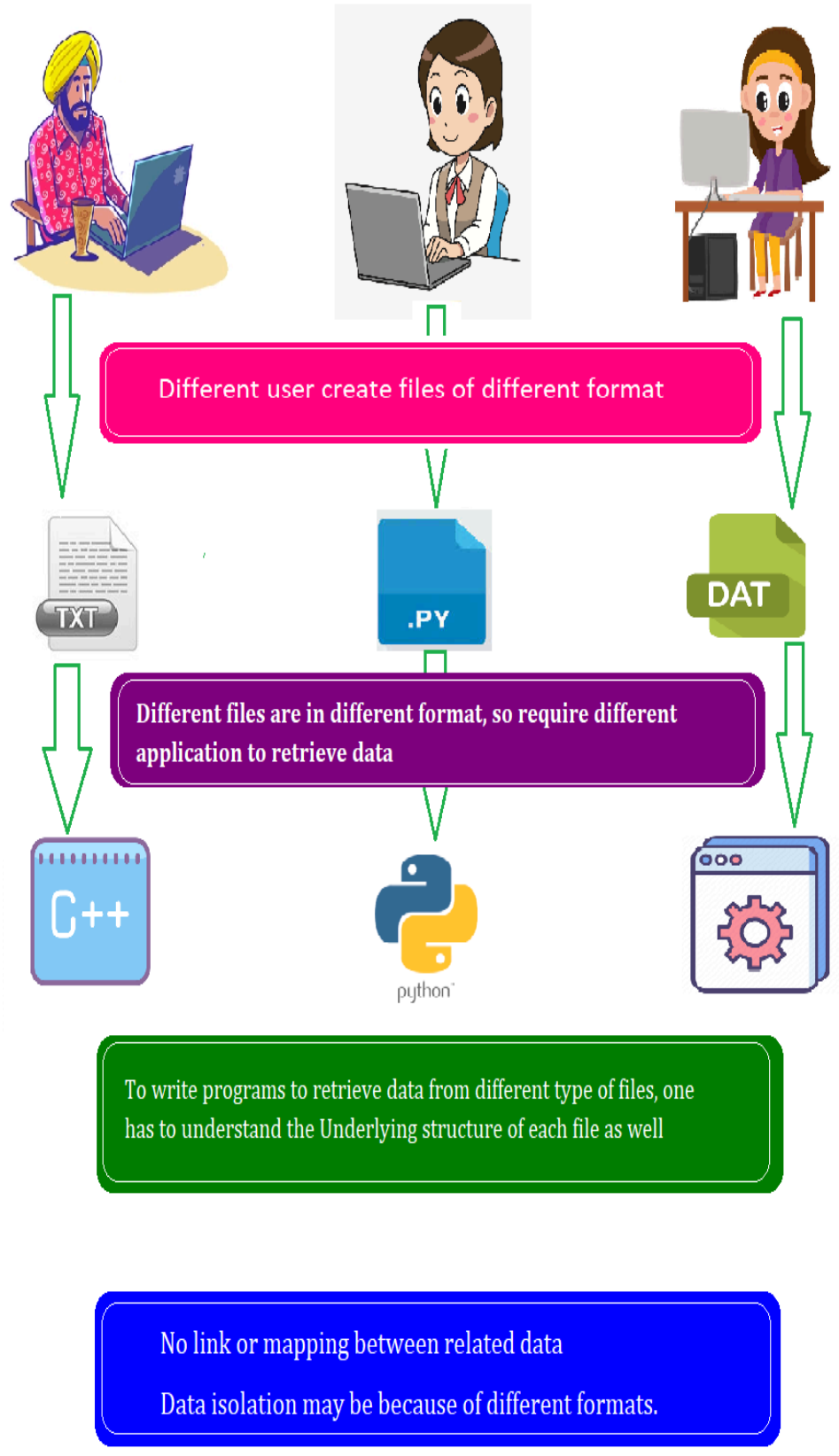
Database Concepts

FILE SYSTEM

file:- container to store data in a computer.

- stored on the storage device of a computer system.



LIMITATIONS OF A FILE SYSTEM

(A) Difficulty in Access:-Accessing data is not convenient and efficient
(B) Data Redundancy:- data are duplicated / multiple copy of same data
(C) Data Inconsistency:- two entry about same data does not match
(D) Data Isolation:- data mapping is not supported in the file system.

