What is data?

The most simple answer: Data can be facts related to any object in consideration.

Ex: Age, Vide, othe data like whatsapp, fb, Insta, twitter.

What is database

Database is systematic collection of data which supports storing and manipulation of data in a easy way.

Ex: (i) Telephone Directory (ii) Power Generation (iii) Social media
- contacts.
- which house power generated details
- stories, images.

Why need a database

Database needed Because, Suppose data can be stored in a spread sheet, but if you add large chunks of data into the sheet, it might not work.

For instance: If you size of data increases into 1000's records, it will simply create a problem of maintaining and efficiency.

* Flexibility to update data as your convenience.
* multipk people also edit data at a some time.
* Security.

Database Management System (DBMS)

DBMS is a collection of programs which enables its user to access database, manipulate data, reporting / representation of data.

1960 — Charles Banchman's integrated data store (IDS) is considered the first DBMS in history.

SQL [structured query language]

SQL is the standard language for dealing with Relational Database which can be used fore to read, update, create and database Records.

Relational Database : A Relational data base is a type of database that stores and provides access to data points that are related to one other.

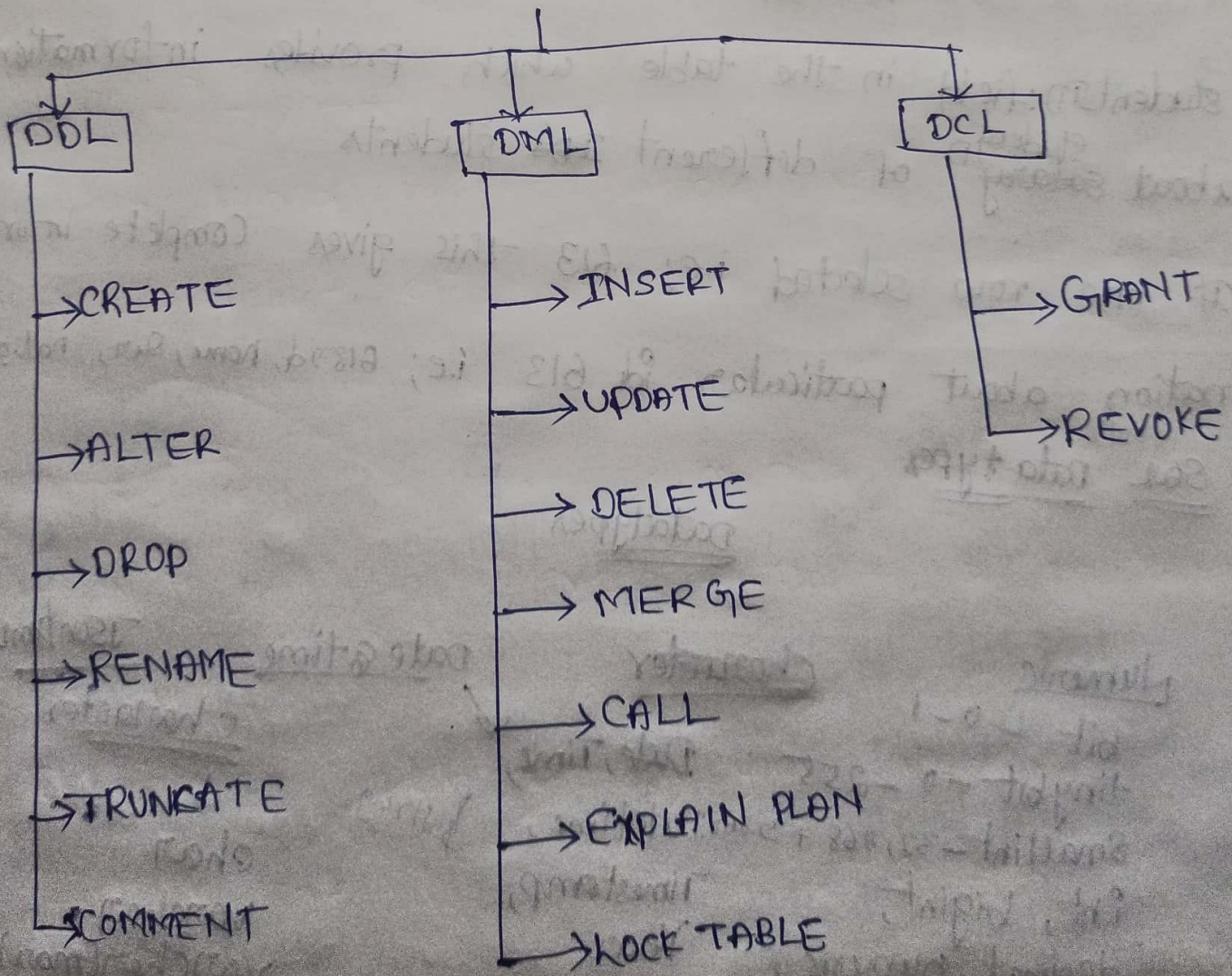
* In a Relational database, each row in the table is a "record" with a unique ID called the 'key'.
* The columns of the table hold attributes of data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Relation database Ex: MySQL, ORacle SQL, MSSQLServer, skybase.etc.

Applications of SQL

* used of Data definition Language (DDL) meaning you can independently create data base, define structure, & use.
* Data manipulation language: changing of data.
* Data control Language: It will protect your database against corruption and misuse.

