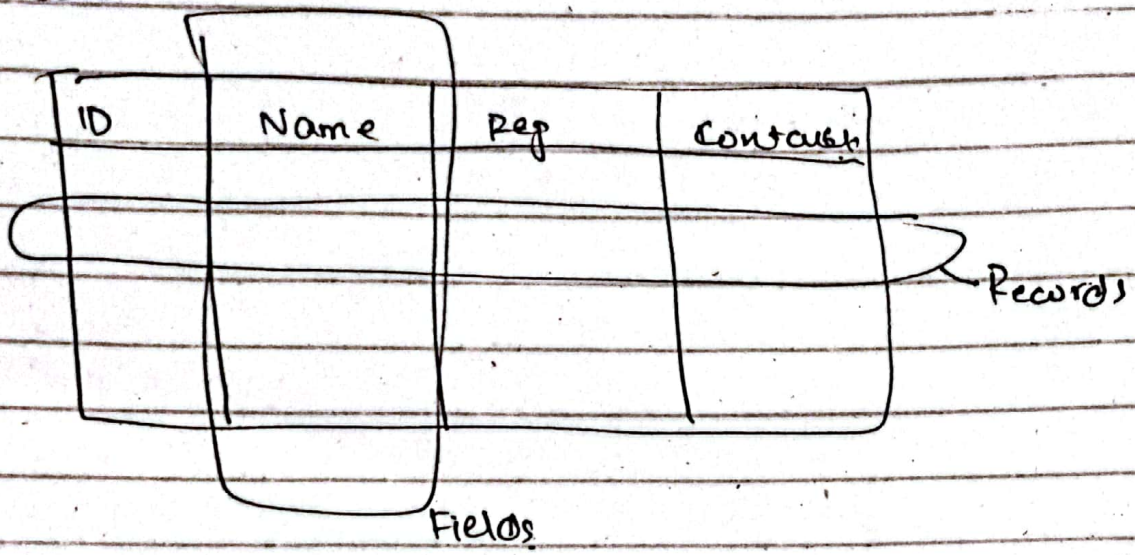
SQL Tables.

1) Fields.
2) Record



→ Data Types.

→ 1) Numerics

1) Bigint = 9234567908 <-> 8887478907
2) INT = 2147836 <-> 2147483647
3) small int = 32468 <-> 54897
4) tinyint = 0 <-> 255
5) decimal (s,d) $-10^{38}+1$ <-> $10^{38}-1$

→ 2) Characters

→ Char(s) 255 characters
→ Varchar(s) 255 characters.
→ text 65,535 characters

→ 3) Date - Time.

→ Date YYYY - MM - DD
→ Time HH: MM: SS
→ Year YYYY

Constraints in SQL.

→ Not null (Fields always has a value)
→ Default (if 25 then all value in field will be 25)
→ Unique (All value in filed are different)

→ Primary key (Not null + Unique)

:

: Steps

Create a Table.

Name the table : Define the Columns : Assign Data Type of Columns

→ Table cannot have more than 1 primary key.

→ Creating Table (Syntax)

```
Create Table employee (
e-id int not null,
e-name varchar (20),
e-salary int,
e-age int,
e-gender varchar (20),
e-dept varchar (20),
primary key (e-id).
);
```

→ Insert Into.

Insert into employee values (
1, 'Sam', 95000, 45, 'Male', 'Operations'
);

1→ Select statement
(How to extract particular fields / columns)

1→ Select e_name from employee;
2→ Select e_name, e_gender, e_salary from employee;

2→ Select statements
(How to extract all info from a table)

1→ Select * from employee;

How?
1→ Creating a Database.

Create database hafly;

2→ Use an Existing Database

Use hafly;

3→ Drop / Delete a Database

Drop database ignews;

→ Select Distinct.
: Select only distinct (different) values

→ Select distinct e_gender from employee;

We will get : [Male, Female]

→ Where (Clause).
(For Single Condition)
: Extract record using single condition.

1) Female employees
→ Select * from employee where e_gender = 'Female';

2) Age less than 30
→ Select * from employee where e_age < 30;

3) Salary greater than 100,000
Select * from employee where e_salary > 100,000;

→ For Multiple Conditions.