:-no link or mapping between related data
:-isolation may be of different formats.
:-difficult to write programs to retrieve data from different files as one has to understand the underlying structure of each file as well

(E) Data Dependence:- Data is stored in a specific format or structure in a file. updating the structure of a data file requires modification in all the application programs accessing that file

(F) Controlled Data Sharing:-Data are scattered in various files. Also different files may have different formats so it is difficult to share data among different applications. very difficult to enforce access control in a file system while accessing files through application programs

DBMS

Database:- collection of interrelated data
-computer based record keeping system
-serve multiple application

DBMS:-(database management system):- a software that creates and manages databases. DBMS serves as an interface between the database and end users.

Example:- MySQL, Oracle, PostgreSQL,
SQL Server, Microsoft Access, MongoDB

Table 8.3 Use of Database in Real-life Applications

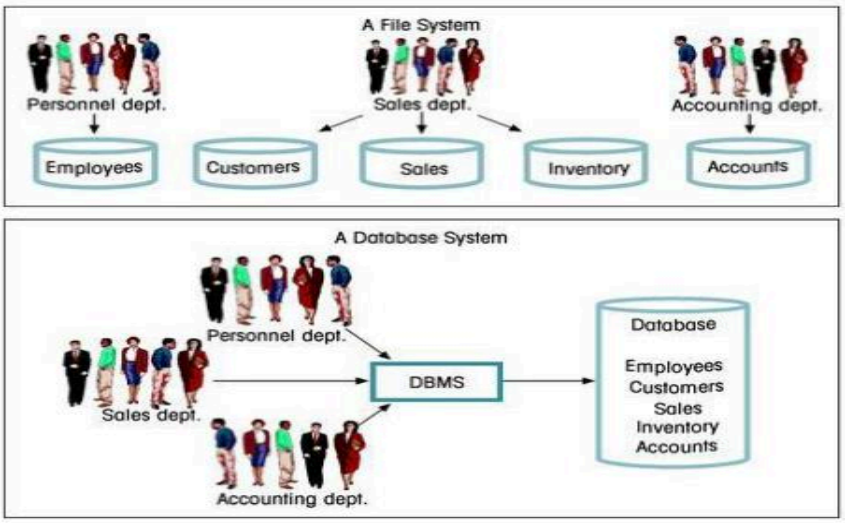
Application	Database to maintain data about
Banking	customer information, account details, loan details, transaction details, etc.
Crop Loan	kisan credit card data, farmer's personal data, land area and cultivation data, loan history, repayment data, etc.
Inventory Management	product details, customer information, order details, delivery data, etc.
Organisation Resource Management	employee records, salary details, department information, branch locations, etc.
Online Shopping	items description, user login details, users preferences details, etc.

BENEFIT OF DBMS

i) Reduce Data redundancy :- data are duplicated
ii) Reduce Data inconsistent :- two entries do not agree
iii) Data Independence:- modification in scheme at one level not affect scheme in next higher level

FILE SYSTEM VS DBMS

File Systems vs Databases



KEY CONCEPT OF DBMS

Data Constraint:- restrictions or limitations on data that can be inserted in one or more columns of a table

Meta Data(Data Dictionary) :- data about the data

Database Instance:-snapshot of the database at any given time

Query: - request to a database for obtaining information in a desired way.

querying the database:- Retrieving data from a database through special type of commands

