Official Documentation: dev.mysql.com

PAGE NO.

MySQL

Database : Organised collection of logically related Data

Understand Database v/s DBMS

1) **RDBMS :** Relational Database Management System.

2). Other Databases : 1) Oracle 2) MongoDB 3) Postgre 4) Fire Bird 5) Redis.

SQL v/s MySQL

SQL is basically a language to talk with Database (Structured Query Language)

MySQL is a DBMS that uses SQL.

MySQL v/s ORACLE DB

Minute differences in Syntax
Major difference in functionalities

MySQL is also a product of ORACLE.

Clear command line: system cls.

PAGE NO.:

- Listing Databases.

→ show Databases ;

- Creating Database.

```
Create Database <db-name>
        OR
Create Database if Not exists store_db;
```

- Working with Database

Use <db-name>

- Delete Databases

Drop if exists <db_name>

eg:→

```
Drop database if exists temp;
```

* Now working with TABLE *

Imp: ( ) separated by ","
enclosed.

eg:→
```
Create Table students (

name  varchar(100),
id    int
);  → {Imp}
```

Check your Table

```
Desc  <Table_Name>
```

eg: Desc students

◆ Adding Data into a Table ◆

(i)
```
Insert into students (id, name)
values (101, "Rahul")

OR

Insert into students values (101, "Rahul")
```

For string use " " strictly.

This second Method has a catch.

For example:

① Insert into students values (101, "Raj");

↳ Here, you need to provide it with the same sequence.

Q.) How to Insert Multiple Data?

Insert Multiple

```
Insert into Students
Values (103, "Raju"), (104, "Shyam");
```

READING DATA FROM A TABLE

Syntax: ① Select * from <table_name>

② Select <column_name> from students

eg: ① Select * from student;

② select id from student;

③ /* Selecting Multiple Columns */

/Select id, name from student/

READ DATA ON CONDITION

Query:

① select * from students where id = 3;

explaination:

मुझे सारा Data chaiye जिसका id is 3;

Modify / Update Data from a Table

①
```
update  students
   set  contact = 12345
where   name = "Rabindra";
```
→ use double inverted only.

②
```
update  students
   set  Contact = 567
   where  id = 2;
```

Note: updating based on primary key is best practice because in other case it could also update multiple records.

Here, id must be a primary key

Notes (Stored Procedure)

PAGE NO.

DELETE TABLE Operations

① Delete From students
where id = 3;

↳ will delete entire Record (Row)

(Imp:) Decode (Extra).

```
Delimeter $$

Create Procedure delete_if_exists (sid int)
Begin

    if Exists ( select 1 From student where id = sid)
    Then Delete From Student where id = sid;

    End if;
End $$

Delimeter ;
```

CASE Study :

(i)
```
Create Table if not Exists Cust (
    id int,
    name Varchar(50)
);
```

We have cust Table:

id	name

```
Insert into Cust (id)
Values (2), (3);
```

(#) Here, we are only inserting into id column

Output :

Id	Name	
2	Null	} Because Default was set to Null
3	Null	

We can also insert NULL values.

eg:->
```
Insert into Customers
Values (NULL, NULL);
```

Output :->

Id	Name
NULL	NULL

NOT NULL

PAGE NO.:

★ Thus needs to Apply Not NULL constraints

```
Create Table Customers (
        id int Not Null,
        name Varchar(100) Not NULL
);
```

'Not Null' is a good Practice because default Null values mostly don't contribute much to Database.

To Delete Entire Table with Table Structure use "DROP"

>> `DROP Table <Table_Name>`

Now after setting Not Null constraint

* Query :->

```
Insert into Customers (id)
Values (3);
```

↳ o/p: Error because name field againd id='3' will become Null which is not allowed.

DEFAULT VALUE

This used to set default values for fields Mostly helps in tackling not null constraint.

eg:->

```
CREATE TABLE employee (

    name varchar(100),
    acc_type varchar(50) DEFAULT 'savings'
);
```

↓

default is been set to 'savings'.

- Insert into custom (id, name)

- Insert into employee (id, name)
  values (1, "Rahul"), (2, "Shyam");

Output:->

id	name	acc-type
1	Rahul	Savings
2	Shyam	Savings

→
```
Create Table Customer3
(
    id int Not NULL,
    name varchar(50) Not Null,
    acc-type varchar(50) Not Null Default 'savings'
);
```

➡ MySQL does not have string as a Datatype.

```
Insert into customer3 (id, name)
Values (101, 'Rajo'), (202, 'Rajo');
```

PRIMARY KEY

1) Primary key is used uniquely to identify each record in a table

2) Primary Key value cannot be Null and must be unique

3) A Table can have one Primary Key

```
Create Table customer4 (
    acc_no int Primary Key,
    name varchar(50) Not NULL,
    acc_type varchar(50) Default 'savings'
);
```

```
Desc customer4;
```

(Note:) <> Not Equal to Operator in MySQL.

```
Insert into customer4 (name)
Values ("Raj");
```

↳ This will throw the Error. because it will set id as Null while id is Primary Key

Soln:-> Auto Increement

AUTO INCREEMENT

```
Create Table Customers (

    acc_no int Primary Key Auto_increement,
    name varchar (100) NOT NULL,
    acc_type varchar (50) NOT NULL Default 'savings'
);
```

Extra :->

Field	Type	Null	Key	Default	Extra
acc_no.	Int	NO	Primary	NULL	Auto_Increement
name	Varchar	NO		NULL	
acc_type	Varchar	NO		Savings	

Query:

```
Insert Into Customers (name)
Values ('Rajesh'), ('Kamlesh');
```

↓

Now, this not Throw error because id is set to Auto_Increement and acc_type has a default value 'savings'.

Although you can also explicitly provide id even after autoIncreement.

O/P:

acc_no	name	acc_type
1	Rajesh	savings
2	Kamlesh	savings

Suppose, you have explicitly provided acc_no. as 101

eg:-> Insert into Customer5 (id, name)
values (101, "Yahu");

->(O/P:)

acc_no.	name	acc_type
1	Rajesh	Saving
2	Kamlesh	Savings
101	Yahu	Savings

IMP

Q.) Now auto-increement will give 3 OR 102?
Ans 102 ie [id will become (102)]
+1 by latest value.

ALIAS

Alias :-> (As) Keyword

Example:

>> Select acc_no as 'Account No.' From customers.

O/P:->

Account No.
1001
1002

Note: Char(2) is a fixed length string;

We can also change Multiple Columns using "ALIAS"

» Select acc_no As 'Account No.', name As 'Customer Name' From Customers

Output :->

Account No.	Customer Name
1001	Raju
1002	Sham

(H.W) Case Study ON <Char v/s VARCHAR>

Ans ①

Char(3) ie fixed length is 3
If we store something less than 3 than the MySQL pads it with spaces ie, remaining length will include blank space
Thus not space Efficient for variable length string.