SQL commands

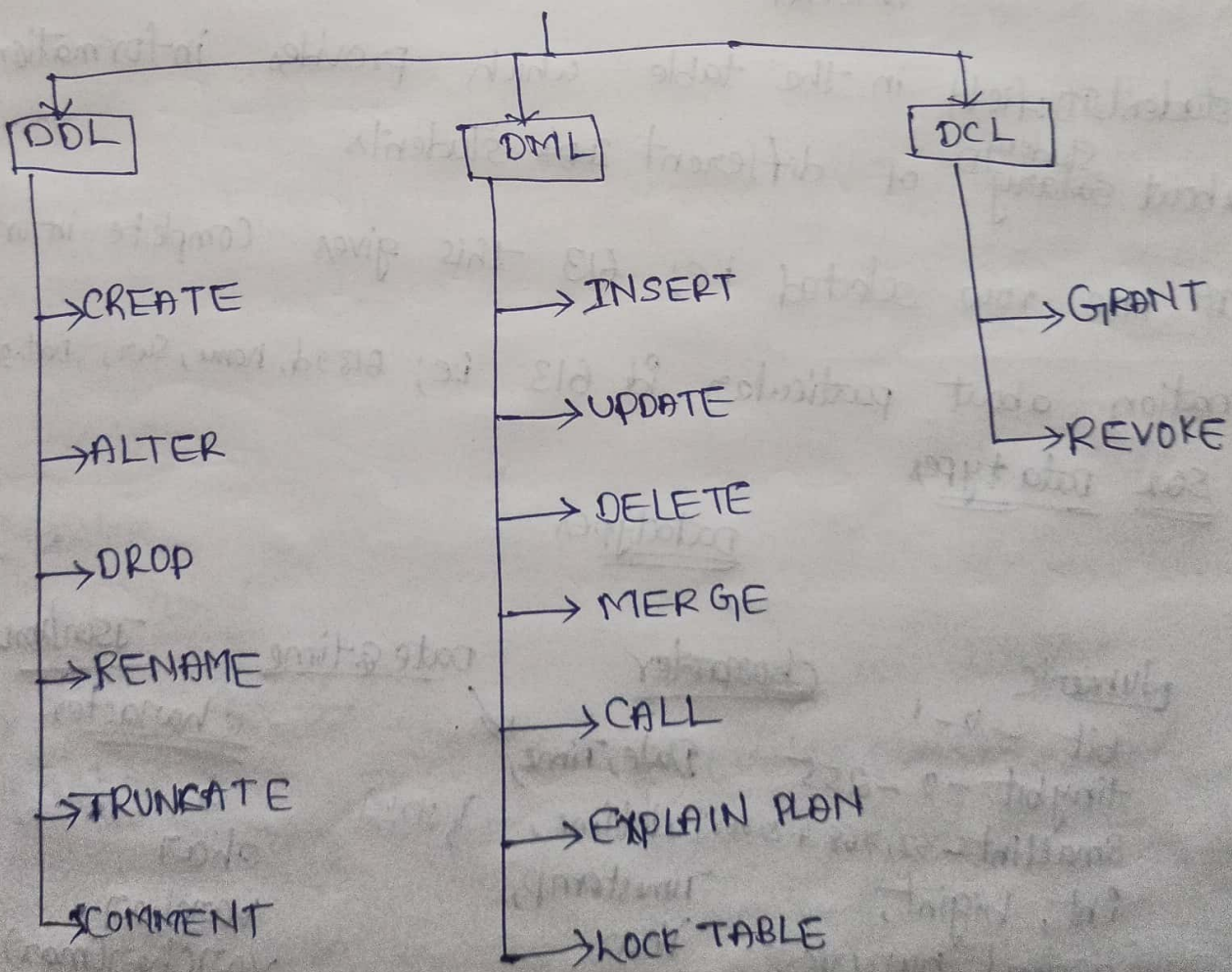


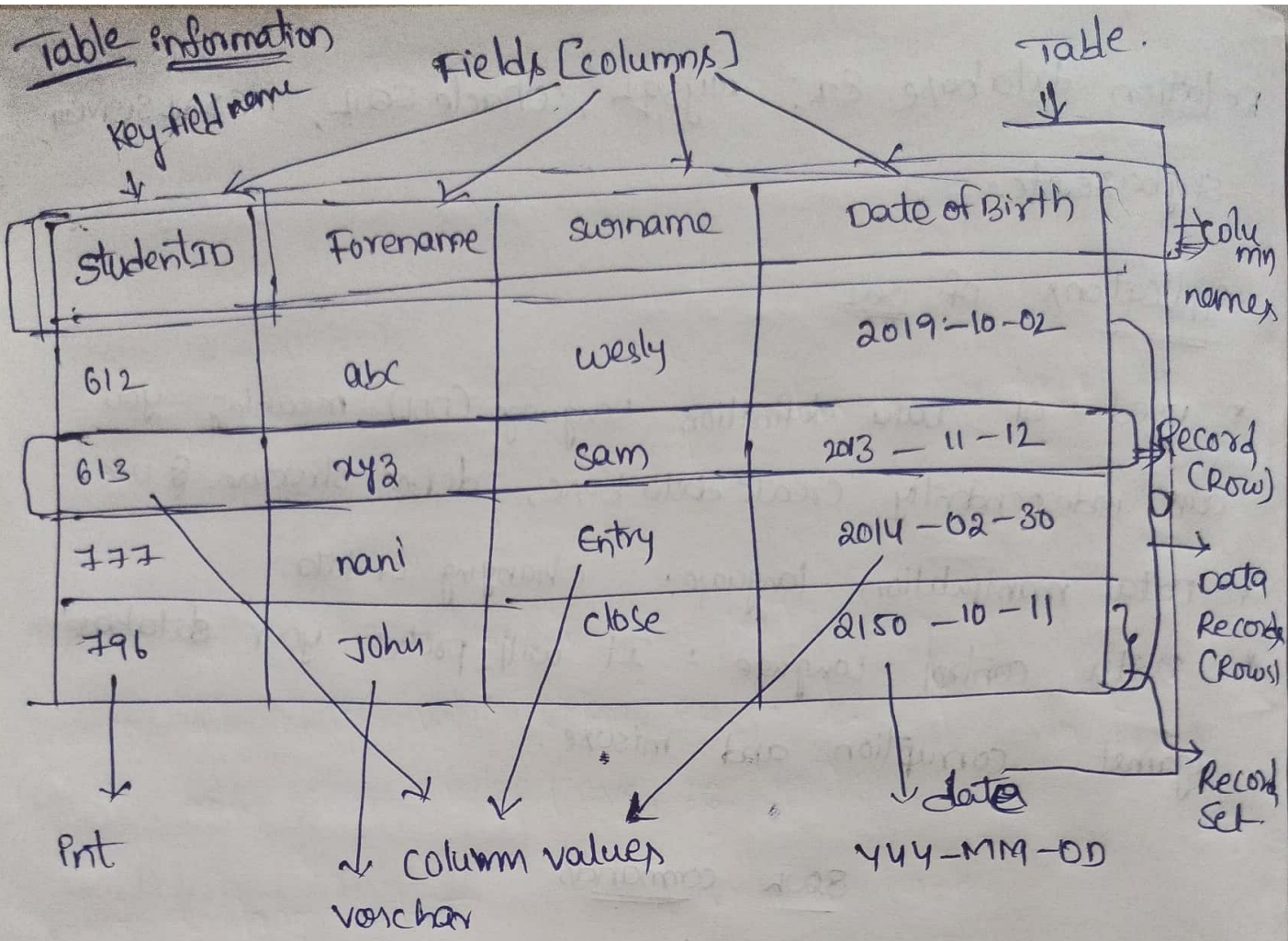
Relation database Ex: MySQL, ORacle SQL, MSSQLServer, sybase.etc.

Applications of SQL

* used of Data definition Language (DDL) meaning you can independently create data base, define structure & use.
* Data manipulation language: changing of data.
* Data control Language: It will protect your database against corruption and misuse.

SQL commands





* studentID: field in the table which provides information about ~~salary~~ studentID of different ~~ID's~~ students

* If one row selected i.e; 613 this gives complete information about particular id 613 i.e; 613 id, name, sur, DOB

SQL Data types

Datatypes

Numeric
- bit - 0-1
- tinybit - 0 - 255
- Smallint - 32,768 + 32768
- int, bigint,
- decimal, numeric,
- float, real

~~character~~

Date & time
- Date, Time,
- Datetime,
- Timestamp,
- Year

JSON/XML => NO

character → string
- char
- varchar
- varchar(max)
- Text

Character Type	Range
char(s)	— 255 characters
varchar(s)	— 255 characters
text	— 65,535 characters

Date - Time Type	Format
date	YYYY-MM-DD
time	HH:MM:SS
year	YYYY

CRUD operations

create

Read

update

Delete.

SQL can do

* Execute queries
* retrive data
* inserts records
* upade records
* delete records
* create new database
* create new table
* create stored procedure → 1000's line programe save in a file & run
* Create views
* set permission on table, procedure & view.

```
create database demo;          // database created name demo

