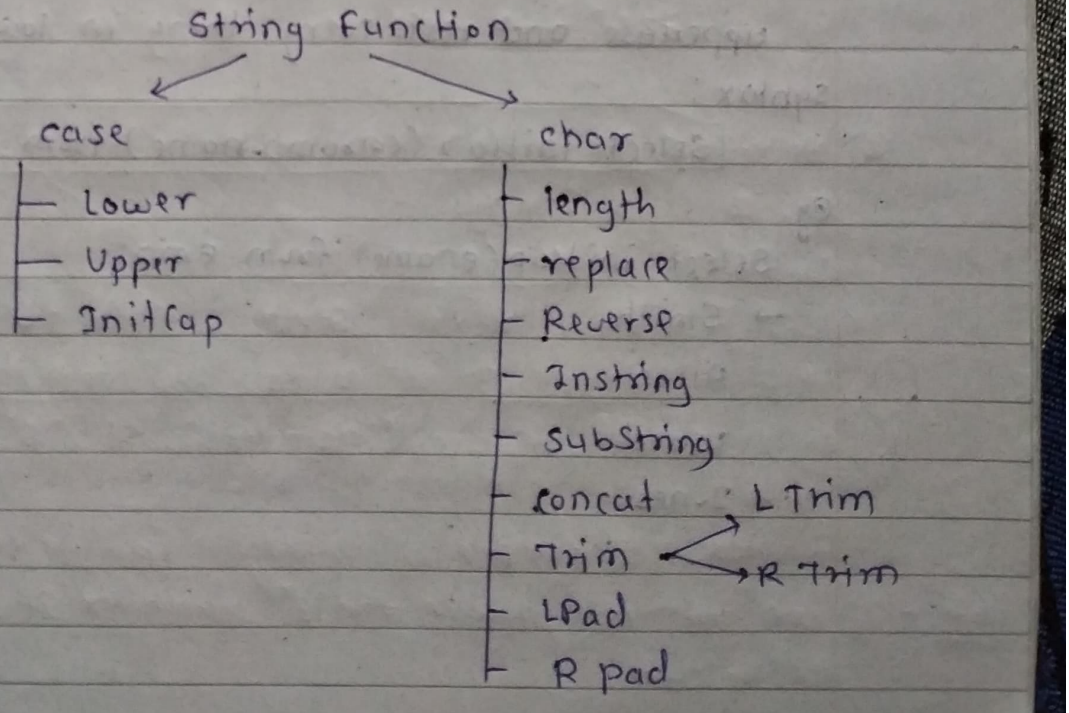
```
select distinct column_name from Table_name;
```

eg. 1) select distinct deptno from emp;
→ 30
20
10

nested function i.e. function under function

2) ~~select distinct de~~
select count(distinct (deptno)) from emp;
→ 3

* Single Row function :-



Case manipulation function :-

① Lower :- It returns character of string in a lower case letter.

Syntax :

```
Select Lower (column-name) from Table-name;
```

e.g.

> Select lower ('SQL') from dual;

→ sql

eg. update the smithname in lowercase in emp table..

→ update empl set ename = lower (ename) where ename = "SMITH";

② Upper :- It is used to convert character of string in uppercase letter.

Syntax;

```
Select upper (column-name) from Table-name;
```

eg. Select upper ('sql') from dual;

→ SQL

③ InitCap :- It returns first character of string in uppercase and rest every thing in lowercase.

syntax;

```
Select initCap (column-name) from emp;
```

eg.

Select initcap (ename) from emp;

→ Smith

Allen

ward

Jones

eg.
Select lower (ename) "Lower", Upper(ename) "Upper", initcap (ename) "Initcap" from emp;

→

lower	Upper	Initcap
smith	SMITH	Smith
allen	ALLEN	Allen
ward	WARD	Ward

* Character Manipulation Function -

① length:- It returns number of character in a given string.

Syntax -

```
Select length (column_name) from table_name;
```

eg.
Select length (ename), ename from emp;

Q Write a query to display an employee whose name is of 5 character.

→ Select ename from emp where length (ename) = 5;

→ SMITH
ALLEN
JONES
BLACK

② Concat : It is used to join two or more strings or columns.

Syntax:

```
Select concat (column_name1, column_name2)
from emp;
```

Page:
Date: / /

eg. Select concat ('QSP', 'Hadapsar') from dual;
→ QSPHadapsar

i) Nested concat :-

Syntax:
```
select concat(column-name1, concat(column-
    name2, column-name3));
```

eg.
```
Select concat('QSP', concat('Pune', 'Hadapsar'));
```
→ QSPPuneHadapsar

11/3/21

③ Reverse : It reverse the characters of a String.