② Memory

Char(5) will always use 5 Bytes even if string is lesser in length.

Varchar(N): NE variable. It will take bytes as per string size + 1/2 Byte to store length of string.

PAGE NO.:

③ Performance.

- Varchar(N) mostly slower than Char() for larger Records. due to handle variable length.

Q.2 Updating the Records.

Ans

```
Update employees SET dept = 'IT' Where
                                emp_id=104;
```

Q.2 Delete a record where emp_id=102

Ans

```
Delete From employees where id=102;
```

{ String Functions }

(in)

MySQL

① CONCAT (first_col , sec_col)

CONCAT (first_word , sec_word , ...)

Small Case Study

Eg :->

a)
```
Select Concat ('Hey', 'Buddy');
```
↳ o/p : Hey Buddy

b) Can also concat multiple strings.

```
Select Concat ('Hi', 'Hello', 'How?');
```
↳ o/p :-> HiHello How?

Q.) How to add spaces while concatenating String?

Ans

```
Select Contact ('Rabindra', ' ', 'Mishra');
```
↳ o/p :-> Rabindra Mishra

Consider a Table,

emp_id	fname	lname	desig	dept
101	Raju	Rastogi	Manager	loan
102	Sham	Mohan	Cashier	Cash
103	Baburao	Apte	Associate	loan
104	Paul	Philip	Accountant	Account
105	ALEX	Watt	Associate.	Deposit

Query:

(i) Select emp_id, Concat(fname, lname) as fullname From employees;

O/P:->

emp_id	Fullname
101	Raju Rastogi
102	Sham Mohan
103	Baburao Apte
104	Paul Philip
105	Alex Watt

② CONCAT_WS

Syntax :→ `Concat_ws ('-', fname, lname);`

ws (with Separator) to provide Separators.

eg :→ `Concat_ws ('-', 'Hi', 'Hello', 'Bye');`

Output : Hi-Hello-Bye

Query 02 :

>> `Select Concat_ws (':', emp_id, fname, desig) as Complete From employee;`

SUBSTRING

Syntax: Substring ('Rabindra Jha', 2, 4);

space is also counted.

↳ output : abin

(ii) We can also write it this way.

→ Select Substring ('Rabindra', 3);

only start.

so it will automatically traverse till last.

» Negative Indexing

Select Substring ('Hey Buddy, Wassup Raju', -4);

Q.) I only want last two digits of emp_id?

Sol^n: (H.W) Find which is better Query.

Select Substring (emp_id, -2);

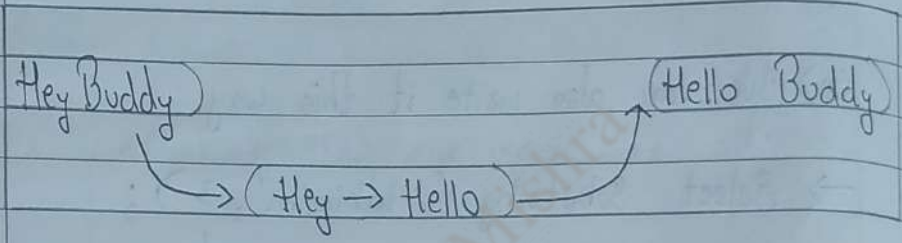
(OR)

Select Substring (emp_id, 2);

Note: MySQL may not allow (-) negative indexing.

Instead We can use Right( ); Function.

REPLACE ()



» Used for Replacing our strings.

Takes 3 parameters.

क्या Change करना है। किस से Change करना है। String क्या है।

Syntax: Replace (str , from_str, to_str);

↓

eg:→ Replace ('He Buddy', 'He', 'Hello');

Similarly, we can also replace characters of words.

» Select replace ("Rabindra", "in", "ea");

↳ o/p :->

Q.2 Replacing the emp_id (whole colum) with certain values let say 1000?

Sol^n :-> Select Replace (emp_id, 10, 1000) as newid, fname from Employee;

↳ o/p : It will show newid & fname.

newid	fname
10001	Raju
10002	Sham
10003	Babusao
10004	Paul
10005	Alex

Comple emp-id could get replace.

(i) Case Study.

Consider a table

emp_id	fname
101	Raju
102	Jignesh
103	Sham
104	Paul
105	Alex

(ii) We can replace This in this way Also.

emp

(>>) Select Replace (emp_id, 10, 'EMP') as ID's, fname from Employee;

Output:

emp_id	fname
EMP 1	Raju
EMP 2	Jignesh
EMP 3	Sham
EMP 4	Paul
EMP 5	Alex

REVERSE

Used to reverse the string.

Select Reverse ('Hello')

↳ olleh

Q.2 Reverse the fname of Employee Table.

Ans

Select reverse (fname) as. rname from Employee;

LOWER & UPPER

(i) » Select Upper ('abcdxyz');

↳ o/p: ABCDXYZ

(ii) Select lower ('RAMESH');

↳ o/p: ramesh.

Similarly we can also use

UCASE: Select UCASE ('HellM');

(UPPER)

↳ o/p: HELLM

(iii) LCASE: (Lower)

```
Select LCASE ('TCET');
↳o/p: tcet.
```

Q.2 Convert all the (fname) to upper?

Soln:

```
Select emp_id, UPPER (fname) from Employee;
```

Char_Length

(i)

```
Select Char_length ('Raju');
↳o/p:-> 4
```

(ii)

```
Select Char_length ('Hi Raju');
↳o/p:-> 7
```

Understanding the Query

```
Select fname, Char_length(fname) as Length
From employees;
```

fname	length
Raju	4
Sham	4
Paul	4
Jignesh	7

Q.) How to find fname whose length >5 ?

Sol^n :->

```
Select fname from Employee
where char_length(fname) > 5;
```

OTHER FUNCTIONS

1. Insert
2. Left
3. Right
4. Repeat
5. TRIM.

Insert :-> To add word/character between a string.

eg :->

```
Select insert('Hey Wassup', 5, 0, 'Raju ');
```

5 → starting position to be inserted.

O/p :-> Hey Wassup → Hey Raju Wassup.