use demo;                      // DB used

create table test(             // created table with name test
  Name varchar(25),
  Age int,                     } fields in a table
  Sex varchar(20));
```

```
insert into test values('Raj', 20, 'M');
insert into test values('Nani', 24, 'M');     } inserting values in fields
insert into test values('SAM', 26, 'M');
```

```
select * from test             // view to table.
```

Name	Age	Sex
Raj	20	M
Nani	24	M
SAM	26	M

⇒ update test
set Age = '19'
where Name = 'Nani'; // Nani 19 M changes

⇒ delete from test where Name = 'Nani';
// Nani row will deleted

⇒ drop database demo // database demo deleted.

Constrains in SQL

Constrains are used to specify rules for data in table.

- Not null
- Default
- unique
- Primary key

Not null constrain

Not null constrain ensure that a column cannot have a null value.

Ex:

e-id	e_name	e-sal	e_age
1	xyz	1000	20
2	223	2000	30
3	abc	3000	40

e_sal Not null // e_sal in column has no null values

Default constrains

Default constrain is used to provide default value for a column. So default values added to all the new records if know the value specified.

Default e_age = 25

e_id	e_name	e_sal	e_age
1	xyz	1000	25
2	223	2000	25
3	abc	3000	25

unique constraint

unique constrain ensures that all values in a column different.

unique e_name;

e_name has different values.

e_id	e_name	e_sal
1	nan	1000
2	sam	2000
3	Raj	3000

Primary key constraint

Primary key constrain uniquely identifies each row ina table.