1) And operator.
2) Or operator
3) Not operator.

1) And Operator

Select * from employee where e_gender = 'Male'
AND e_age < 30;

→ Select * from employee where e_dept = 'Operations'
AND e_salary > 60000;

2 → OR operators..

→ Select * from employee where e_dept = 'Operations'
OR e_dept = 'Analytics';

3 → Not operators..

→ Select * from employee where not e_gender = 'Female';

→ Select * from employee where not e_age < 30;

---

1) like Operator. — wild card characters: percentage % symbol, underscore _ symbol

2) Between Operator

1) like operator

Extract records where a particular pattern is present.

J = conditions

John Johnathon Johny Marcus

→ used in conjunction with wild card characters

- % symbol (Represent zero, one or multiple characters)
- Underscore _ Symbol (Represents a single character)

→ Select * from employee where e_name LIKE 'J%';

Names starting with J will be extracted.

→ Employees in 30s.
Select * from employee where e_age LIKE '3_';

2⇒ Between Operator
Used to select values within a given : range.

→ Extract records whose age is b/w 25 & 35.
Select * from employee where e_age BETWEEN 25 AND 35.;

→ salary b/w 90,000 to 1,00,000.
Select * from employee where e_salary BETWEEN 90000 AND 100000;

Basic Functions :-

1. Min
2. Max
3. Count
4. Sum
5. AVG

Min.

```
Select min (e_age) from employee;
```

Max.

```
Select max (e_age) from employee;
```

Count.

```
Select count (*) from employee where
e_gender = 'Male';
```

Sum.

```
select sum (e_salary) from employee;
```

Avg.

```
select AVG (e_age) from employee;
```

→ String Functions (SQL).

1) LTrim (removes blank spaces on left side)
2) Lower (converts characters to lower case)
3) Upper. (Converts all characters to upper case letters)
4) Reverse (Reverses all characters in the string)
5) Substring. (Gives a substring from original string)

LTrim.

Removes blank spaces on the left side.

→ Select ' spartaaaaaa'

Now it has blank spaces on left
To remove it

→ Select Ltrim (copy paste it here)

Lower.

Converts characters to lower case

→ Select 'THIS IS SPARTAAAAA'
To apply lower

→ Select lower (copy paste)

Upper.

Converts to characters to upper case

→ Select 'Hello World'
Convert to upper.

→ Select upper (copy paste)

Reverse

Reverses all characters in the string.

→ Select 'i love ice cream'
→ Now Applying reverse

→ Select reverse (copy paste it)

Substring -

→ Gives a substring from the orginal string.

→ Orginal string:
Select 'this is sparta'

→ Substring:
select substring (copypaste, 9, 6)
- 9 → counting from first letter to s (from t to s) (counting also includes spaces)
- 6 → Alphabets of character word (sparta has 6)

Clauses (SQL)

1) Order By.
2) Top (clause)

1) Order By

Used to sort data (Ascending & Descending)
Default it sorts in Ascending.
(ASC) (DESC)

→ Select * from employee order by e_salary;
↓
Sorting with respect to e_salary (Ascending)

→ Descending :
Select * from employee order by e_salary DESC;

2) Top Clause.

Top 10 record, Top 1000 etc

→ Top 3 records.:
Select Top 3 * from employee;

→ Using both Top Clause & Order By clause to get
(3 oldest employee of company)

→ Select Top 3 * from employee order by e_age DESC;

3 Group By (statement)

Used to get aggregate result with respect to a specific group.

e.g (with respect to gender)
Avg salary of male employee
Avg salary of female employees.

↓

~~e_gender~~

→ ~~Select avg (e_salary), from employee group by e_gender;~~

→ Select avg (e_salary), e_gender from employee group by e_gender;

→ Avg age with respect to dept and sort in DESC

↓

~~from employee.~~

→ ~~Select avg (e_age), e.dept group by e_dept order by avg(age) DESC;~~

→ Select avg (e_age), e_dept from employee group by e_dept order by avg(e_age) DESC;

→ Having Clause:
Used in conjunction / combination with Group by to impose condition on groups

: Where cannot be used with aggregate functions

→ Display only those depart. whose avg salary is greater than 100000

→ Select e_dept, avg(e_salary) as avg_salary from employee group by e_dept having avg(e_salary) > 100000

→ Update statements:
Used to modify / update existing records in table

→ Syntax:
→ update employee set e_age = 42 where e_name = 'Sam';

→ update employee set e_~~gend~~dept = 'Tech' where e_gender = 'Female';

→ Update employee set e_salary = 50000;

→ Delete statement.