② LEFT

```
:: Select Left ('Abcd', 3)
↳ o/p :-> Abc
```

③ Right

```
Select Right ('AbcD', 2)
↓
o/p :-> CD        (#2 From Right ie Second last)
```

④ TRIM (use to remove Spaces)

eg :->

```
Select TRIM (' Hello ');
↓
o/p :-> Hello
```

⑤ Repeat (used to Repeat Words)

```
Select Repeat ('HA', 5);
o/p :-> HAHAHAHAHA
```

PAGE NO.:

Task 3
(i) 101:Raju:MANAGER:LOAN
Task 4
(ii) 1101 Raju

- Please check github for Solution of Tasks

DISTINCT

Select Distinct fname from Employees;

Suppose,

fname	lname
Paul	Watt
Watt	Paul
Rsem	Shu

// Select fname, lname // and remove similiar pairs

Query

① Select Distinct fname, lname from employees;

ORDER BY

used for sorting the name/other fields.

② Select * From employees Order By fname;

o/p:

emp_id	fname	lname	Dept	....
103	Alex	Jha	Cash	
104	Anurag	Thakur	Loan	
101	Sagar	Mishra	Admin	
105	Tushar	Yadav	Clerk	

Sorted

PAGE NO.:

Reverse Sorting.

(1) Select * from employees
Order by fname Desc;
↳ reverse sorting.

(2) Select dept, fname from employees
Order by dept, fname;

→ In this tie breaking criteria for dept would be fname.

Like Keyword

① Select * from employees
Where dept Like "%Acc%";

Used for partial word Matching.

② Select * from employees
where desig Like "%cash%"

emp_id	fname	lname	Desig	Dept
102	Sham	Mohan	Cashier	Cash

↑
related to cash

Q.1 Find fname having four length?

Ans

Select * from employees
where fname Like "____";

four underscore

Q.2 Find all fname starting with R?

Ans

Select * from employees
where fname like "R___";

Here, '%' means zero or More characters.

① where fname like "%x%";
This means any number of sequence before and after is okay.
Typically means any word containg character x.

② where fname like "%x".
This means x is at the end of the word.

Q.2 Find a word starting with Ale and last word as m?

Ans.

```
Select * from employee
where fname like "Ale%m";
```

Making changes to our table structure.

```
Alter Table employees
Add column
Salary int not null
Default 25000;
```

Note:

Insert into inserts a whole new record ie (row) thus cannot be used to insert Data into newly created column.

General SQL Query:

1. Select
2. From
3. Join
4. where
5. Group By
6. Having
7. Order By
8. Limit

} Sequence of SQL clauses.

PAGE NO.

Limit

① Helps to limit the records which is been displayed.

eg:-> Select emp_id from employee limit 3;

{It will select first 3 Record}

② We can also give the range.

eg:-> Select emp_id from emp limit 3,3;

Q.) Find Top 4 records from salary.?

Ans) Select salary, emp_id from emp
Order by salary Desc limit 4;

order by is used for sorting.

COUNT

>> Show tables;

Count the No. of records in a Table.

① Select Count(*) from employees;

O/P :->

Count(*)
9

(records).

② If a certain column does not have data (null) count on value may differ from total count of Table

③ Select Count(fname) from employees;

Q.) Count number of distinct Departments in Dept column?

(Ans)

```
Select Count(Distinct Dept) From Employees;
```

O/P :-> 5

PAGE NO.:

Q.2 Find Number of Managers from our employees table ?

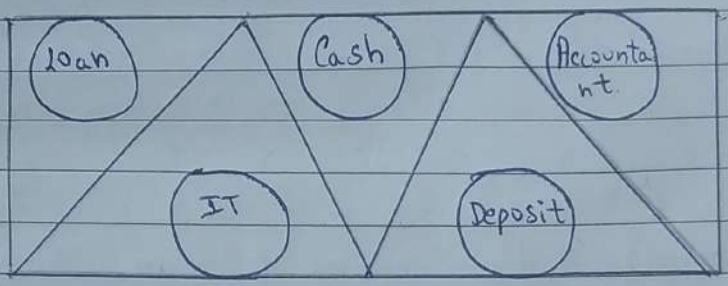
Ans

```
Select Count(emp_id) from employees
Where desig = "Manager";
```

GROUP BY

→ Used for grouping based on different column.

→ Suppose we are Applying Group By on 'dept' column.



They all will form a group.

① → Select dept From employees Group By dept;

② Select dept, count (fname) from employes Group By dept;

It will give count of (fname) Department wise.

But Best practice is to use Primary Key.

③ Select dept, Count (emp_id) from employees Group By dept;

↓

{ Primary Key. }

④ Select dept as Department from emp;

Q.2 Find count of employee salary more than 32000 designation wise.

Ans

Conditional Count.:
Where clause cannot be used inside an aggregate function like count.
Thus for conditional count we can use 'case when'.

SQL Query :->

```
Select desig, Count (case when salary > 32000 Then
                          emp_id)
from emp
Group By desig;
```

Min & Max Functions

① » Select max(salary) from employees;

② Select min(salary) from employees;

Q.1 Give emp_id, fname of employee whose salary is Maximum.

(Ans)

Select fname, emp_id, max(salary) from employees; ⊗

Thus we need to Use Sub Queries.

Sub Queries

Select emp_id, fname, salary from employees where
Salary = (select Max(Salary) from Employees);

In Sub queries we simply combine multiple Queries to get our results.

In SubQuery concept Inner Query runs first ie (select Max(salary) from employees) runs first.

Note:

```
Case
  when condition 1 then result1,
  when condition 2 then result2
  ⋮
End
```

can be used inside function.

Min & max can also be used along with strings.

» Select Max(fname) from employee;
o/p:-> sham

» Select Min(fname) from employee;
o/p:-> Alex.

Sum & Avg

```
Select sum(salary) from employee;
select Avg(salary) from employee;
```

Remember :→ cls;

①
```
Select dept, sum(salary) from employees
Group By dept;
```

Output: It will give output of sum(salary) based on departments.

②
```
Select dept, Count(emp_id), sum(salary)
From Employees
Group By dept;
```

Q.2 Give me the dept name with lowest Avg Salary?

Ans

```
Select Dept
From emp
Group By Dept
Order By Avg(salary) Asc
Limit 1;
```

Q.8 Write a Query to find total salary of 'Loan' Department ?

Ans)

```
Select Sum(salary) from employee
Where Dept = "Loan";
```

DATATYPES

1. Decimal

eg:

Field	Type	Null	Key	Default	Extra
PRICE	INT	YES		NULL	

```
Insert into num
Values (156.28);

select * from num
```

#num is int type

olp:

Price
156

} However number stored will be integer

Syntax: (6,3)
- 6 → Total Digits (includes decimal)
- 3 → Precision

Maximum Number of Digits allowed is 65.

```
Create Table num1 (
  Price Decimal (5,2)
);

Insert into num1 Values (155.78);
Select * from num1;
```

(i) 119.12 ✓ (ii) 28.15 ✓ (iii) 1150.15 ✗

3 2 (Total 5). 2 2 Total 6

② FLOAT

4 Bytes of Memory., can represent upto 7 digits

③ Double

Upto 7-digits, takes 8 bytes of Memory.