Primary key = Notnull + unique

e_id primary key

e_id	e_name	e_sal
1	nan	1000
2	Sam	2000
3	Raj	300

e_id has not null and unique values.

Create table

Name the table → Define the columns → Assign data types of columns.

* create table book (
  column1 datatype,
  column2 datatyp,
  ----
  columnN datatype,
  Primary key (column_x));

} syntax to create table.

⇒ create table employee (
  e_id int,
  e_name varchar(20),
  e_salary int,
  e_age int,
  e_gender varchar(20),
  e_dept varchar(20),
  Primary key (e_id));

// Table name employee

} fields names in table

} primary key e_id

* INsert into book
  values (value1, value2, ... valueN);

} Inserting data into table Syntax.

* Select * from employee; (all from employee data select)

Total table display.

⇒ select Distinct syntax

select distinct column_1, column_2 --- column_N from table_name;

select Distinct is used to select only distinct values from our column.

* ⇒ select distinct e_gender from employee;

e_gender
Female
male

⇒ Where clause

Where clause is used to extracts records which satisfy condition.

Ex:- Age > 60 occupation = "Doctor".

Syntax

select column1, column2, columnN

from table_name where [condition]

⇒ select * from employee where e_gender = 'Female';

e_id	e_name	e_sal	e_age	e_gen	e_dept
3	Anne	12500	25	Female	Analyst
4	Jane	13000	30	Female	Analyst

⇒ ~~select~~

⇒ Alter table employee
change column e_sal TO e_age; || e_sal changes to e_age.

⇒ Alter table employee ~~TO~~ student; || Table name changes to student.
↳ ~~TO~~ Rename TO

⇒ update employee
set
e_name = 'Julia' ~~where~~
where e_id = 4;

|| we are changing name for any field
We use mainly 'SET'

e_id	e_name	e_salary	e_age	e_gen	e_dep
4	Julia	13000	35	female	Analyst

⇒ * select * from employee where e_dept = 'analyst';

e_id	e_name	e_salary	e_age	e_gender	e_dept
3	anne	125000	25	female	analyst
4	Julia	73000	35	female	analyst

⇒ To select max salary

* select e_name, max(e_salary) from employee;

e_id	e_name	max(e_salary)
5	sam	159000

// For min to change min

⇒ To select salary lowest to highest using "order by"

* select e_salary from employee order by e_salary;

e_salary
73,000
80,000
95,000
125,000
1,59,000

⇒ * select e_salary from employee order by e_salary desc

e_salary
159000
125000
95000
80000
73000

Group by Alphabetic order

select e_name from employee group e_name;

e_name
anne
bob
Julia
matt
sam

To add a Newcolumn to table

* Alter table employee add column nani varchar(20);

e_id	e_name	e_sal	e_age	e_gen	e_dept	nani
						John

Insert values into New colum particular column

* insert into employee (nani) values ('John'); John added

To delete column in a table

* Alter table employee drop nani;

e_id	e_name	e_sal	e_age	e_gen	e_dept	~~nani~~

To delete Row in a table

* delete from employee, where e_dept = 'sales';

Sales department Row department deleted.

Operators

1) AND operator
2) OR operator
3) NOT operator
4) Like operator
5) Between operator.

1) AND operator: AND operator displays Records if all the Conditions seperated by AND are TRUE

Age > 60 AND occupation = "Doctor"

syntax: select * from employee where e_gender = 'male' AND(age < 30;

e_id	e_name	e_sal	e_age	e_gend	e_dept
2	bob	80000	21	male	Support
6	Jeff	112000	27	male	operations

2) OR operator: OR operator displays records if any of the conditions seperated by OR is True.

occupation = 'Soft eng' OR occupation = "Doctor"

Any one is true / satify is enough.

Syntax:

Select * from employee where e_dept='operations' OR e_dept='Analyst';

e_id	e_name	e_salary	e_age	e_gen	e_dept
1	sam	95K	45	male	operations
3	anne	125K	25	female	Analyst
4	Julia	73K	30	Female	Analyst
6	Jeff	112K	27	Male	Operations

00	0
01	1
10	1
11	1

operation & Analyst both satisfied & displayed both depts.

select * from employee where e_salary>10000 OR e_age>30;

e_id	e_name	e_sal	e_age	e_gen	e_dept
1	sam	95K	45	male	operations
3	anne	125K	25	female	Analyst
5	matt	159K	33	male	sales
6	Jeff	112K	27	female	operations

there e_name sam has 95K salary but condition is e_salary>100K but another condition e_age>30 the sam age 45 2nd condition is satisfy. If any one condition is true / satisfy display output

3) NOT operator: Not operator displays a record if the condition is not true.

NOT ⇒ occupatio='Sof eng' occupation='Doctor' ⇒ O/p occupati='Acto'

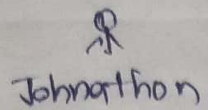
select * from employee where not e_age<35;

			age		
1	sam	95K	45	male	Support

4) Like operator

Like operator is used to extract records where a particular pattern is present.

