1. Check constraint
2. Date time
3. Numerical function
4. Limit clause
5. Self join
6. Acid , rollback and commit
7. Views
8. Temporary tables
9. Select case

<https://www.mysqltutorial.org/mysql-string-functions/>

insert into employee values(6,'rameya','345234','Delhi','India','rameya@test.com');

insert into employee values(7,nisha,'345234','Delhi','India',nisha@test.com');

# constraint

1. create table supplier(suppid int PRIMARY key AUTO\_INCREMENT,

suppname varchar(50) not null,

status int DEFAULT 0,

city varchar(50));

1. alter table supplier modify city varchar(50) not null;
2. INSERT INTO supplier(suppname, city) values('Haldiram','Nagpur');
3. alter table shipment add CONSTRAINT FOREIGN key (itemno) REFERENCES item(itemno);
4. create TABLE shipment (id int PRIMARY KEY,

itemno int,

suppid int,

qty int,

FOREIGN KEY(itemno) REFERENCES item(itemno),

FOREIGN KEY(suppid) REFERENCES supplier(suppid)

);

create table employee(eid int PRIMARY key AUTO\_INCREMENT,

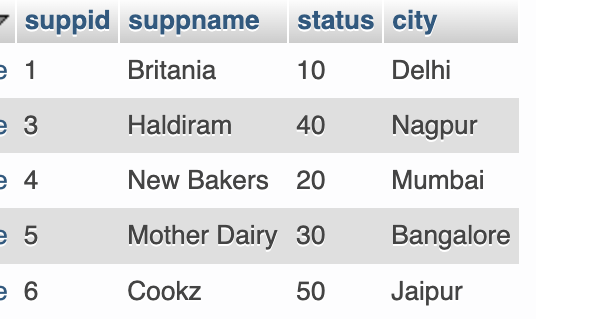
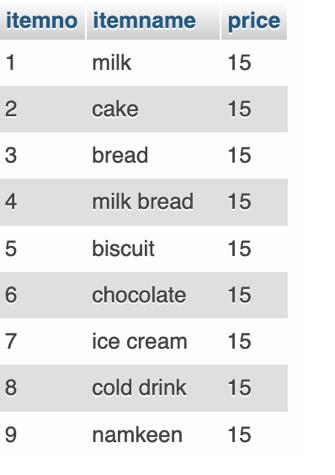
ename varchar(50) not null,

phone varchar(20) UNIQUE,

city varchar(50) not null,

country varchar(50) DEFAULT 'India',

email varchar(50) not null UNIQUE);



SELECT QUERIES

1. PROJECTION : fetch data for specific columns  
   select ename from employee;
2. after as keyword give the alias name. If alias name is more than 1 word then wrap it within quotes  
   select email as 'Email Id', ename as Name from employee;
3. To get those employees name whose phone is not null  
   select ename, phone from employee WHERE phone is NOT null;
4. To get those employees whose phone is null  
   select ename, phone from employee WHERE phone is null;
5. Get movie title for all the movies directed by director id 1
6. To not get duplicate values for column city  
   SELECT DISTINCT city from supplier
7. Mathematical operations with select  
   SELECT 1+2 as 'Addition of 1 and 2';
8. To get the current date  
   SELECT curdate();
9. To substitute for null value us ifnull  
   select ifnull(phone,'NOT PROVIDED') , ename , email from employee;

HR\_DATA provided by GL

1. # put text in query output

select employee\_id ,'gets',salary,'/-' from emp\_payment;

RELATIONAL OPERATORS

<, >, <=, >=, =, <>

1. Get employee data for salary >= 20000  
   SELECT first\_name, email, salary FROM `employees` WHERE salary>=20000;
2. SELECT first\_name, email, salary, department\_id FROM `employees` WHERE department\_id <>50;

LOGICAL OPERATOR

OR AND NOT

1. SELECT first\_name, email, salary FROM `employees` WHERE salary>=20000 && salary <= 30000;  
   SELECT first\_name, email, salary FROM `employees` WHERE salary>=20000 and salary <= 30000;
2. SELECT first\_name, email,job\_id, salary FROM `employees` WHERE job\_id = 'AD\_VP' and salary >20000;