Eg:->

① Create Table num2 (

f float,
d Double
);

Insert into num2 Values
(123.456, 123.456);

Select * from num2;

O/P:->

f	d
123.456	123.456

No Difference

(2) Insert into num2 values
(123.123456789, 123.123456789)

Select * from num2;

f	d
123.456	123.456
123.123	123.123456789

(3) DATE, TIME, DATETIME

(i) Date: yyyy-mm-dd format

(ii) Time: HH:MM:SS

(iii) DateTime: (yyyy-mm-dd HH:MM:SS)

Syntax:

```
Create Table Person
(
    joindate    DATE,
    join time   Time,
    join_DT     DateTime
);

Desc Person;
```

```
Insert into Person
Values ('2022-04-17', '23:01:08',
'2023-05-16 22:05:02');

Select * from Person;
```

PAGE NO.:

CURDATE, CURTIME, NOW

(i) CURDATE() :- yyyy-mm-dd
{current Date}

(ii) CURTIME() :- hh:mm:ss (उसी का time)
{current Time}

(iii) Now() :-> yyyy-mm-dd hh:mm:ss
(now).

```
Select  curtime();
Select  Now();
Select  Cur
```

Actual use:

```
Insert into person
Values ( curdate(), curtime(), Now() );
Select * from Person;
```

Additional Functions

(1) DAYNAME (2) DAYOF MONTH (3) DAYOF WEEK

a) Select DAYNAME ("2007-02-03");

b) Select DAYOF MONTH ("2007-02-03");

c) Select DAYOF WEEK ("2007-02-03");

a) Select DAYNAME (curDate());

b) Select DayOfMonth (curDate());

c) Select DAYOF WEEK (curDate());

④ MONTHNAME

```
Select Monthname ('2023-01-31');
```

We can also use this in a table.
(for example :→)

```
Select jd, Monthname(jd)
from Person;
```

⑤ YEAR (converts to Year)

```
Select Year('2021-03-20');
```

⑥ HOUR

↱ Hours (Time)

```
Select Hour ("23:42:56")
```

↳ O/P :→ 23

DATE FORMATTING

a) Tue Mar 28th
b) 21st Tue at 21:20:28
c) 2023/04/18.

① Date_Format (now(), '%D %a at %T')

o/p:-> 21st Tue at 21:20:28

② Date_Format (now() , '%m /%d /%y ');

o/p:-> 04/16/23

Format Specifier :-

%a = WeekDay
%b = Month Name
%c = Month (Numeric)
%D = Day of Month English
%d = Day of month, numeric (00...31)
%e = Day of Month, numeric (0 0...31)

(H.w) Please visit official MySQL website for more specific Details.

Date Math Function

Datediff (expr1, expr2); #returns null if not valid Date.

Date_ADD (date, Interval expr unit)
Date_SUB (date, Interval expr unit)

Syntax:->

DateDiff ('2023-04-20', '2023-04-15')
O/p:->5 (Result will be in terms of Days)

Note: If first date is smaller then answer may come in negative.

//Example:

(a) Date_ADD ('2023-05-01', Interval 1 Day)

↳ Adds one Day to the given Date

(b) Date_ADD ('2023-05-01', Interval 1 year);

(c) Date_SUB ('2023-05-01', Interval 1 Month);

PAGE NO.

➡ SELECT DATE_ADD (Now(), Interval 1 year);

Output :

Date_ADD (Now(), Interval 1 year)
2024-04-21 22:15:43

Q.1 Find Date Previous to One Year ?

(Ans)

Select DATE_SUB (Now(), Interval 1 year);

TIMEDIFF (expr1 , expr2)

eg:-> TimeDiff ('23:00:00' , '20:00:50')

o/p :-> 02:59:10

Default & ON Update Timestamp

- Simple concept which uses timestamps.

```
Create Table blogs (
        text varchar(150),
        created_at Datetime default Current_Timestamp,
        updated_at Datetime ON Update Current_Timestamp
);
```

```
Create Table blogs (
blog    varchar(150),
ct      Datetime Default Current_Timestamp,
ut      Datetime ON Update Current_TimeStamp
);
```

```
Desc blogs;   (Analyze Properly).
```

```
Insert into blogs.(text)
Values.('This is my first blog');

// ct field will store current_Timestamp By Default.
```

O/P:->

Blog	ct	ut
This is my first Blog	2023-04-21 22:28:02	Null

But Now, after updating ut will get updated

(i)
```
Update blogs
set blog = 'This is updated'

select * from blogs
```

Now, (o/p :→) | Ct would not be affected but ut would have been changed

(ii) Decimal v/s Float v/s Double

Decimal:
Variable ; based on the number of digits specified

Float:
4 bytes (32 bits)

Double:
8 bytes (64 bits)

```
Create Table blogs (

    title varchar(150),
    crtime Timestamp Default Current_Timestamp,
    updatetime Timestamp Default Current_Timestamp
     ON UPDATE Current_Timestamp

);
```

↳ correct Syntax.

Relational Operators

Q.1 Find employees whose salary > 65,000?

Ans

```
Select * From Employees
Where salary > 65,000;
```

Q.2 Find Employees salary < 65000?

Ans

```
Select * from Employees
Where salary < 65,000;
```

Operators	Meaning
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
=	Equal to
!=	Not Equal to

Q.3 Find employees salary <= 65,000?

Ans

```
Select * from Employees
Where salary <= 65,000;
```

Q.2 Find the employees greater than equal to 65,000?

Ans

```
Select * From Employees
Where Salary >= 65000
```

Q.2 Find employees whose salary is not equal to 25K?

Ans

```
Select * From Employees
Where Salary != 25,000;
```

Logical Operators

(1) AND (Both True)

Q.1 Find employees of loan department whose salary is 25,000?

Ans

```
Select * From Employees
Where Salary = 25,000 and dept = "Loan";
```

(2) OR Operators.

Q.2 Find employees either working in loan dept or having salary of 25,000?

Ans

```
Select * From Employees
Where Salary = 25000 or dept = 'Loan';
```

(3) IN Operators

Q.2 Find employees from following department

a) Accountant
b) Cash
c) Loan

Query :

```
Select * From Employee
Where dept = 'Accountant'
  OR  dept = 'Cash'
  OR  dept = 'Loan';
```

But, "IN" operator will simplify it.

(i)
```
Select * From employees
Where dept in ('Accountant', 'Cash', 'Loan');
```

(4) NOT IN Operator

```
Select * From Employees
Where dept NOT IN ('Loan', 'Cash', 'Accountant');
```

PAGE NO.:

BETWEEN

Used to talk about Range

Q.) Find employees whose salary is more than 40000 and less than 65000?