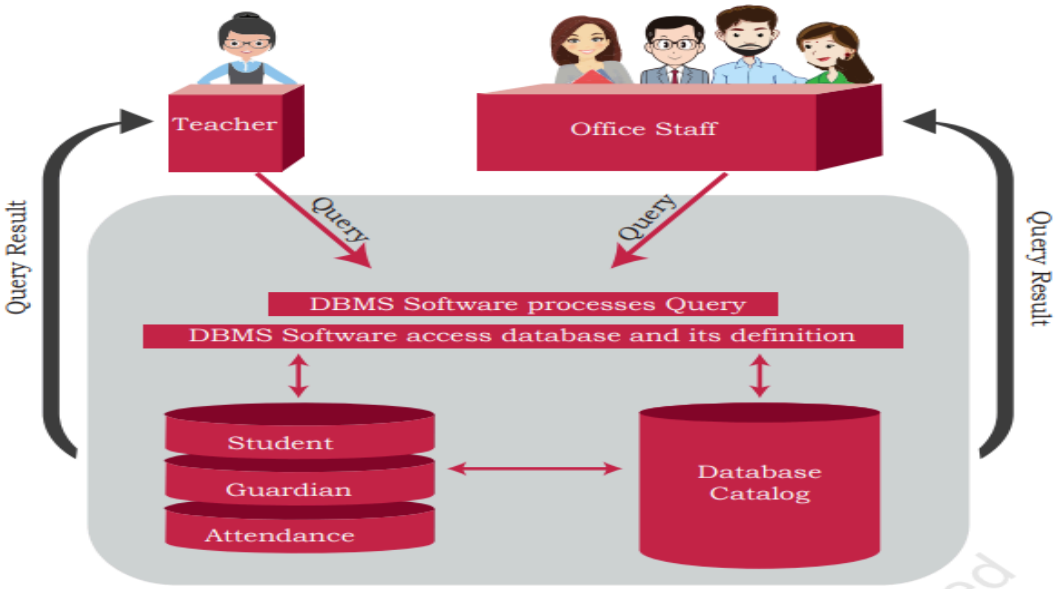


Figure 8.2: STUDENTATTENDANCE database environment

Data Model:- describes the structure of the database
:- how data are defined and represented
:- relationships among data

:- the constraints.

Database Schema:- is the design of a database
:-skeleton of the database
:- depend on data model
:-visual or logical architecture as it tells us how the data are organised in a database

Relational data model:- data arranged into row and column
RDBMS:-Relational database management :- MySQL, Microsoft SQL Server, PostgreSQL, Oracle
Relation : - table - data arranged into and column
Three Important Properties of a Relation:-

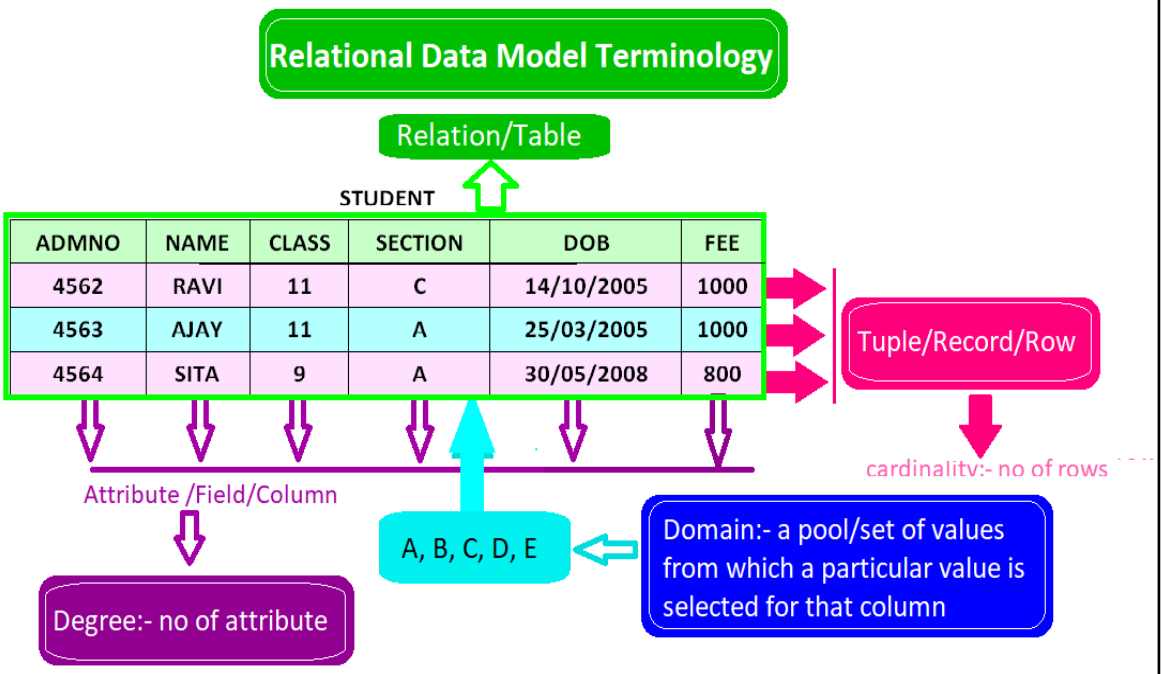
Property 1: imposes following rules on an attribute/column of the relation.
• Each attribute/column in a relation has a unique name.
• Sequence of attributes in a relation is immaterial.