John%



o/p Johnathon, Johny

% → percentage symbol → Represents zero, one (or) multiple characters

_ → underscore symbol → Represents a single character

syntax

```
select col_list from table_name where
    column_N Like '_xxxx%';
```

select * from employee where e_name Like 'J%';

e_id	e_name	e_sal	e_sal	e_gen	e_dept
4	Julia	13K	30	Female	Analyst
6	Jeff	16K	27	male	operations

select * from employee where e_age Like '3_';

4	Julia	13K	30	female	Analyst
5	matt	15gK	33	male	sales

5) Between operator

→ Between operator is used to select values within a given range. Range.

Age  Between  o/p = 20 - 25

20 25

select * from employee where e_age between 25 AND 35;

From employee table o/p → colum

e_age
25
30
33
27

4 mem b/w 25 - 35.

Functions

1) MIN() function
2) Max() function
3) count() function
4) sum() function
5) Avg() function.

1). **MIN() function:** min function gives smallest value in a column.

* select min(e_age) from employee; o/p:

e_age
21

2) **Max() function:** Max function will gives a Largest value in a column.

* select max(e_salary) from employee; o/p →

e_salary
15gK

3) **count() function:** count function returns the number of rows that match specified criteria.

female 33 male 67 // count b/w 33 - 67 Rows.

Syntax: select count (*) from table_name
where condition;

* select count(*) from employee where e_gender = 'male';

o/p = | male |
      | 4 | 4 males

4) sum() function: sum() function gives the total sum of a numeric column. // only numeric colum.

* select sum(e_salary) from employee;

o/p

Sum
64,54000

5). Avg() function: Avg() function gives the average value of a numeric column.

* select Avg(e_salary) from employee;

o/p → | 90,000 | Avg salary.

String functions

1) LTRIM()
2) LOWER()
3) UPPER()
4) REVERSE()
5) SUBSTRING()

1) LTRIM( ): LTRIM function Removes blanks on the left side of the character expression.

select ' spartaa' ↵ → o/p spartaa.

string with left side space.

* select LTRIM(' spartaa') ⇒ op spartaa → No spaces.

2) Lower( ) and upper( )

string ip select 'This is spartaa'.

* select upper('this is spartaa') ⇒ o/p THIS IS SPARTAA

upper ip string select 'THIS IS SPARTAA'..

* select Lower('THIS IS SPARTAA') ⇒ o/p this is spartaa

4) Reverse( )

select 'This is spartaa'

select reverse('This is spartaa') o/p => aatraps si sihT

5) Substring( )

select 'This is spartaa' // This is complete original string.

I need spartataa has substring. Substring fallows 3 parameters 1). original string 2). Index value of substring where starts 3). Total index of Substring.

* select 'this is sparta'

* select substring ('this is parta', 9, 6) op⇒ // sparta

ORDER BY

order by is used for sort the data in assending or decending order.

acending ↗ ↙ decending

* select * from employee order by e_salary;

e_salary
13K
80K
95K
112K
125K
159K

o/p ⇒

// if e_salary DESC;

e_salary
159K
125K
112K
95K
80K
13K

o/p

TOP clause: Top clause is used to fetch the TOP N Records.

* select TOP 3 * from employee; // It will show 1st 3 rows in employee table

e_id	e_name	e_sal	e_age	e_gen	e_dept
1	sam	95000	45	male	operations
2	bob	80000	21	male	support
3	Anne	125000	25	female	Analyst

Suppose you need top 3 oldest employees from company.

* select TOP 3 * from employee order by e_age DESC;

e_id	e_name	e_sal	e_age	e_gen	e_dept
1	sam	95K	45	male	operations
5	matt	159K	33	male	sales
6	Julia	73K	30	feral	Analyst

Group by :

Group by is used to get aggregate results with respect to group.

	male	female
sal	83K	90K

|| By using Group by to find avg sal male & female separatly.

* select avg(e_salary), e_gender from employee group by e_gender;

Avg	e_gender
99000	female
111500	male

Avg salary

* ~~select Avg(e_salary),~~
* select avg(e_age), e_dept from employee Group by e_dept order by Avg(e_age) DESC;

	e_dept
36	operations
33	sales
27	Analyst
21	support

o/p ⇒ Avg age

Having clause :

Having caluse is used in combination with group by to impose conditions on groups.

why :⇒ where keyword cannot be uses aggregate functions

Ex: depA depB depC depD o/p we need Avg of depA & depC

depA	depB	depC	depD		depA	depC
123K	73K	45K	92K	⇒	128K	115K.

* select e_dept, avg(e_salary) as avg_salary // renaming e_salary as avg_salary.
from employee
group by e_dept;
having avg(e_salary) >10000;

(i) Avg all dept o/p Avg sal com.
(ii) condition satisfy by dept

(i)

e_dept	Avg_salary
Analyst	99K
operations	103500
sales	159000
support	8000

e_dept	Avg_salary
operations	103500
sales	159000

using having clause.

update statement: update statement is used to modify the existing records in a table.

* update employee set e_age = 42 where e_name = 'sam';

e_id	e_name	e_sala	e_age	e_gen	e_dept
1	sam	95K	45	mak	operation

after update function.

e_id	e_name	e_sal	e_age	e_gen	e_dept
1	sam	95K	42	male	operat

value e_age updated 45 to 42;

* update employee set e_dept='Tech' where e_gender='female';

o/p → female department changes to Tech.

Delete statement: Delete statement is used to delete existing records in the table.