Ans

Method 1:

```
Select * from Employees
Where
Salary >= 40000 and Salary <= 65,000;
```

Using Between Operator (Method 02)

```
Select * From employees
Where salary Between 4000 and 65000;
```

PAGE NO.:

CASE

(a) Used to create Reports or Categories based on different cases and Thus the name

```
Select
    fname, Salary,
Case
    When Salary >= 50,000 Then 'Higher Salary'
    ELse 'Low Salary'
END
AS 'Salary Category' (# will create new column)
FROM
    employees;
```

fname	Salary	Salary Category
Mohan	35,000	Low Salary
Gopal	55,000	High Salary

IS NULL

eg:->
```
Select * from employees
where fname is NULL;
```

Not like

```
Select * from employees
Where fname Not like 'R%';
```

↳ O/P:→ It will give all names starting with any letter except 'R'.

Q.2 Find all employes whose end character of fname is not x?

Ans.

```
Select * from employees
Where fname not like '%x';
```

CONSTRAINTS
(Applies Restriction)

1) **Unique**

Because we can only use one Primary Key in a Table Thus to enter only unique values we can use this constraint.

eg:→ / mobile Number Unique ; /

```
Create Table Contact (
        mob Varchar(15) Unique
);
```

Field	Type	Null	Key	Default	Extra
mob	Varchar(15)	Yes	UNI	NULL	

IMP

2) **Check** { Safety Contraint }

We want to make sure Phone No. is Atleast 10 Digits...

Ans

```
Create Table contacts (
        name Varchar(50),
        mob Varchar(15) Unique Check
                (length(mob) >= 10)
);
```

Named Constraints :

To display name or constraint error example.

```
Create Table Contacts2(
        name Varchar(50),
        mob Varchar(15) Unique,
        Constraint mob_no_less_than_10digits
         Check (Length(mob) >= 10)
);
```

named Constraint

Now, error will be displayed with this name.

Alter Command

Used to Alter the structure of the Table.

(1) Add or Remove a Column

```
Alter Table concats
Add column City Varchar(50);
```

Field	Type	Null	Key	Default	Extra
mob	varchar(15)	Yes	UNI	Null	

Query

```
Alter Table contact
Add column name Varchar(50);
```

#It will add name column.
But By defaul values in name would be stored as a Null

mob	name
12345	Null
12345	Null

Drop Column : (Removes the column)

```
Alter Table contacts
Drop Column city;
```

```
Desc contacts;
```

<Rename Column or Table Name>

eg:->
```
Alter Table contacts
Rename Column name to full_name;
```

```
Alter Table contact
Rename Column name to fname;
```

Rename the Table :

```
Alter Table contacts
Rename to mycontacts;
```

```
Rename Table contacts To mycontacts;
```

Modify the Column

Example: Changing datatype or adding default values etc.

- Use 'Modify' Keyword.

Query:

```
Alter table contacts
Modify mob Varchar(15) Default 'unKnown';
```

Query: (changing length)

```
Alter Table contacts
Modify mob Varchar(20) Default 'unKnown';
```

RELATIONSHIPS

ONE to ONE

ONE to MANY

MANY to MANY

(i)

emp_id	name	dept
101	Raju	IT
102	Sham	Finance

1:1

emp_id	city	Title
101	Delhi	Manager
102	Patna	Accountant

Only one record you will get to see mapped against Raju.

(ii)

Cone employe can have many tasks).

emp_id	name	dept	1:m
101	Raju	IT	
102	Sham	Finance	

emp_id	task_no.
101	TS-1
102	TS-2
101	TS-3

Note: Maximum Time we get to see use case of 1: Many

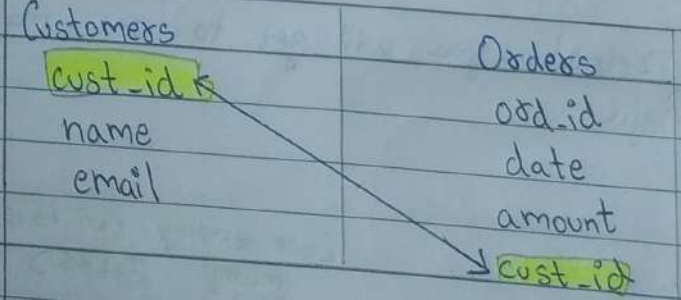
1: Many

Why? Consider an example of Customer Table.

Cust_name	email	order_daate	amount
Raju	raju@123	2023-05-15	200
Sham	Sham@123	2023-04-28	500
Raju	raju@123	2023-05-14	1000
BabuRao	Bab@123	Null	Null
Sham	Sham@123	2023-03-15	800

Name is being repeated due to repeated orders and at different dates.

FOREIGN KEY



```
Create Table Customers (

    cust_id int Auto_increement Primary Key,
    name Varchar (50),
    email Varchar (5)
);
Desc Customers
```

```
Create Table orders (

    ord_id int Auto_increement Primary Key,
    date Date,
    amount Decimal (10, 2),
    cust_id int,
    Foreign Key (cust_id) References customers (cust_id)
);
```

↓

This how we declare a foreign_key

Desc Orders; # To check for primary key.

☆☆ Imp ↴ (This is very important)

Q.] How to find Foreign Key for a given table.

(Ans) Select Constraint_Name, Column_Name, Referenced_Table_Name FROM

INFORMATION_Schema.Key_Column_Usage Where Table_Name = 'orders';

Case Study

1) Insert into orders (date, amount, cust_id)
Values
(CURDATE(), 105.38, 1);

Select * from orders;

2) Insert into orders (date, amount, cust_id)
Values (CURDATE(), 500.38, 1);

3) Insert into orders (date, amount, cust_id)
Values (CURDATE(), 503.38, 2);

Suppose, consider this Table Customers

Cust_id	name	email
1	Raju	raju@gmail
2	Sham	sham@gmail

Query: Insert into orders (date, amount, cust_id)
Values (CURDATE(), 503.38, 10);
↳ error pat

O/P:→ Error [This will throw error because in customer table we only have {cust_id [1,2]} but trying to insert cust_id = 10 in Orders Table ∵ Cust_id is "Foreign Key"

JOINS

This operation is used to combine rows from two or more tables based on a related column between them.

Note: Understand the `` Backticks functionality.