BETWEEN

1. SELECT first\_name, email, salary FROM `employees` WHERE salary BETWEEN 20000 and 30000;

IN – condition based on a list

1. SELECT cust\_name, city FROM `customer` WHERE city In ('Mumbai','Delhi')
2. List firstname and last name of all the employess whose manager id is 1, 6 and 52

PATTERN MATCHING

1. % matches any substring
2. \_ matches any character
3. LIKE
4. Customer info whose name starts with r  
   SELECT \* FROM `customer` WHERE cust\_name LIKE 'r%';
5. Customer info whose email ends with techgatha.com  
   SELECT email, customer\_id FROM `customer` WHERE email LIKE '%techgatha.com';
6. Customer info whose email contains the word test  
   SELECT email, customer\_id FROM `customer` WHERE email LIKE '%test%';
7. Escape sequence this is \   
   SELECT customer\_id, email FROM `customer` WHERE email LIKE '%\\_%'
8. Display all employee details whose job id ends with ‘MGR’

**Regular expressions – pattern matching [ FYI ]**

<https://dev.mysql.com/doc/refman/8.0/en/regexp.html>

1. Character classes
   1. ^ - starts with
   2. $ - ends with
2. Quantifiers ->
   1. ? – 0 or one
   2. \*
   3. +
3. Customer info whose gst starts with any alphabet only  
   SELECT email, gst FROM `customer` WHERE gst REGEXP '^[a-zA-Z]'
4. Customer info whose name starts with R or P  
   SELECT cust\_name FROM `customer` WHERE cust\_name REGEXP '^[RP]'
5. Customer info whose gst starts with digits only  
   SELECT cust\_name, gst FROM `customer` WHERE gst REGEXP '^[0-9]'
6. Match the string that has exactly 0 or 1 occurrence of u

SELECT 'color' REGEXP 'colou?r'; -> 1

SELECT 'colour' REGEXP 'colou?r'; -> 1

SELECT 'colouur' REGEXP 'colou?r'; -> 0  
#SELECT 'color' REGEXP 'colou?r';

#SELECT 'colouur' REGEXP 'colou\*r';

#SELECT 'color' REGEXP 'colou\*r';

#SELECT 'colouur' REGEXP 'colou+r';

SELECT 'color' REGEXP 'colou+r';

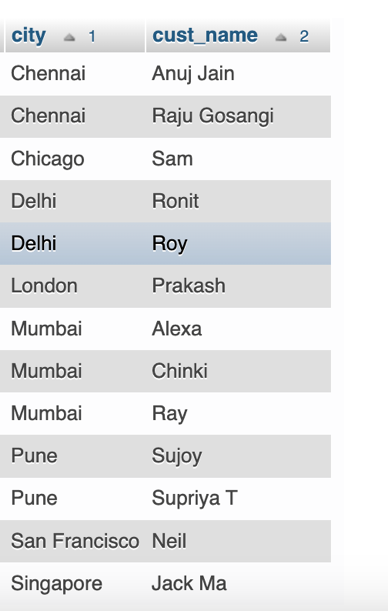
ORDER BY CLAUSE

Sorting in asc or desc order

Default is ascending

SELECT cust\_name, city FROM `customer` order BY city

For descending order

SELECT cust\_name, city FROM `customer` order BY city desc  
  
SELECT city, cust\_name FROM `customer` ORDER BY city, cust\_name  
  
select city, cust\_name from customer order by city;  
  
 

select city, cust\_name from customer order by city, cust\_name;

STRING FUNCTIONS

<https://dev.mysql.com/doc/refman/8.0/en/string-functions.html>

1. Concat  
   SELECT concat(cust\_name,' ',customer\_id) as Name from customer;

If gst is null replace with the text 'NO GST'  
SELECT email , concat(cust\_name,' ',ifnull(gst,'NO GST')) as Name from customer;  
  
If gst is null replace with the text empty space else the value for gst  
  
SELECT email , concat(cust\_name,' ',if(gst is null,'',gst)) as Name from customer;