Delete existing record.

→ Delete from employee where e_age = 33;

→ Delete from employee where e_name = 'Sam';

→ Truncate statement.

Deletes all data inside table except the structure ~~such~~ / headings.

→ Truncate table employee ;

Joins in SQL.

1. Inner Join.

Also known as simple join.

It gives us those records that have matching values in both the tables.

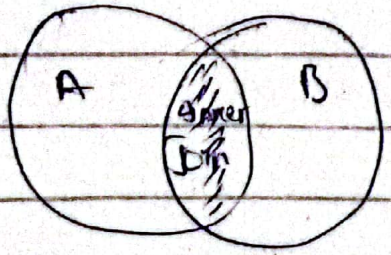


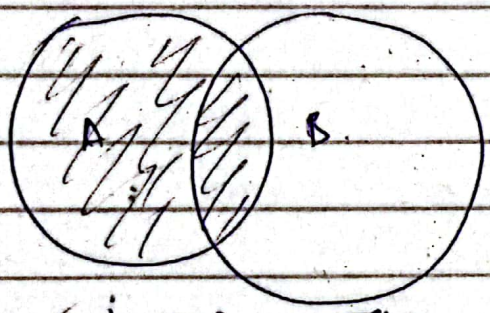
Table 1 = employee Table 2 = department

↳ Syntax

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;

→ left Join.

Returns all records from left table
and the matched records from
the right table

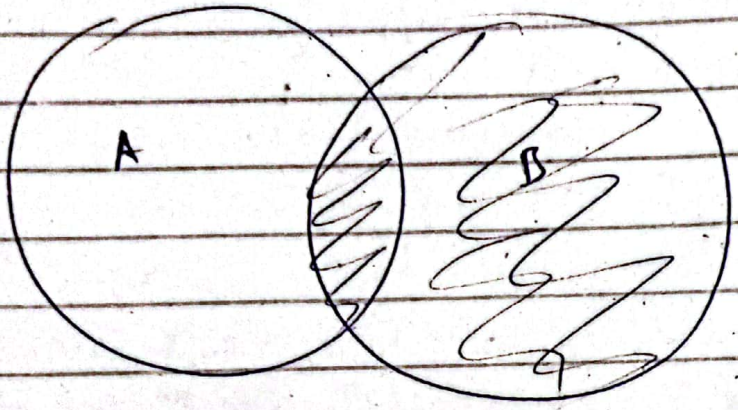


left جدول right جدول ← ↳ display null.

→ Select employee. e_name, employee. dept,
department. d_name, department. d_location
From employee
left join department ON employee. e_dept :=
department. d_name;

→ Right Join -

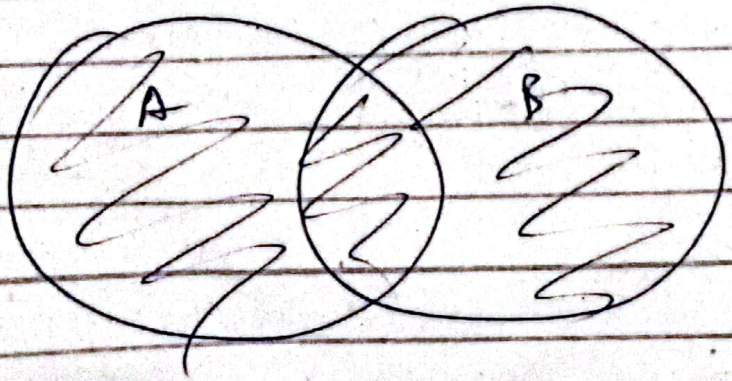
Returns all records from right table and the matched records from left table



→ 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 ;

→ Full Join.

Returns all records from both left & right table and gives a null value where join condition is not met.