Truncating a Table that is referenced by a foreign key constraint is not allowed. Directly

Solutions :

① Disable foreign key

```
Set Foreign_Key_checks = 0;
Truncate Table customers;
Set Foreign_Key_checks = 1;
```

② Delete all Rows.

```
Delete From customers;
```

③ Drop Foreign Key Constraint

```
Select * FROM customers
Where cust_id = 5;
```

Types of JOINS

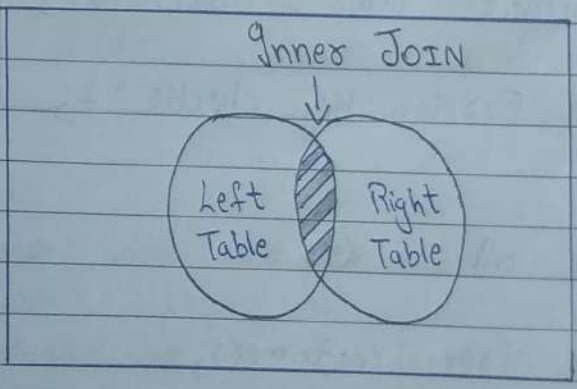
1. Cross Join
2. Inner Join
3. Left Join
4. Right Join

(i) **Cross Join**: Every Row from one Table will be combined with every Row from another Table

```
Select * From customers, orders;
```

↳ CROSS JOIN (Mostly not used).

(ii) **INNER JOIN** :



Returns only the Rows where there is a match between the specified columns in both the

left (OR first) and right (or second) Tables

①
```
Select * From Customers
INNER JOIN Orders
ON orders.cust_id = customers.cust_id;
```
(cust_id) (cust_id)

②
```
Select name FROM Customers
    INNER Join Orders
    ON orders.cust_id = customers.cust_id
Group By name.
```

Q.2 Find the Purchase for different Customers?

Ans

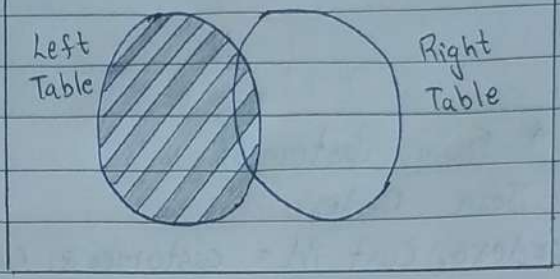
Thus use Grouping

```
Select name, Sum (amount) FROM Customers
    INNER Join Orders
    ON Orders.cust_id = customers.cust_id
Group By name
```

(iii) Left JOIN :

Returns all rows from the left (or first) table and the matching rows from the right Table.

Left Join




```
Select * From Customers
    Left Join Orders
    ON orders.cust_id = customers.cust_id;
```

Example:

Cust_id	Name
1	A
2	B

ord_id	Amt	Cust_id
5	500	1
6	200	1
7	600	2

Left Join

Cust_id	Name	Amt	ord_id
1	A	500	5
1	A	200	6
2	B	600	7

Query:

```
Select name, Sum(amount) FROM customers
Left Join Orders
ON orders.cust_id = customers.cust_id
GROUP By Name;
```

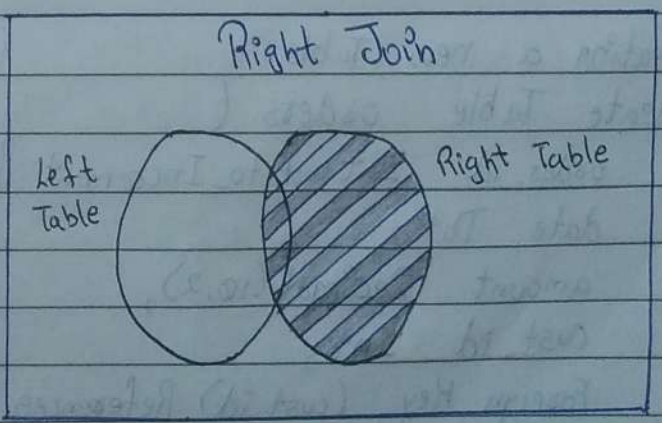
We can represent Null with 0 using this query

(i) IFNULL (sum(amount), 0);

(ii)
```
Select name, IFNull ( sum(amount), 0 ) from
customers Left Join orders
ON orders.cust_id = customers.cust_id
GROUP By name;
```

(iv) RIGHT JOIN :

Returns all rows from the right table and the matching rows from the left table.



Query :->

```
Select * From customers
    Right Join orders
    ON orders.cust_id = customers.cust_id;
```

Delete ON CASCADE

```
Delete From customers
Where name = 'Raju';
```

Problem :-> :: "customer" Table has a foreign key mapping to "orders" Table.

Foreign Key constraints
(cannot delete or update a Parent Row).

Creating a new Table

1.
```
Create Table orders (
    order_id Int Auto_Increment Primary Key,
    date Date,
    amount Decimal(10,2),
    cust_id Int,
    Foreign Key (cust_id) References (cust_id)
    ON DELETE CASCADE                (customers)
);          ↳ This will help in maintaining Integrity
```

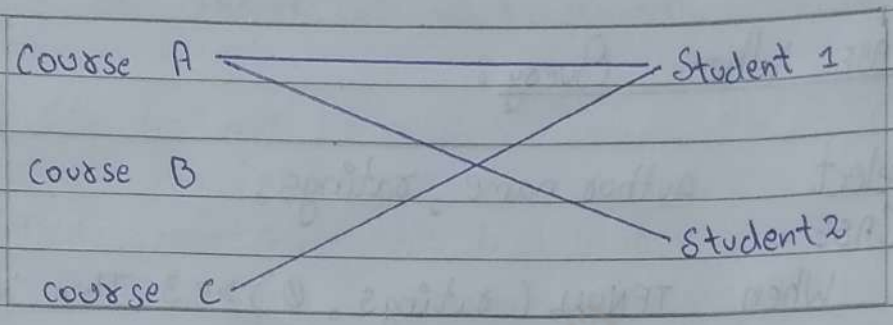
```
Insert Into orders (date, amount, cust_id)
Values ( curDate(), 100.50, 1), (CURDATE(), 500, 2);
```

CASE When Query:

```
Select author_name, ratings,
CASE
    When IFNULL(ratings, 0) >= 3 Then 'Good'
    Else 'Average'
END as remark
from authors
inner join books
on books.au_id = authors.author_id.
```

PAGE NO. :
DATE : / /

Many To Many Relationship



One course → Many students
one Student → Many course

Students
1) id
2) std_name

Courses
1) id
2) course_name
3) fees

In Many to Many we have a concept called Junction Table

Students_course
1) std_id
2) course_id

PAGE NO.:
DATE: 13/8/2024

(i) Create DATABASE Institude;
Use institute;

a) Create Table students (
id int Auto_increment Primary Key,
student_name Varchar (250)
);

b) Create Table courses (
id int Auto_increment Primary Key,
course_name Varchar (250),
fees int
);

Junction Table

c) Create Table student_course (
Student_id int,
course_id int,
Foreign Key (student_id) References
students (id),
Foreign Key (course_id) References
courses (id)
);

PAGE NO.:
DATE: / /

Inserting Data Into Table.