1. Concat\_ws : concatenate with separator  
   SELECT email , concat\_ws(',',cust\_name, if(gst is null,'NO GST',gst)) as Name from customer;
2. length  
   SELECT cust\_name, length(cust\_name) as 'NO OF CHARACTERS' FROM customer
3. substring\_index(str, delimeter, n)  
   Returns a part of string before the 2nd occurrence of ‘,’  
   SELECT substring\_index('shalini,mittal,mumbai,12312323',',',2);
4. replace  
   SELECT replace('shalini mittal,mumbai,12312323','mittal','gupta');
5. how many characters from the right or left  
   SELECT right('shalini mittal',7);  
   SELECT left('shalini mittal',3);
6. trim, ltrim, rtrim  
   SELECT ltrim(' shalini mittal ');
7. repeat  
   SELECT repeat('\*',3);
8. substring : index of a string starts from 1  
   SELECT substring('sky is blue', 5);  
   start from index 5 and return 2 character  
   SELECT substring('sky is blue', 5, 2);

AGGREGATE FUNCTIONS

1. count -> returns number of records in a table

select count(\*) as count,

sum(price) as sum,

min(price) as min,

max(price) as max,

avg(price) as avg

from item;

GROUP BY clause

Groups a set of rows and give a summary of the same.

It returns 1 row for each group

Used in conjunction with aggregate functions

Returns the number of customers in every city

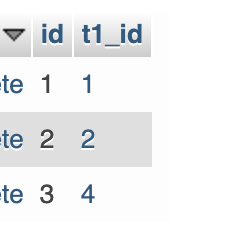
SELECT count(city), city from customer group by city;

#display the total amount of purchase made by each customer

select customer\_id, sum(order\_total) from order\_table GROUP BY customer\_id;

JOINS

  
Join is used to fetch data from more than 1 tables



Assume table t1 and t2

1. cartesian product / cartesian join - all possible combination from both the table  
   row1 of t1 \* all rows of t2  
   row2 of t1 \* all rows of t2
2. inner join / equi join-> common between t1 and t2  
     
   SELECT \* FROM t1, t2 where t1.id = t2.t1\_id; // cannot use ON with comma separated list of tables  
   SELECT \* FROM t1 JOIN t2 where t1.id = t2.t1\_id;  
   SELECT \* FROM t1 JOIN t2 ON t1.id = t2.t1\_id;  
     
   To get specific columns from two tables. Below query gives ambiguous error as id is a common column name in both t1 and t2. Hence need to specify the table from which to pick the data for that column  
   SELECT id, name FROM t2,t1 WHERE t1.id = t2.t1\_id;

Below does resolves previous issue as t1\_id is only one of the table  
SELECT t1\_id, name FROM t2,t1 WHERE t2.t1\_id = t1.id;

Alternatively use the table name to specify column of which table for ambigous column names  
SELECT t1.id, name FROM t2,t1 WHERE t2.t1\_id = t1.id;

Can also create aliases for table names  
SELECT b.id, name FROM t2 as a,t1 b WHERE a.t1\_id = b.id;

1. left join -> t1 join t2  
    all data of t1 and common between t1 and t2  
     
   SELECT a.id as 'T1 ID',name, b.id as 'T2 ID', t1\_id as 'FK OF T1'

from

t1 a

LEFT JOIN

t2 b

ON a.id = b.t1\_id

1. right join-> t1 join t2  
    all data of t2 and common between t1 and t2  
     
   SELECT a.id as 'T1 ID',name, b.id as 'T2 ID', t1\_id as 'FK OF T1'

from

t1 a

RIGHT JOIN

t2 b

ON a.id = b.t1\_id

1. full join -> common and uncommon between t1 and t2

SELECT a.id as 'T1 ID',name, b.id as 'T2 ID', t1\_id as 'FK OF T1'

from

t1 a

LEFT JOIN

t2 b

ON a.id = b.t1\_id

UNION

SELECT a.id as 'T1 ID',name, b.id as 'T2 ID', t1\_id as 'FK OF T1'

from

t1 a

RIGHT JOIN

t2 b

ON a.id = b.t1\_id

Student and course table

1. Get course name , course fee , email, phone from course and student  
   select course\_name, course\_fee, email, phone from courses c, student s  
   where c.course\_id = s.selected\_course