Property 2: governs following rules on a tuple/row of a relation.
• Each tuple/row in a relation is distinct. Each tuple of a relation must be uniquely identified by its contents.
• Sequence of tuples in a relation is immaterial. The tuples are not considered to be ordered, even though they appear to be in tabular form.

Property 3: imposes following rules on the state of a relation.
• All data values in an attribute must be from the same domain (same data type).
• Each data value associated with an attribute must be atomic (cannot be further divisible into meaningful subparts)
• No attribute can have many data values in one tuple
• A special value "NULL" is used to represent values that are unknown or non-applicable to certain attributes. (Data unknown). a null value is not the same as a value of zero

Tuple/record:-row in table
attribute /field:-column in table

degree: - no of attribute/column
cardinality: - no of rows
Domain:- a pool/set of values from which a particular value is selected for that column



view:- virtual table that not exist in its own derived from existing table
Example: in student table monthly fee is stored and annual fee can be calculated from there that is not present in table

---

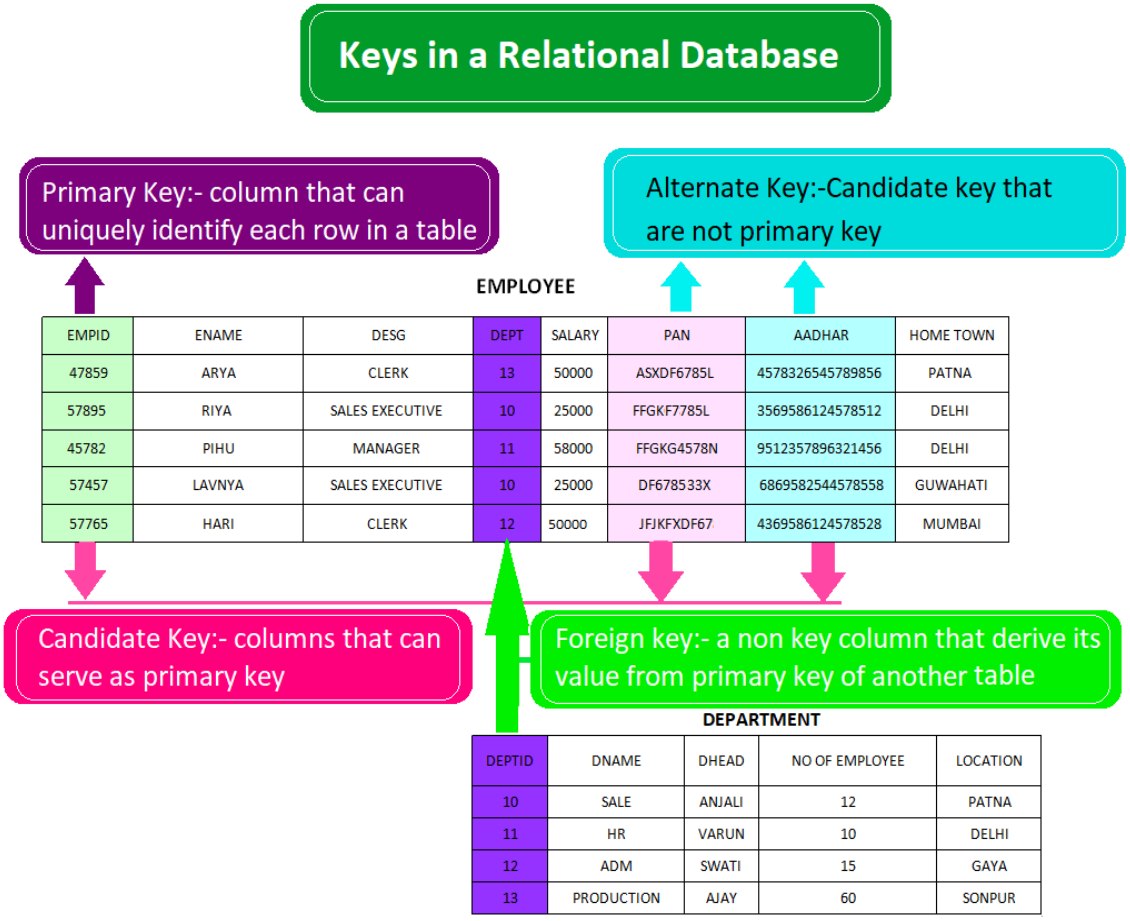
KEYS IN A RELATIONAL DATABASE

Primary Key:- a column/attribute or set of columns/attribute that can uniquely identify each and every tuple/row in a relation/table is called primary key. a table can have only one primary key. primary key must be unique and cannot have null values

Candidate Key:- all columns/attribute or set of columns/attribute that can serve as primary means uniquely identify each and every tuple/row in a relation/table are called candidate key.