Syntax:
```
select reverse(column-name) from Table-name;
```

eg.
```
Select reverse(ename) from emp;
```
→ HTIMS (SMITH)
→ NELLA (ALLEN)

If we want to reverse of number i.e 321 → 123 (Reverse)
then we should pass with single cot i.e.
'321' ⟶ 123

④ Replace : It is used to replace the character of a String.

Syntax:
```
Select replace(String, 'char to be replace',
'char to be replace by') from table-name;
```

eg. 1> select replace ('qspiders', 's', '$') from dual;
→
q$pider$

Note - If we will not provide the 3rd parameter it will acts as remove.

eg. 2> Select replace ('qspiders', 'qspiders', 'qsp') from dual;

→ qsp

eg. 3) select replace ('ekata', 'ekta', 'e') from dual;

→ ekata

eg. 4) select replace ('qspiders', 'pid', '*') from dual;

→ qs*ers

eg. 5) selecet replace ('qspiders', 's') from dual;

3rd parameter not provided so it will acts as remover

→ qpider

⑤ Instring: It is used to identify position of character in given string.

Syntax:

```
Instring ('string', 'char', start position,
          no. of occurances);
```

- Start position & number of occurances are optional
- By default start position will be 1 & number of occurances also.
- If start position is negative number it will start search from end.
- Start position and number of occurances cannot be zero (0);
- We cannot provide number of occurances as a negative number.

eg.

1) select instr ('qspiders', 's', 1, 1) from dual;
→ 2

2) select instr ('qspiders', 's', 1, 2) from dual;
→ 8

3) select instr ('qspiders', 's', 3, 1) from dual;
→ 8

4) select instr ('qspiders', 's', 3, 2) from dual;
→ 0

5) select instr ('qspiders', 'z', 1, 1) from dual;
→ 0

6) select instr ('qspiders', 's', -1, 1) from dual;
→ 8

7) select instr ('qspiders', 's', -1, 2) from dual;
→ 2

8) select instr ('qspiders', 's', -2, 1) from dual;
→ 2

9) select instr ('qspiders', 's') from dual;
→ 2

10) select instr ('qspiders', 's', 0, 1) from dual;
→ 0

11) select instr ('qspiders', 's', 0, 0)
→ error (Number of occurance cant be zero)

a) Substring :- It display substring from main string.

Syntax:

substr (string, start position, no. of char);

- If start position is negative number then it will start from end.
- number of characters are option
- By default it will display the string till end

eg. select ename, substr (ename, 1, 3), substr (ename, 2, 2) Substr (ename, 3) from dual;

ENAME	SUB	SU	SUBSTR
1) SMITH	SMI	MI	~~LEN~~ ITH
2) ALLEN	ALL	LL	LEN
3) WARD	WAR	AR	RD

eg. select ename, substr (ename, -3, 3), substr (ename, -3, 2) substr (ename, -1, 2), substr (ename, -1, 1) from emp;

→

ENAME	SUB	SU	S	S
SMITH	~~SMI~~ITH	IT	H	H
ALLEN	LEN	LE	N	N
WARD	ARD	AR	D	D

eg. 1) select substr ('@spiders', 0, 1) from dual;

→ @

2) select substr ('@spiders', 1, 1) from dual;

→ @

3) select substr ('@spiders', 1, 0) from dual;

→ no output.

4) select substr('@spiders', 1, -1) from dual;
→ & no output
5) select substr('@spiders', 4, 1) from dual;
→ $ no output

12/3/21

5 Display name of emp who's job starts with man:
→ select ename from emp where job like 'man%';
select ename, job from emp where substr(job, 1, 3) = 'MAN';

Ename	JOB
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER

⑥ Trim :- Is used to ~~trim~~ the character from given string.
Case 1 ; To delete start char
Syntax:
select trim (leading 'char' from 'string')

Case 2 : To delete end char
Syntax
trim (trailing 'char' from 'string')

Case 3 ; To delete start & end char
Syntax:
trim (both 'char' from 'string')

"To trim middle char we ~~use~~ make use of replace queary."

eg.

