-------to find sum,min,max,avg of sal for each department for all employees with bonus= sal\*0.15 and if I want to display all rows with bonus>700

10 325345

20 23453453

30 3453

select deptno,sum(sal),min(sal),max(sal),avg(sal)

from emp

where sal\*0.15>700

group by deptno

order by deptno

product

prno,pname,price,qty, discount =price\*8%

--------to display min of sal+comm for each job if count of the job is > 3

select job,min(sal+ifnull(comm,0)) ,count(\*) count\_F

from emp

group by job

having count(\*)>3

order by job

----List all names and job so that the length of name should be 15 if it is smaller then add spaces to left.

name ----10

select ename,lpad(ename,15,” “)

from emp;

or

select concat(space(15-length(ename)),ename)

from emp;

28) Display thousand separator and $ symbol for commission, if it is null then display it as 0 for all employees whose name starts with A and ends with N

12 ksdhfsh 23434 sdhkjshd 23,456,675

select ename,sal, concat(‘$’,format(ifnull(comm,0)))

from emp

where ename like ‘A%N’

or

select ename,sal, concat(‘$’,format(ifnull(comm,0)))

from emp

where ename like ‘^A.\*N$’

select ename,sal,format(sal,2,en\_US)

13,234.00

1356,67.00

select ename,sal,format(sal,2,en\_UK)

192.34.45.101

Group by, having,aggregate function

DML ---insert , update, delete

TCL ---- commit, rollback, savepoint

case statement

DDL -data definition language

if comm is null then need improvement

comm<500 ok

comm>=500 and <1000 good

com>=1000 excellent

select empno,ename,sal,comm,case when comm is null then ‘need improvement’

when comm<500 then ‘ok’

when comm<1000 then ‘good’

else ‘excellent’ end “Remark”

from emp;

------display pid,pname,price from product table

price<50 less expensive

>=50 <100 moderate

>=100 expenssive

if price is null wrong price

select pid,pname,price,case when price is null or price=0 then ‘wrong price’

when price<50 then ‘less expenssive’

when price>=50 and price<100 then ‘moderate’

else ‘expenssive’ end “remark”

from product;

-------display pid,pname,price,cid

if cid=1 then chips

cid=2 biscuits

cid =3 snackes

otherwise others

select pid,pname,price,case when cid=1 then ‘chips’

when cid=2 then ‘biscuits’

when cid=3 then ‘snacks’

else ‘others’ end “category”

from product;

or

select pid,pname,price,case cid when 1 then ‘chips’

when 2 then ‘biscuits’

when 3 then ‘snacks’

else ‘others’ end “category”

from product;

-----to display grade level based on job

select empno,ename,job,case when job in ('CLERK','salesman') then 'grade 1'

-> when job='Manager' then 'grade 2'

-> when job in ('ANALYST','designer') then 'grade 3'

-> when job='president' then 'grade 4' end "grade"

-> from emp;

DDL statements

create table -----new table

alter table ----- modify existing table

drop table------ delete table

truncate-------delete only data

all DDL statements are autocommit

------to delete entire table

drop table emp;

drop table dept;

------delete all records but keep empty table

truncate table emp;

or

delete

from emp;

|  |  |
| --- | --- |
| Truncate | Delete |
| DDL | DML |
| Autocommit --🡪 no rollback | Rollback is possible till we commit; |
| Cannot use where condition | We may use where condition to remove few rows |
| Nested query cannot be used | Nested query is possible to use |

**Data types in mysql**

* Numeric
* Date and Time
* String Types.
* enum

**Numeric Data Types**

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions −

* **INT**− A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
* **TINYINT** − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
* **SMALLINT** − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
* **MEDIUMINT** − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
* **BIGINT** − A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
* **FLOAT(M,D)** − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
* **DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
* **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M)

**Date and Time Types**

The MySQL date and time datatypes are as follows −

* **DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
* **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
* **TIMESTAMP** − A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 ( YYYYMMDDHHMMSS ).
* **TIME** − Stores the time in a HH:MM:SS format.
* **YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

**String Types**

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

* **CHAR(M)** − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.

**citycode** ------1000 ----char(5)

**Asc**

**asr**

* **VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.

city name varchar(30)

pune -------30 4

Calcutta -------30 8

Mumbai ------30 6

dkjk jk sjdkfljskldsjfks jdlkfjsdklj ----30

* **BLOB or TEXT** − A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.
* **TINYBLOB or TINYTEXT** − A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
* **MEDIUMBLOB or MEDIUMTEXT** − A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
* **LONGBLOB or LONGTEXT** − A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
* **ENUM** − An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

create table mytab(id int,name varchar(10));

constraints

2 types of constraints

1. table level –primary key,unique,check,foreign key auto\_increment is technique which allows to insert unique values.
2. field level ---- not null , default

table level constraints can be written immediate after the field or at the end in create table query

field level constraints has to be written immediate after the field

create table myemployee(empid int primary key,

ename varchar(30) not null,

price decimal(11,2) default 100,

qty int check(qty>10),

passportnum int unique,

adharnum int unique not null)

product(pid,pname,qty,price,cid)

category(catid,cname,desc)

create table category(catid int,

cname varchar(20) not null,

description varchar(20),

primary key(catid));

create table product(

pid int primary key,

pname varchar(20) not null,

qty int check(qty>0),

price decimal(11,2) check(price>0),

cid int,

foreign key fk\_cid(cid) references category(catid)

on delete set null

on update cascade)

faculty(fid,fname,skill)

room(rid,rname,loc)

course(cid,cname,fid,rid)

create table faculty(

fid int primary key,

fname varchar(20) not null,

skill varchar(20)

)

room(rid,rname,rloc)

create table room(

rid int,

rname varchar(20),

rloc varchar(50)

primary key(rid)

)

course(cid,cname,fid,rid)

create table course(

cid int primary key,

cname varchar(20),

fid int,

rid int,

foreign key fk\_fid(fid) references faculty(fid)

on delete set null

on update cascade,

foreign key fk\_rid(rid) references room(rid)

on delete set null

on update cascade

)