Alternate Key:- all candidate key that are not primary key are alternate key

Foreign key:- a non key column that derive its value from primary key of other table



STRUCTURED QUERY LANGUAGE (SQL)

SQL:- query language used to access and manipulate data from the database
:- not case sensitive
:-SQL isn't a procedural language, as are FORTRAN, BASIC, C, COBOL, Pascal, and Java
Simply tell SQL what you want (as if you were talking to Aladdin's genie) instead of telling the system how to get you what you want.

Three languages-within-a-language offer everything
you need to create, modify, maintain, and provide security for a relational database:

SQL for Data Definition:-

The Data Definition Language (DDL): The part of SQL that creates a database/table, modifies its structure, and destroys it when you no longer need it.

SQL for Data Manipulation: -

The Data Manipulation Language (DML): Data Manipulation using a database means either insertion of new data, removal of existing data or modification of existing data in the database

DDL	DML
Data definition language	Data Manipulation language
Work on structure	Work on data
Create table, create database, alter table, drop table,	update , insert, delete

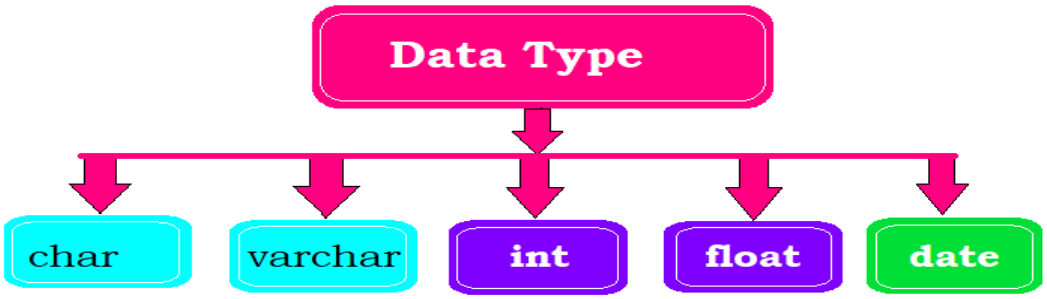
SQL for Data Query: -

SQL provides efficient mechanisms to retrieve data stored in multiple tables in MySQL database (or any other RDBMS).

The SQL statement SELECT is used to retrieve data from the tables in a database and is also called a query statement.

Data type: - Indicates the type of data value and operation that can be performed on that data

Different Data type:- data type (char(n), varchar(n), int, float, date)



Char(n):- Specifies character type data of length n where n could be any value from 0 to 255. if you don't specify a value for n, sql assumes a field length of one character that means column size is char(1). CHAR is of fixed length, which means, declaring CHAR (10) implies to reserve spaces for 10 characters. 'city' has four characters, MySQL fills the remaining 6 characters with spaces padded on the right.

Varchar(n):- Specifies character type data of length n where n could be any value from 0 TO 65535. if you don't specify a value for n, it will produce an error . VARCHAR(n) is a variable-length data type. That is, declaring VARCHAR (30) means a maximum of 30 characters can be stored but the actual allocated bytes will depend on the length of the

entered string. So 'city' in VARCHAR (30) will occupy space needed to store 4 characters only.

	CHAR(N)	VARCHAR(N)
	SPECIFIES CHARACTER TYPE DATA OF LENGTH N	
WHERE N COULD BE	ANY VALUE FROM 0 TO 255	ANY VALUE FROM 0 TO 65535
IF DON'T SPECIFY A VALUE FOR N	SQL ASSUMES A FIELD LENGTH OF ONE CHARACTER THAT MEANS COLUMN SIZE IS CHAR (1)	IT WILL PRODUCE ERROR
ACTUAL ALLOCATED BYTES	CHAR IS OF FIXED LENGTH	VARCHAR(N) IS A VARIABLE-LENGTH DATA TYPE
	CHAR (10) IMPLIES TO RESERVE SPACES FOR 10 CHARACTERS	VARCHAR (30) MEANS A MAXIMUM OF 30 CHARACTERS CAN BE STORED ACTUAL SPACES WILL DEPEND ON THE LENGTH OF ENTERED STRING
ENTER FEWER CHARACTERS THAN THE SPECIFIED NUMBER	SQL FILLS THE REMAINING CHARACTER SPACES WITH BLANKS	STORE EXACTLY THE NUMBER OF CHARACTERS THAT THE USER ENTERS

Int :- specifies an integer value. Occupies 4 bytes of storage. Unsigned values allowed are 0 to 4,294,967,295. For values larger than that, we have to use BIGINT, which occupies 8 bytes.