1) select trim(leading 'a' from 'akshaya') from dual;

→ TRIM(L

kshaya

2) select trim('a' from 'akshaya') from dual;

→ TRIM(

kshay

3) select trim('a' from 'ekata') from dual;

→ TRIM

ekat

4) select trim('a' from 'akshaya') from dual;

→ TRIM

kshay

5) select trim(leading 'a' from 'akshya') from dual;

→ TRIM

kshaya

6) select trim(trailing 'a' from 'akshaya') from dual;

→ TRIM

akshay

7) select trim(both 'a' from 'akshya') from dual;

→ TRIM(

kshay

8) select trim(leading 'ak' from 'akshya') from dual;

↑

→ error → we can not take two char.

Page:
Date: / /

@ Replace a - @ and a - # in string 'Ekata'

Select ~~concat~~ replace ('Ekata', 'a', '@', replace
c

—

Ltrim : Used to ~~trim~~ trim the char from left side

Syntax :

ltrim('string', 'substring')

eg. 1) select ltrim('eaeaaaweaea', 'ae') from dual;

→ Ltrim
weaea

2) select ltrim('eaeaaaweaea', 'wae') from dual;

→ Ltrim

3) select ltrim('eaeaaaweaea', 'war') from dual;

Ltrim
eaeaaaweaea

It will trim substring upto new / ~~upper~~ chareactor found. It will starts from left side & check the 1st char present in substring or not.

Rtrim; Rtrim is wed to trim the char from right side.

Syntax :

rtrim ('string', 'substring')

eg. 1) select rtrim ('eaeaaaweaea', 'ae') from dual;

RTRIM
eaeaaaw

2) select rtrim('eaeaaaaweaea', 'aew') from dual;

→ R

3) select rtrim('eaeaaaaweaea', 'ew') from dual;

RTRIM

eaeaaaweaea.

4) Select rtrim (ltrim ('@spider', '@'), 'r') from dual;

→ Spide

spider

sider

* Rpad :- To upend the blank space.

eg. ***qsp***

→

Select rpad (lpad ('qsp', 6, '*'), 9, '*') from dual;

LPAD (RPAD

qsp

Syntax: rpad ('string', number of char, 'Padded char');

lpad ('string', number of char, 'Padded char')

eg. Select lpad ('qsp', 3, '*') from dual;

LPA

qsp

2) Select lpad ('qsp', 1, '*') from dual;

→ q

3) select lpad ('qsp', -6, '*') from dual;

→ —

4) select lpad ('qsp', 0, '*') from dual;

→ —

Page :
Date : / /

* Number functions -

ceil

floor

Round

Trunc

ABS

mod

Power

Sqrt

Page
Date 19/03/21

③ Round :-

Eg. 1> select round(123.456, 3) from dual; 123.456
② select round(123.456, 2) from dual; 123.46
③ select round(123.425, 1) from dual; 123.4
④ select round(123.425, 0) from dual; 123
⑤ select round(123.541, 0) from dual; 124
⑥ select round(123.425, -1) from dual; 120
⑦ select round(123.345, 1.5) from dual; 123.3
⑧ select round(123.456, -1.5) from dual; 120
⑨ select round(123.456, -2) from dual; 100
⑩ select round(123.425, -3) from dual; 0
⑪ select round(123.425, -4) from dual; 0
⑫ select round(555.555, -3) from dual; 1000
⑬ select round(555.555, -4) from dual; 0

4) Trunc (); It will goes on only deleting value, It will not roundup the value

① Select trunc(123.456, 3) from dual; 123.456
② select trunc(123.456, 2) from dual; 123.45
③ select trunc(123.456, 1) from dual; 123.4
④ select trunc(123.456, 0) from dual; 123

-ve number - remove digits before decimal
+ve number - After decimal how many digits you want

⑤ Select trunc(123.456, -1) from dual; 120

⑤ Absolute Value :- abs()
convert -ve number to +ve
remove unwanted zero

① select abs(10.01) from dual; 10.01
② select abs(-10.01) from dual; 10.01
③ select abs(-10.10) from dual; 10.1
④ select abs(10.0001) from dual; 10.0001
⑤ select abs(-01.10) from dual; 1.1

⑥ Power() :- It will return power of no

① select power(12,2) from dual; 144
② select power(15,2) from dual; 225

⑦ Squar sqrt(): It returns squaroot

① select sqrt(625) from dual; 25
② select sqrt(1000) from dual; 10

⑧ Mod() :

① select mod(12,4) from dual; 0
② select mod(12,3) from dual; 0
③ select mod(12,5) from dual; 2

* General Functions :-
① NUL (exp1, exp2) or (arg1, arg2)
② NUL2
③ decode