1. Find all salesman in each department
   select deptno,count(*)
   -> from emp
   -> where job='SALESMAN'
   -> group by deptno;

2. to find how many employees are working under same manager;

Select mgr, count(*)

From emp

Group by mgr;

3. To find max, min and sum of salary for all employees whose sal > 1500 and < 3000
   Select max(sal),min(sal),sum(sal)
   From emp
   Where sal between 1501 and 2999
4. Display all departments whose avg sal > 2000
   Select deptno,avg(sal)
   From emp
   Group by deptno
   Having avg(sal)>2000;

5. Display all mgr under whom more than 2 analyst or salesman are working
   Select mgr,count(*)
   From emp
   Where job in ('ANALYST','SALESMAN')
   Group by mgr
   Having count(*) > 2;

   Types of keys in database
   1. Primary key---- minimal set of columns which identifies the row uniquely is called as primary key.
   2. Candidate key--- Every possible combination which is minimal and identifies the row uniquely is called as candidate key

Studid	Sname	Adhar card	Passport	Mobile	email

Candicate keys ---- studid, adharcard,passport,mobile,email
Primary key ----studid

Alternate key --- all candidate keys which are not considered as primary key is called as alternate key
Super key---any combination which identifies the row uniquely is called as super key

Foreign key-----when we store the data in a table is we need to refer data in some other column of same table or different table then it is called as foreign key.

a/c id	Custid	Type of a/c	balance
1	1	Saving	34567
2	1	Demat	456788
1	1	Saving	5555555555

Customer

Customerid	Cname	Mobile	email
1	Kishori	98222222	jhasgdhg@ksjdh

Primary key –customerid--- customer table a/c id --→account table
Candidate key
Customerid, mobile, email--- customer table , a/c id --→account table
Super key----any combination with custemerid or mobile or email
For account table any combination with a/c id
Ac id+custid, a/cid+type, a/cid+balance,a/cid+custid+balance
Foreign key --- custid in account table references customerid in customer table, so it is foreign key in account table

Using DDL

1. Primary key
    - It is table level constraint
    - There can be only one primary key to a table
    - It cannot be null, and it should be unique
2. Check
   Before storing value in the column if you want to check any condition then use check constraint
3. Unique
   It is a table level constraint
   There can be more than one unique key
   It can store any number null values
4. Not null
   It is a field level constraints.
   It do not allow to store null value in the column
   Duplicates are allowed
5. Default
   It is a field level constraint
   If user enters null value in the column then null value will be replaced by some default value
6. Foreign key
   If a column references other column for correctness of data then it is called as foreign key
   mysql> create table student_2(

```
   -> studid int primary key,
   -> sname varchar(20),
   -> marks int);
Query OK, 0 rows affected (0.06 sec)

mysql> insert into student_2 values(100,'Rajan',99);
Query OK, 1 row affected (0.00 sec)

mysql> insert into student_2 values(100,'Rajan',99);
ERROR 1062 (23000): Duplicate entry '100' for key 'student_2.PRIMARY'
mysql> insert into student_2 values(101,'Revati',99);
Query OK, 1 row affected (0.01 sec)

mysql> insert into student_2(studid,sname) values(102,'Revati');
Query OK, 1 row affected (0.00 sec)

mysql> insert into student_2(sname,studid) values('Revati',103);
Query OK, 1 row affected (0.00 sec)

mysql> select * from student_2;
+--------+--------+-------+
| studid | sname  | marks |
+--------+--------+-------+
|    100 | Rajan  |    99 |
|    101 | Revati |    99 |
|    102 | Revati |  NULL |
|    103 | Revati |  NULL |
+--------+--------+-------+
4 rows in set (0.00 sec)

mysql> insert into student_2 values(104,'Rajan',-99);
Query OK, 1 row affected (0.00 sec)

mysql> drop table student_2;
Query OK, 0 rows affected (0.02 sec)

mysql> create table student_2(
   -> studid int primary key,
   -> sname varchar(20),
   -> marks int check(marks between 0 and 100));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into student_2 values(101,'Revati',99);
Query OK, 1 row affected (0.02 sec)

mysql> insert into student_2 values(100,'Rajan',99);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into student_2 values(102,'Rajan',-99);
ERROR 3819 (HY000): Check constraint 'student_2_chk_1' is violated.
mysql> insert into student_2 values(102,'Rajan',123);
ERROR 3819 (HY000): Check constraint 'student_2_chk_1' is violated.
Create table student(
Stdid int primary key,
Sname varchar(20),
Marks int check(marks>0))

112  xxx   -23
112  xxxx  99
112  xxx   78
```

Data types in mysql

- Numeric
- Date and Time
- String Types.

Let us now discuss them in detail.

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.
- **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.

- **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.

1. Create a product table to store product id,name,qty and price
   Product id is primary key
   Name should be unique and not null
   Qty should not be null, but if user does not give the values then by default 10
   Price cannot be -ve

   Create table product_2(
   Prodid int primary key,
   Pname varchar(30) not null unique,
   Qty int default 10,
   Price decimal(9,2) check(price>0),
   Mfgdate date);

   insert into product_2(prodid,pname,price) values(11,'lays_onion',-34);
   ERROR 3819 (HY000): Check constraint 'product_2_chk_1' is violated.

   insert into product_2 values(10,'lays',23,40.00,'2022-10-30');
   Query OK, 1 row affected (0.01 sec)

   mysql> insert into product_2(prodid,pname) values(11,'lays_onion');
   Query OK, 1 row affected (0.01 sec)

   insert into product_2(prodid,pname) values(11,'lays_onion');
   ERROR 1062 (23000): Duplicate entry '11' for key 'product_2.PRIMARY'
   mysql> insert into product_2(prodid,pname) values(12,'lays_onion');
   ERROR 1062 (23000): Duplicate entry 'lays_onion' for key 'product_2.Pname'
   mysql> insert into product_2(prodid,pname) values(12,'nachos');
   Query OK, 1 row affected (0.00 sec)

a/c id	Custid	Type of a/c	balance
1	1	Saving	34567
2	1	Demat	456788
1	1	Saving	5555555555

Customer

Customerid	Cname	Mobile	email
1	Kishori	98222222	jhasgdhg@ksjdh

```
Create table customer_2(
Custid int primary key,
Cname varchar(20) not null,
Mobile int unique,
Email varchar(30));

Create table account_2(
Acid int primary key,
Cid int,
Type varchar(30),
Balance double(9,2) check(balance>1000),
Constraint fk_cid_2 foreign key(Cid) references customer_2(custid))
```

Studid	Cname	Marks
1	Java	95
1	Database	97
2	Java	96
2	Database	98

When a table contains more than one column in primary key, then it is called as composite primary key.

```
Create table student_3(
Studid int,
Cname varchar(20),
Marks int,
Constraint pk_id_nm primary key(studid,cname));

Create table mytab_3(
Id int primary key auto_increament,
Name varchar(30));
```

Statements can be used with foreign key
On delete <action>
On update <action>

1. No action ----- donot allow changes or deletion in parent table, unless there is no dependent child row
2. Cascade--- delete corresponding child rows if you delete parent row, update corresponding child rows if parent rows are updated
3. Set null---- if we delete parent table entry then change corresponding column in child row entry to null, and same for update