* Delete from employee where e_age=33;

o/p → e_age=33; complete row deleted:

* Delete from employee where e_name=sam;

o/p → e_name=sam row has been deleted.

Truncate statement: Truncate statement is used all of the data inside the table.

* Truncate Table employee;

o/p →

e_id	e_name	e_salary	e_age	e_gen	e_dept

All of the records is empty. Even though delete all of the records the structure of the table is not deleted this is advantage. Header will not deleted.

Joins

1) Inner Join

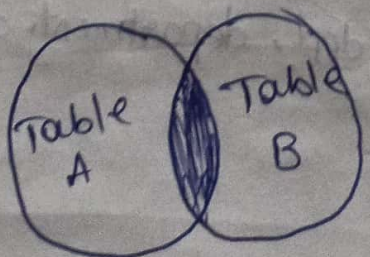
Employee

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	male	operations
2	bob	80000	21	male	support
3	Anne	125000	25	female	Analyst
4	Julia	73000	30	female	Analyst
5	matt	159000	33	male	sales
6	Jeff	112000	27	male	operations

Department

d_id	d_name	d_location
1	Content	New-York
2	Support	Chicago
3	Analyst	Newyork
4	Sales	Boston
5	Tech	Dallas
6	Finance	Chicago

Inner join: Inner join returns that have matching values in both the table. It is also known as simple Join



Syntax:

```
Select columns
From table1
INNER Join table2
ON table1.column_1 = table2.Column_Y;
```

• ON → condition

* select employee.e_name, employee.e_dept, department.d_name, department.d_location

From employee
inner Join department
ON employee.e_dept = department.d_name;

o/p =>

e_name	e_dept	d_name	d_location
bob	support	support	chicago
anne	Analyst	Analyst	Newyork
Julia	Analyst	Analyst	Newyork
matt	sales	sales	Boston

2) Left Join : Left join returns all the records from the Left table, and the matched records from right table

* select employee.e_name, employee.e_dept, department.dname, department.d_location

From employee
Left Join department
ON employee.e_dept = department.d_name;

e_name	e_dept	d_name	d_location
Sam	operations	Null	Null
bob	Support	Support	Chicago
Anne	Analyst	Analyst	Newyork
Julia	Analyst	Analyst	Newyork
matt	Sales	Sales	Boston
Jeff	operations	Null	Null

O/P ⇒

3) Right join : Right join returns all the records from the right table, and the matched records from Leftside table.

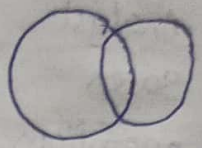
* select employee.e_name, employee.e_dept, department.d_name, department.d_location.
From employee
Right Join department
ON employee.e_dept = department.d_name;

O/P ⇒

e_name	e_dept	d_name	d_location
Null	Null	content	Newyork
bob	Support	Support	Chicago
Anne	Analyst	Analyst	Newyork
Julia	Analyst	Analyst	Newyork
matt	Sales	Sales	Boston
Null	Null	Tech	Dallas
Null	Null	Finance	Chicago

Full Join :

It returns all rows from the left table and Right table with Null values in place where the Join condition is not met.



* select * from employee.e_name, employee.e_dept, department.d_name, department.d_location

from employee

full Join department

ON employee.e_dept = department.d_name;

e_name	e_dept	d_name	d_location
sam	operation	Null	Null
bob	Support	Support	Chicago
anne	Analyst	Analyst	Newyork
Julia	Analyst	Analyst	Newyork
matt	Sales	sales	Boston
jeff	operation	Null	null
Null	Null	content	Newyork
Null	Null	Tech	Dallas
Null	Null	Finance	Chicago

update using Join

Increase +10 age where d_location = 'Newyork' & only Analytic depart ment.

* update is used to change/update value in a row.

* update employee
  set e_age = e_age+10
  from employee
  Join department on employee.e_dept = department.d_name.
  where d_location = 'Newyork'

e_age	e_dept
25	Analytics
30	Analytics

Previously

o/p

e_age	e_dept
35	Analytics
40	Analytics

Delete using Join

Just change update employee to delete employee. Then after fallow join Syntax and apply conditions. where you wan to delete.

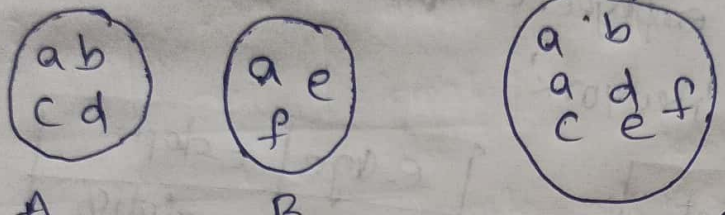
union operation: A union operator is used to Combine the result-set of two or more select Statement.

* Select * from employee → o/p both Combin' 16 row

* When both colums are same in two table then only union operation done.
* It will remove duplicate.

Union all operator (A ∪ B)

union all operator gives all the rows from both the tables including the duplicates.



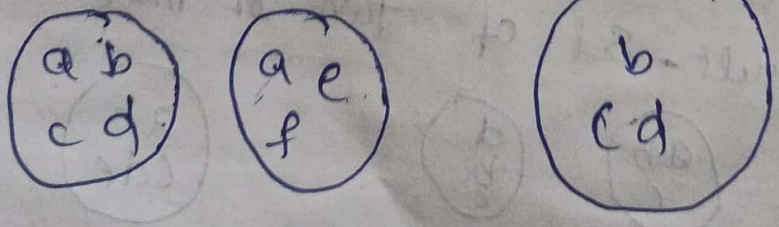
A B A union all B.