SUBQUERIES

1. Get names of courses for which students have registered

select DISTINCT Course\_name from courses t1, student t2

where t1.course\_id=t2.selected\_course;

OR

get the names of course for which students have registered

SELECT course\_name

from courses where course\_id IN

(SELECT DISTINCT selected\_course from student);

1. get the names of course for which students have not registered

SELECT course\_name

from courses where course\_id NOT IN

(SELECT DISTINCT selected\_course from student); # [1,2,3]

HAVING CLAUSE

Is used to apply conditions along with the group by clause

1. count of students joined from different sources  
   SELECT COUNT(source\_of\_joining) , source\_of\_joining

from student

GROUP by source\_of\_joining

1. count of students joined from different sources where count is <=5  
   SELECT COUNT(source\_of\_joining) , source\_of\_joining

from student

GROUP by source\_of\_joining

HAVING COUNT(source\_of\_joining) <=5

1. filter by location

SELECT COUNT(source\_of\_joining) as total , source\_of\_joining, location

from student

where LOCAtion ='Delhi'

GROUP by source\_of\_joining

HAVING total <=5

SELF JOIN

CLASSIC MODELS

SELECT e.firstName FROM `employees` e

(SELECT firstName from employees where employeeNumber =

select reportsTo

from employees where employeeNumber=1056 )

select e.employeeNumber, e.firstName as'REPORTS TO',

m.employeeNumber, m.firstName as Manager

from employees e , employees m

where e.reportsTo = m.employeeNumber limit 10;

select e.employeeNumber, e.firstName as'REPORTS TO',

m.employeeNumber, m.firstName as Manager

from employees e left join employees m

on e.reportsTo = m.employeeNumber limit 10;

TRANSACTION AND ACID PRINCIPLES

start TRANSACTION;

insert into t1 values(10,'N10');

insert into t2 values(5,3);

commit; // or rollback;

set autocommit=0;

set autocommit=1;

LIMIT CLAUSE

Used for pagination.

First 5 records

SELECT \* from customer order by city limit 5 ;

5 records from row 3

SELECT \* from customer order by city limit 5 OFFSET 2;

CHECK CONSTRAINT / COMPOSITE KEY

Applied on a column to keep a check on range of values

Below query, while inserting data will only allow the country value to be either of USA, UK or India.

Composite key – combinatoion of more than 1 key

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL ,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL CHECK (country IN ('USA','UK','India')),

home\_city varchar(25) NOT NULL,

PRIMARY KEY (aut\_id,home\_city));

TEMPORARY TABLE:

Are just stored in temporary memory and deleted automatically when the session is destroyed.

create temporary table test(id int primary key);

VIEWS:

View can be created for the select queries which are repeatedly used and then executed using the viewname

SYNTAX:

Create View <viewnmae> as

<query>

Create view customerinfo

As

Select cust\_name, email, phone, order\_id from customer , order\_table

where customer.customer\_id = order\_table.customer\_id

To execute the view

Select \* from customerinfo;

create view iteminfo AS

SELECT item\_name, manufacturer

from item WHERE item\_id =item\_id

ALTER THE VIEW

alter view customerinfo AS

Select cust\_name, email, phone,city, order\_id, order\_total from customer , order\_table

where customer.customer\_id = order\_table.customer\_id

DROP THE VIEW

drop view customerinfo;

DATE TIME

SELECT \* FROM `order\_table` WHERE date(order\_date)='2021-03-09'

YYYY-MM-DD

SELECT now()

SET @dt = now();

SELECT @dt , year(@dt);

YYYY-MM-DD HH:MM:SS

**DATE FUNCTIONS:**

SET @dt = NOW();

Select @dt;

Select date(@dt);

Select time(@dt);