① Insert into students (student_name)
Values ('Raju'), ('Utkarsh'), ('Rabindra'),
('KanishK');

Insert into students (student_name)
Values ('Raju') ⊗

② Insert into courses (id, course_name, fees)
Values (101, 'PD', 3000)

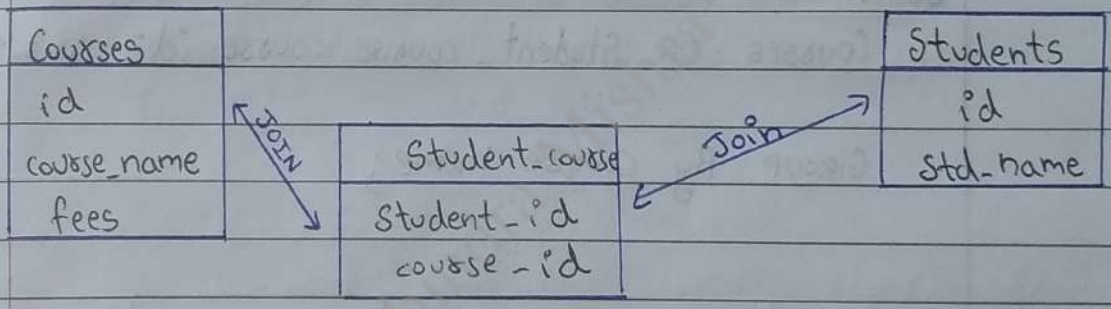
Insert Into courses (course_name, fees)
Values ('Java', 5000), ('SQL', 4000), ('Python', 6000);

③ Insert into student_course (student_id, course_id)
Values (1,101), (1, 102), (2,105), (1,105), (3,103);

PAGE NO.:
DATE: / /

```
Select student_name, course_name From students
JOIN
    student_course ON student_course.student_id =
                      students.id

JOIN
    courses ON student_course.course_id =
                courses.id.
```



Student_name	Course_name
Raju	PD
Raju	Java
Raju	Linux
Sham	Linux
Sham	Java
Paul	SQL
Alex	Python

PAGE NO.:
DATE: / /

Q.1) Print No. of students for each course.

Ans)

```
Select course_name, COUNT (student_name)
FROM students
JOIN
    student_course ON student_course.Student_id=
    student.id

JOIN
    Courses ON Student_course.course_id= courses.id

    GROUP By course name;
```

O/P :->

Course_name	Count
PD	1
Java	2
Linux	2
SQL	1
Python	1

PAGE NO.:
DATE: / /

Q.2) No. of courses for each student? (fees)

Ans)

```
Select Student_name, Sum(fees) FROM Students
JOIN
    Student_course ON Student_course.student_id=
    students.id
JOIN

    Courses ON student_courses.course_id =
    courses.id
    GROUP By Student_name;
```

Q.3) No. of courses for each student?

Ans)

```
Select student_name, Count(course_name) FROM
Students

JOIN
    student_course ON student_course.student_id=
    students.id

JOIN

    courses ON student_course.course_id= courses.id
GROUP By student_name;
```

PAGE NO.:
DATE: / /

VIRTUAL TABLES

Repetition of similiar Data from multiple Queries. It does not store actual Data. Parameters cannot be given. unlike Procedure.

Step 1: Create a virtual view and choose only the relevant information to be displayed.

(1) Create View inst_info As
Select student_name, course_name, fees
FROM Students

JOIN
Student_course ON student_course.student_id = Students.id

JOIN
Courses ON student_course.course_id = courses.id;

It is a stored Procedure and whenever invoked produces a result

Invoking our View Query.

```
>> Select *From inst_info;
```

Output:

Student_name	course_name	fees
Raju	PD	3000
Raju	Java	5000
Raju	Linux	10000
Sham	Linux	10000
Sham	Java	5000
Paul	SQL	4000
Alex	Python	6000

We can also perform other operations like

```
>> Select * from inst_info where student_name=
   'Raju';
```

Note: Where filters before grouping
Having filters after grouping.

PAGE NO.:
DATE: / /

HAVING CLAUSE

Q.1 Find the fees (Total) Payable By students?

Ans

```
Select student_name, SUM(fees) FROM
inst_info GROUP By student_name;
```

Output:

Student_name	SUM (Fees)
Raju	18000
Sham	15000
Paul	4000
Alex	6000

Q.2 Find students having Paid Fees More than 10,000?

Ans

```
Select student_name, SUM (fees) FROM inst_info
GROUP By student_name
Where SUM (fees) >10,000;
```

Error: Where clause does not work with GROUP By.

Where clause is allowed (x) (not) in Aggregate functions like Count(), SUM(), AVG() etc. are calculated after rows has been selected and grouped.

Therefore,

Correct Query

```
Select student_name, SUM (fees) From inst_info
GROUP By student_name
(Having) SUM (fees) > 10,000;
```

PAGE NO.:
DATE: / /

Group By Roll up

Sum (fees)
18000
15000
4000
6000

} Find Total

Total = ?

Query:

(i) Select student_name, Sum (fees) From inst_info Group By Student_name With Rollup;

(ii) Handling Null

Select IFNULL (student_name, "Total"), Sum (fees) From inst_info Group By Student_name with Rollup;

Note: Roll up is used with Group By To find Sum, count, Avg

PAGE NO. :
DATE : / /

STORED ROUTINE

A SQL statement or a set of a SQL Statements that can be stored on Databases server which can be call no. of times.

It is a type of a "FUNCTION".

(A) STORED PROCEDURE
(B) User Defined Functions

STORED PROCEDURE

These are routines that contain a series of SQL statements and procedure logic.

Often used for performing actions like Data modifications, transaction control and executing sequences of statements.

```
Drop procedure if exists p_name;

Delimeter $$

Create Procedure p_name()
Begin
    Select * From Employees
    Limit 10;
END $$

Delimiter ;
```

} Procedure

Query

Note: Show Status;

#This will display full status of SQL

Delimeter : (correct spelling)

PAGE NO.:
DATE: / /

Q.1) How To create a Procedure in MySQL?

Ans

```
Delimeter $$
Create Procedure emp_info()
Begin


END $$
```

(Skeleton Structure)

Query

```
Delimeter $$
Create Procedure emp_info()
Begin

  Select * FROM employees Order By Salary;
END $$

Delimeter ;
```

Q.2) How to call a Procedure?