* Select * from student 1
union all
select * from student 2

// Both two tables colums same only satisfy union all.

Except operator (A - B)

Except operator combain two select statement and retun unique records from the left query which are not part of the right query.



A B A-B

S_id	S_name	S_marks
1	sam	45
2	bob	87
3	anne	73
4	Julia	92

Student 1

S_id	S_name	S_marks
3	Anne	73
4	Julia	92
5	matt	65

Student 2.

* select * from student1
except
select * from studen 2

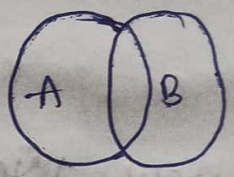
⇒ o/p ⇒

S_id	S_name	S_mark
1	sam	45
2	bob	87

Intersect operator (A∩B)

Intersect operator helps to combain two select statement and return the records which are common to both the select statements.

Gives common records from both tables.



(A∩B)

* select * from student1
intersect
select * from student2

S_id

o/p ⇒

S_Id	S_name	S_mark
3	Anna	73
4	Julia	92

Views

Views are virtual tables used to limit the information that you want to display.

Student table

S_id	S_name	S_gen
1	navi	male
2	sam	female
3	dall	female

you need female view ⇒ o/p

female employees table

S_id	S_name	S_gen
3	dall	female

* create view female employees (Tablename o/p from i/p) as
  select * from student
  where S_gen = 'Female';

o/p ⇒

female employees

S_id	S_name	S_gen
3	dall	female

Drop view

lets drop the view using "drop view" syntax.

* drop view female employees; // droping view of female employees
* select * from female employees; // checking female employees

⇒ o/p ⇒ Invalid object name 'female employees'.

Alter table statement

Alter table statement is used to 1) add
2) delete
3) modify columns in a table.

1) Add column

* Alter table student 1
  Add S_salary int; ⇒ o/p

column name ↓ variable Type ↓

student 1

S_id	S_name	S_mark	S_salary (column added)

* select * from student 1; ⇒

2) Drop column

* Alter table student 1
  Drop column S_salary;

student 1

S_id	S_name	S_mark

* select * from student 1;

1) when we droping column
drop column & colum name;
Syntax

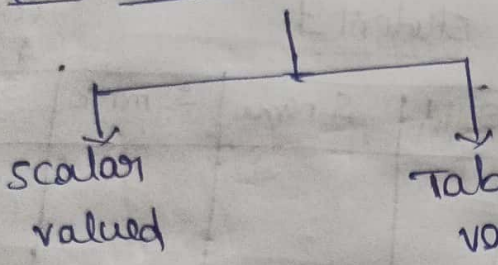
If you need enter data into student 1.

* insert into student1 values (', 'ram', 20);

Merge Statement

Merge is the combination of insert, delete and update statement.

User Defined functions



Scalar valued function

scalar valued function always returns a scalar value.

i.e; int, varchar, data ... so on.

Syntax

```
create function function_name (@ nanil datatype-)
returns return_datatype
AS
Begin
-----function body
Return value
end.
```

name ↓ parameter ↓ ← Return parameter int

* create function add five (@num as int)

```
Returns int
AS
begin
Return (
    @num +5)
)
```

statement.

* select dbo.add five(10) o/p => 15.

// To calling created function.

Table valued function

Table valued function returns a table insted of scala.

syntax

create function function_name (@nani datatype, @ syan dat ----)

Return table

AS

Return (select colum list from table_name where (condition))

* select * from employee;

// we are creating table valued function to employee table.

=>* create function select_gender (@gender as varchar(20))

Retun table

AS

Return

(

select * from employee where e_gender=@gender

)

// employee table e_gender shows.

* select * from dbo.Select_gender ('male')

o/p => shows male gender.

Temporary table

Temporary tables are created in tempdatabase and delete as soon as the session is terminated. used to store temp data.

Syntax: create table #table_name ( );

* create table #student (// #student table name #means it will temp table..
  s_id int
  s_name varchar(20)
  );
* select * from #student

#student.

s_id	s_name

Case Statement

Case statement is used to multiway decision making.

1 → Teq
2 → coffie
3 → nani
⋮
N → water.

Syntax:

Case
  when condition 1 then result 1.
  when condition 2 then result 2.
  when condition N then resultN

  Else result

END;

```
* select
  case
  when 10> 20 then '10 is great 20'
  when 10< 20 then '10 is less 20'
  else '10 is equal to 20'
  end.
```

O/P ⇒ 10 is less 20

selecting employee table & performing case statements to e_salary and adding result to new column name as grade.

```
* select * from employee
* select *, grade  (new colum name)
  case
  when e_salary<90000 then 'C'
  when e_salary<120000 then 'B'
  else 'A'
  end
  from employee
  go
```

results added in this.

```
select * from employee;
update employee set grade= case
then condition & end;
```

O/P ⇒

e_id	e_name	e_sal	e_age	e_gen	e_dep	grade
1	sam	95K	45	mal	operx	B
2	bob	80K	21	mal	suppor	C
3	anne	125K	25	femal	Analy	A
4	Julia	73K	30	femal	Analy	C
5	matt	159K	33	mal	Sales	A
6	Jeff	112K	27	mal	oper	B

IIF() function

If function is an alternate of the case statement

it takes first value (boolen expression, true value, false value)
1 2 3

* Select
iif (10>20, '10 greter 20', '10 Less 20').

o/p ⇒ 10 Less 20

Select * from employee

select e_id, e_name, e_age, iif(e_age>30, 'oldemp', 'youngemp')
as employee_generation from employee.

o/p ⇒ employee_generation added new colum for e_age> condition

Stored procedure in SQL

stored procedure is a prepared SQL code which can be saved and reused.