→ Select employee. e_name, employee. e_dept, ~~employee~~ department. d_name, department. d_loca
from employee
Full join department ON employee e_dept =
department d_name ;

:

Update using Join

Update employee set e_age = e_age + 10
from employee
Join department ON employee. e_dept = department. d_name
where d_location = 'New York';

Delete Using Join

Delete employee
from employee
Join department ON employee. e_dept = department. d_name
where de_location = 'New York'

Operators (SQL).

↳ Union Operator.
  ↳ combine
  ↳ The output will not consits of duplicates
  ↳ Before performing union operator, keep in mind that the no. of columns should be same in both the selected statements / tables

Table 1 = Student_Details 1 &
Table 2 = Student_Details 2.

→ Select * from student_details 1
Union
Select * from student_details 2. ;

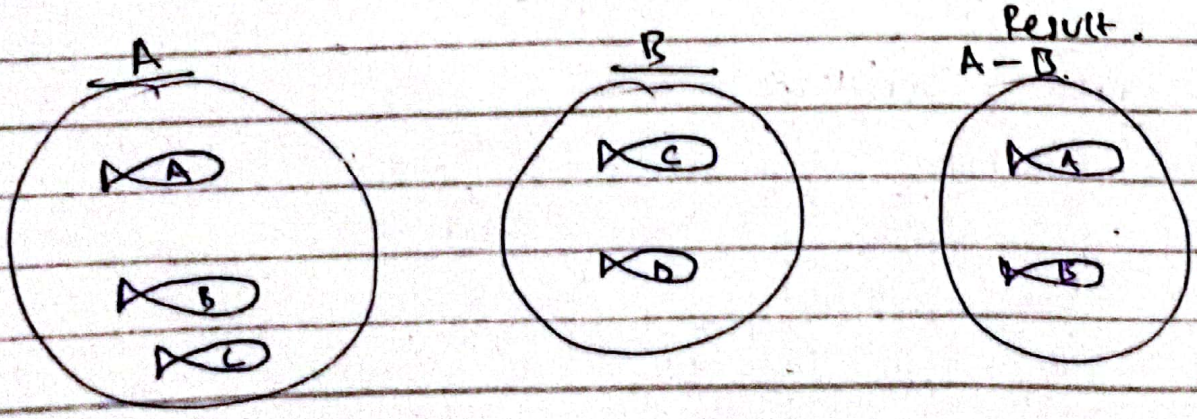
3 Union All Operator.

→ Also gives the duplicates.

→ Select * from student_details 1
Union All
Select * from student_details 2. ;

3 Except Operator.

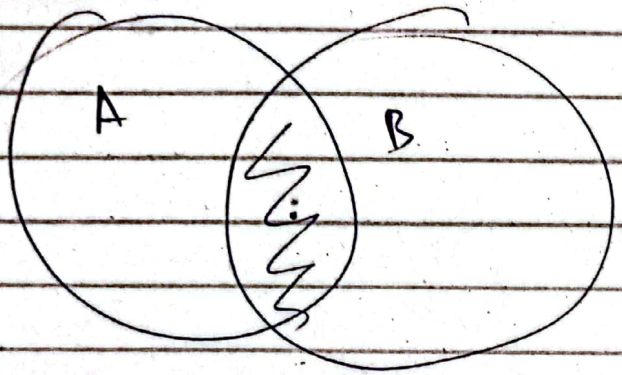
→ Combine 2 select statements & returns unique records from the left query which that are not part of right query.



→ Select * from student_details1
Except
Select * from student_details2;

→ Intersect Operator.

Combines. Returns records that are common in both select statements.



No. & order of columns should be same

→ Select * from student_details1
Intersect
select * from student_details2;

Views in SQL.

→ Virtual table :
→ such as Create view of female employees
→ Difference b/w group by & this is that it ~~appears~~ creates another view / seperate table

1. Create view.

Create view data of

→ ~~Extract data~~ of ↑ all female employees

→ view name

→ Create view (female_employees) As.

Select * from employee

where e_gender = 'Female';

→ First create e view (give name) AS

→ select all columns from table (employee)

→ Give condition

2. Drop view.

→ Drop view (female_employees);

↓

view name

Alter Table.