Ans Syntax : Call <database>.(procedure)

eg:-> Call bank-db. emp_info()

PAGE NO. :
DATE : / /

Using Dynamic Procedure (Arguments)

```
Delimeter $$

Create procedure get_empid (IN p_fname
                            Varchar(50))
Begin

    Select emp_id from employees
    Where fname = p_fname;
END $$

Delimeter ;
```

Call get_empid ("Sham")
↳ argument

PAGE NO.:
DATE: / /

Q.] How to store result of Procedure in a variable?

Ans.

```
Delimeter $$

Create procedure get_sum (IN p_fname,
                          OUT p_sum Decimal
                          (10,2))
Begin
    Select Sum (salary) into p_sum from employees;
END $$

Delimeter ;
```

Another Query:

```
Delimeter $$
Create Procedure get_sum_by_dept
(IN p_dept Varchar(100), OUT p_sum Decimal (10,2))
Begin
    Select sum(salary) into p_sum from employees
    where dept= p_dept;
END $$

Delimeter ;
```

Now, while calling the function we are expecting Both the aurgoments.

```
Set @p_sum = 0      # Declaring a variable

Call bank_db.get_sum_by_dept ('loan', @p_sum)

Select @p_sum;      # Displaying the Result.
```

Another way

```
Set @emp_sum = 0;

Call get_sum_by_dept ("loan", @emp_sum);

Select @emp_sum;
```

↓

Display the sum.

PAGE NO.:
DATE: / /

Note:

Using Group By has two conditions.

① Appear in the Group By clause
② Be used with an aggregate function (like SUM(), Count() etc)

Some Important Statements

(i)
```
Use employee;
Select Database ();

Select @@GLOBAL.sql_mode;
```

(ii)

PAGE NO. :
DATE : / /

User Defined Functions

Example:->

```
Delimiter $$

Create Function get_sum (p_name Varchar(50))
Returns Varchar(50)      #Define return type

Deterministic No SQL Reads SQL Data

BEGIN

    Declare v_max Int;                  } Declare
    Declare v_name Varchar(50);         } Variable

    Select Max(salary) into v_max from employ
    Select fname into v_name from employ
        where Salary = v_max;

    return v_name;

END $$

Delimiter ;
```

PAGE NO.:
DATE: / /

Skeleton Structure of FUNCTION

```
Delimiter $$

Create Function emp_name_max_salary()
Returns Varchar(50)
Begin

END $$

Delimiter ;
```

ANOTHER FUNCTION (user Defined).

```
Delimiter $$

Create Function emp_name_max_salary() Returns
Varchar(50)

BEGIN
    Declare v_max INT;
    Declare v_emp_name Varchar(50);

    Select Max(salary) into v_max FROM employee;
    Select fname into v_emp_name FROM employee
        Where salary = v_max;
END $$
Delimiter ;
```

(Will give Error because nothing is returned.)

PAGE NO.:
DATE: / /

Corrected Code:

```
Begin

Declare v_max INT;
Declare v_emp_name Varchar(50);

Select max(salary) into v_max from employee;

Select fname into v_emp_name from employee
    where salary = v_max;

return v_emp_name;
END $$
```

>> Deterministic NO SQL Reads SQL DATA

This statement is necessary (Analyze Why)

PAGE NO.:
DATE: / /

WINDOW FUNCTIONS

Window functions, also known as analytic functions allow you to perform calculations across a set of Rows related to the current row.

Defined by an `OVER()` clause.

example:

```
Select
    sum(salary)
from employees;
```

But To get result for every Row.

```
Select
    sum(salary) over() as sum_salary
from employees;
```

o/p:->

379000
379000
379000
379000
379000

PAGE NO.:
DATE: / /

Query :

①
```
Select
    emp_id,
    fname,
    SUM(salary) OVER () as sum_salary
from employees;
```

②
```
Select
    emp_id,
    fname,
    SUM(salary) over (order By emp_id) as
    sum_salary
from employees;
```

Output:

101	Raju	37000	37000
102	Sham	32000	69000
103	Babarao	25000	94000
104	Paul	45000	13900
105	Alex	35000	174000
106	Rick	65000	239000
107	Leena	25000	264000
108	John	75000	339000
109	Alex	40000	379000

PAGE NO.:
DATE: / /

Q.2 Find Total sum of Salary department Wise ?

(Ans)

Note: Analyzse "Partition By" Keyword

Query

```
Select emp_id, dept, Salary,
Sum (Salary) over (Partition By emp_id)
as sum_Salary
From employees;
```

↓

(Similar to Grouping)

PAGE NO. :
DATE : / /

DEFAULT FUNCTIONS

① Row_Number()
② RANK()
③ DENSE_RANK()
④ LAG()
⑤ LEAD()

All these functions are used with over().

⑩] **Row_Number() :** Produces row_no. as per mysql

```
Select

    Row_Number()  OVER()  as  row_no,
    emp_id,
    dept,
    Salary
from  employees;
```

Using Order By

```
Select
   Row_Number()  Over (order By Salary)  as
   row_no,
   emp_id,
   dept,
   Salary      from employees;
```

PAGE NO.:
DATE: / /

Note: Row No. will be created order By Salary.

Partition Key Word :

```
Select
    Row_Number() OVER (Partition By Salary) as
    row_no,
    emp_id,
    dept,
    Salary
from employees.
```

Salary ↕ Dept

* Analyze the Difference *

1	104	Account	45000	}
2	106	Account	50000	}
3	107	Cash	30000	} Partition By dept
4	108	Cash	12000	}
5	109	Cash	15000	}
6	110	IT	18000	}

PAGE NO.:
DATE: / /

RANK

Will Help to create ranks based on given Salary or any other field.

Difference: Duplicates will have same rank.

```
Use employee;

Select
    emp_id,
    dept,
    Salary,
Dense_Rank() Over(order By Salary) as Sal_rank
from emp;
```

example:

Salary	Rank
250	1
250	1
1000	2
1000	2
2000	3
5000	4

PAGE NO.:
DATE: / /

DENSE_RANK() V/S RANK()

Salary	Rank
250	1
250	1
300	2
400	3

Salary	Rank
250	1
250	1
300	3
400	4

LAG FUNCTION

```
Select
    emp_id,
    dept,
    Salary,
    LAG(salary) over() as lag_sal from emp;
```

101	loan	37000	
102	cash	32000	37000
103	loan	25000	32000

```
Select                  | lag(salary) over(order By emp_id)
    emp_id,             | as lag_sal
    dept,               | From emp;
    Salary,             |
                        |
```

PAGE NO.:
DATE: / /

Note: IFNULL(LAG(Salary) OVER(order By emp_id), 'zero') As Lag_Sal;

LEAD FUNCTION

It is exactly opposite to Lag Function.

```
Select
    emp_id,
    dept,
    Salary,
    Lead(salary) Over() as lag_sal
from employees;
```

#All these are used to find the Difference between two salaries.