* Create procedure employe_age ← procedure name
as
select e_age from employee
go

// stored procedure created.

lets execute stored procedure

* exec employee_age o/p →

e_age
45
21
25
30
33
27

=> Another example

* create procedure employeedetails
as
select * from employe..
go

exec → exec employeedetails o/p => All employee details come as well like employee table.

Example: we are calling female employees from employee table using stored procedure using parameter.

create procedure employee_gender @gender varchar(20)
as
Select * from employee
where e_gender = @gender
go
exec employee_gender @gender = 'female'

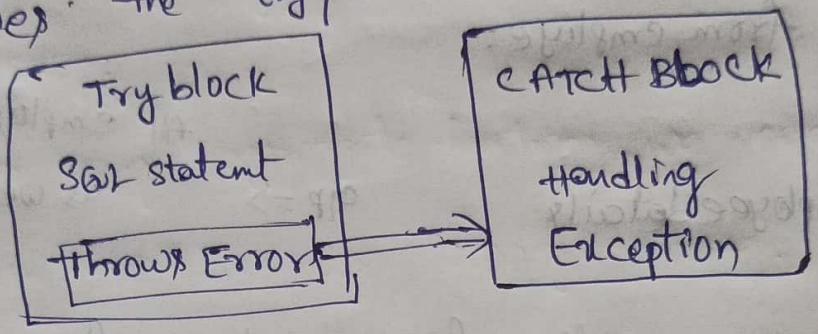
o/p => 2 rows effected
female rows come output.

Exception handling

Exception: An error condition during a program execution is called an exception.

Exception Handling: The mechanism for resolving such an exception is exception handling.

SQL provides the try/catch blocks for exception handling.

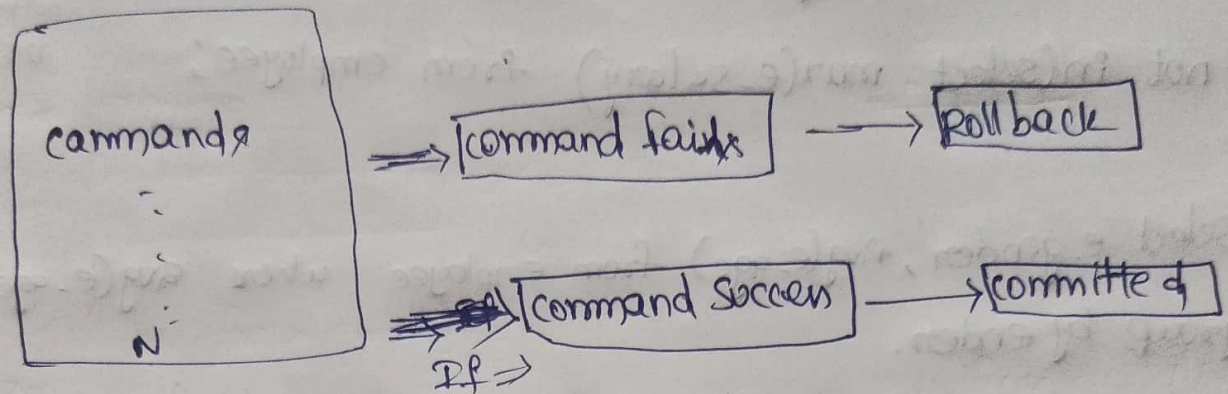



```
declare @val1 int;
declare @val2 int;

begin try
set @val1 = 8;
set @val2 = @val1/0
end try

begin catch
print error_message()

end catch;
```

o/p => divide by zero error commited.

Transactions in SQL

Transaction is a group of commands that change data stored in a database.



* begin transaction

  update employee set e_age = 30 where e_name = '30'.

  o/p = e_age changes to 30 in sam, ⇒ // Previously samage = 45 changed to = 30

  rollback transaction. ⇒ // come to 45 original

* begin transaction

  update employee set e_age = 30 where e_name = 'Julia'

  commit transaction.

// o/p ⇒ e_age = 30 where e_name = 'Julia'

** when we apply commit transaction we cannot rollback.

Question

① Fetch 2nd highest salary in employee table.

* select max(e_salary) from employee where e_salary not in(select max(e_salary) from employee);

② select e_gender, Avg(e_age) from employee where avg(e_age)>30 group by erdon.

* whenever using aggregate functions we cannot use where clause. we can use having clause.

* select e_gender, Avg(e_age) from employee group by gender having(e_age)>30.

③ Stuff( ) function.

Stuff(string1, position, length, string2)

By using this we are changing 'SQL Tutorial' to Python'.

* select Stuff('SQL Tutorial', 1, 3, 'Python')

o/p => Python tutorial.