→ Used to add, delete, modify columns in a table

Add Column.

→ Alter table employee. → table *

ADD e_dob date;

↓ ↳ datatype

column name

Drop / Delete column.

→ alter table employee
Drop column e_dob ;

Merge.

→ Combination of Insert, Delete & Update statements.

→ Requirement.
Source Table
Target table. → changes will be applied to this from the source table
→ should have common column.

→ Source Table => Employee_Source
→ Target Table => Employee_Target

→ Merge Employee_Target AS T
~~Merge~~ Using Employee_Source AS S.
ON T.e_id = S.e_id
When matched
Then update set T.e_salary = S.e_salary,
T.e_age = S.e_age
When not matched by Target
Then insert (e_id, e_name, e_salary, e_age, e_gender, e_dept)
Values (S.e_id, S.e_name, S.e_salary, S.e_age, S.e_gender, S.e_dept)

When not matched by source
Then delete ;

Types of User Defined Functions

Scalar valued | Table valued

→ Scalar valued. int, varchar, date etc
Returns a scalar value.

```
→ Create function add_five (@num as int)
Returns int
As
Begin
Return (
@num + 5
)
End
```

→ Select dbo.add_five (10)

→ Table Valued Function..
Returns a table

```
→ Create function select_gender (@gender as varchar(20))
Returns Table
As
Return (select * from employee where e_gender = @gender)
```

lets extract all male employe.

→ Select * from dbo.select_gender('Male')

Extract female employee

→ Select * from dbo.select_gender('Female')

→ Temporary Table

They are created in temp DB and are deleted as soon as the session is terminated.

connection

when user connect to DB server

→ Create table #student (
s_id int,
s_name varchar(20)
);

→ select * from #student

→ Insert into #student values(
1, 'Sam'
);

Continued from Temporary Table.

→ Case Statement

Helps in multi-way decision making

Like If statement.

→ Select *, grade =
case
when e_salary < 90000 then 'C'
when e_salary < 120000 then 'B'
else 'A'
~~from~~ employee end
from employee
go ;

→ IIF Function / Statement.

→ Alternative of case statement.

→ Select *,
IIF (e_age > 30, 'Old', 'Young') as employee generation From employee ;

Stored Procedure in SQL

It is a prepared sql code which can be saved & reused.

save query as a stored procedure that we oftenly use

Without Parameter

→ Create procedure employee_age
As
select e_age from employee
GO;

To see the procedure
→ Exec employee_age

→ Create procedure employee_details
As
select * from employee
GO;

→ Exec employee_details

With Parameter.

→ Sete Create function select_gender (@gender
as varchar (20))
Returns table
As
Return
(select * from employee
where e_gender = @gender)

→ To see.
Selow * from dbo. select_gender ('Male')

Exception Handling.

Exception.
An error condition during a program execution

Exception Handling.
Mechanism for solving such exception is exception handling.

→ We can do Exception handling using try catch block.

Try Block

Throws Error

Catch Block.

Handles Exception.

→ Begin try
select e_salary + e_name from employee
end try

begin catch
Print 'cannot add numerical & string values'
End catch

→ Declare @val1 as int; Declaring variable
Declare @ val2 as int;

Begin Try
Set @ val1 = 8
Set @ val2 = @val1/0

End Try

Begin Catch
Print error_message ()
End Catch.

Transactions in SQL

↘ one single unit

Group of commands that change data stored in a database

→ Begin transaction
Update ~~tt~~employee set e-gender = 30
where e-name = 'Sam'

→ To roll back
Begin transaction
Update employee set e-age = 30 where
e-name = 'Sam'
roll back transaction

→ Because we are using transaction (since command is inside a transaction) we can roll back and commit

→ Begin transaction

Update employee set e-age = 30 where

e-name = 'Sam'

commit transaction

→ Begin try

Begin Transaction

Update employee set e-salary = 50 where

e-gender = 'Male'

Update employee set e-salary = 45/0

where e-gender = 'Female'

commit transaction

Print 'transaction committed'

End try

Begin catch

roll back transaction

print 'transaction rolled back'

End catch