SELECT

HOUR(@dt),

MINUTE(@dt),

SECOND(@dt),

DAY(@dt),

WEEK(@dt),

MONTH(@dt),

QUARTER(@dt),

YEAR(@dt);

MySQL DATE\_FORMAT function

To format a DATETIME value, you use the DATE\_FORMAT function. For example, the following statement formats a DATETIME value based on the %H:%i:%s - %W %M %Y format:

SELECT DATE\_FORMAT(@dt, '%H:%i:%s - %W %M %Y');

**MySQL DATE\_ADD function**

To add an interval to a DATETIME value, you use DATE\_ADD function as follows:

SELECT @dt start,

DATE\_ADD(@dt, INTERVAL 1 SECOND) '1 second later',

DATE\_ADD(@dt, INTERVAL 1 MINUTE) '1 minute later',

DATE\_ADD(@dt, INTERVAL 1 HOUR) '1 hour later',

DATE\_ADD(@dt, INTERVAL 1 DAY) '1 day later',

DATE\_ADD(@dt, INTERVAL 1 WEEK) '1 week later',

DATE\_ADD(@dt, INTERVAL 1 MONTH) '1 month later',

DATE\_ADD(@dt, INTERVAL 1 YEAR) '1 year later';

ADDDATE(date,INTERVAL expr unit)

Or

ADDDATE(expr,days)

**MySQL DATE\_SUB function**

To subtract an interval from a DATETIME value, you use DATE\_SUB function as follows:

SELECT @dt start,

DATE\_SUB(@dt, INTERVAL 1 SECOND) '1 second before',

DATE\_SUB(@dt, INTERVAL 1 MINUTE) '1 minute before',

DATE\_SUB(@dt, INTERVAL 1 HOUR) '1 hour before',

DATE\_SUB(@dt, INTERVAL 1 DAY) '1 day before',

DATE\_SUB(@dt, INTERVAL 1 WEEK) '1 week before',

DATE\_SUB(@dt, INTERVAL 1 MONTH) '1 month before',

DATE\_SUB(@dt, INTERVAL 1 YEAR) '1 year before';

**MySQL DATE\_DIFF function**

To calculate a difference in days between two DATETIME values, you use the DATEDIFF function. Notice that the DATEDIFF function only considers the date part of a DATETIME value in the calculation.

return the days between two datetime expressions 2008-05-17 11:31:31 and 2008-04-28.

select datediff('2023-02-22',@dt); // will give negative result

select datediff(@dt, '2023-02-22'); // will give positive result

Table of format specifiers

|  |  |
| --- | --- |
| **Name** | **Description** |
| %a | Abbreviated weekday name (Sun..Sat) |
| %b | Abbreviated month name (Jan..Dec) |
| %c | Month, numeric (0..12) |
| %D | Day of the month with English suffix (0th, 1st, 2nd, 3rd, …) |
| %d | Day of the month, numeric (00..31) |
| %e | Day of the month, numeric (0..31) |
| %f | Microseconds (000000..999999) |
| %H | Hour (00..23) |
| %h | Hour (01..12) |
| %I | Hour (01..12) |
| %i | Minutes, numeric (00..59) |
| %j | Day of year (001..366) |
| %k | Hour (0..23) |
| %l | Hour (1..12) |
| %M | Month name (January..December) |
| %m | Month, numeric (00..12) |
| %p | AM or PM |
| %r | Time, 12-hour (hh:mm:ss followed by AM or PM) |
| %S | Seconds (00..59) |
| %s | Seconds (00..59) |
| %T | Time, 24-hour (hh:mm:ss) |
| %U | Week (00..53), where Sunday is the first day of the week |
| %u | Week (00..53), where Monday is the first day of the week |
| %V | Week (01..53), where Sunday is the first day of the week; used with %X |
| %v | Week (01..53), where Monday is the first day of the week; used with %x |
| %W | Weekday name (Sunday..Saturday) |
| %w | Day of the week (0=Sunday..6=Saturday) |
| %X | Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V |
| %x | Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v |
| %Y | Year, numeric, four digits |
| %y | Year, numeric (two digits) |
| %% | A literal “%” character |
| %x | x, for any “x” not listed above |