Float:- Holds numbers with decimal points. Occupies 4 bytes. For float(M,D) M must be >= D otherwise it will produce error.
For example float(4,2) , can hold a float number of 4 digit ,out of which 2 digits out of 4 digit are decimal points.

Date:- The DATE type is used for dates in 'YYYY-MM-DD' format. YYYY is the 4 digit year, MM is the 2 digit month and DD is the 2 digit date. The supported range is '1000-01-01' to '9999-12-31'. Take input in "" or ''. produce error if you give value without "" or ''

Constraints:- are the certain types of restrictions on the data values that an attribute can have

Type of constraints :-

Constraint	Description
NOT NULL	Ensures that a column cannot have NULL values where NULL means missing/ unknown/not applicable value.
UNIQUE	Ensures that all the values in a column are distinct/unique
DEFAULT	A default value specified for the column if no value is provided
PRIMARY KEY	The column which can uniquely identify each row/record in a table
FOREIGN KEY	The column which refers to value of an attribute defined as primary key in another table

Constraint	**Create table command**
PRIMARY KEY	CREATE TABLE staff (sno integer **primary key**, name varchar(20), salary integer **not null**, mobileno float(10) **unique**, bonus integer **default** 5000, dno integer **REFERENCES** dept(deptid));
NOT NULL	
UNIQUE	
DEFAULT	
FOREIGN KEY	

Constraints can be added in two ways: as a column constraint or as a table constraint.

SQL command

1) **SHOW DATABASES** :- show list of existing database present in mysql server

```
mysql> show databases;
+----------------------------+
| Database                   |
+----------------------------+
| information_schema         |
| data                       |
| mysql                      |
| performance_schema         |
+----------------------------+
4 rows in set (0.09 sec)
```

2) **CREATE DATABASE** :- it is **DDL** command to create a database
syntax: CREATE DATABASE databasename;
mysql> create database xiicip;

Query OK, 1 row affected (0.03 sec)

3) to open database /use database / make database current database
syntax: USE databasename;
mysql> use xiicip;
Database changed

4) **DROP DATABASE**: - it is a **DDL** command to remove a specified database from the server.
syntax: DROP DATABASE databasename;
mysql> drop database xiics;
Query OK, 0 rows affected (0.23 sec)

5) **CREATE TABLE** :-it is a **DDL** command to create a table

syntax: CREATE TABLE tablename(attributename1 datatype constraint, attributename2 datatype constraint, : attributenameN datatype constraint);

Consider description of following table, write command to create these table

department

field name	type	size	constrain
deptid	integer		PRIMARY KEY
dname	character	10	
dhead	character	50	

Employee

field name	type	Size	constrain
Eid	integer		primary key
Ename	character	50	
gender	character	1	
deptid	integer		Foreign key
salary	integer		
doj	date		

```
mysql> create table dept (deptid integer primary key, dname
varchar(10), dhead varchar(50));
Query OK, 0 rows affected (0.13 sec)

mysql> create table employee(eid integer primary key, ename
varchar(50), gender char(1), deptno integer, salary integer, doj date,
FOREIGN KEY (deptno) REFERENCES dept(deptid));
```

or

```
mysql> create table employee(eid integer primary key, ename
varchar(50), gender char(1), deptno integer REFERENCES
dept(deptid), salary integer, doj date);

Query OK, 0 rows affected (0.07 sec)
```

Note :- create parent table before creating child table

6) **SHOW TABLES** :- display the list of table present in current database

```
mysql> show tables;
+-----------------+
| Tables_in_xiicip |
+-----------------+
| dept            |
| employee        |
+-----------------+
2 rows in set (0.00 sec)
```

7) **DESCRIBE/ DESC** :-display structure of a table/ column's name , type, constraints of table

syntax: DESC tablename;

```
mysql> desc dept;
 or
mysql> describe dept;

+---------+---------------+-------+-------+------------+-------+
| Field   | Type          | Null  | Key   | Default    | Extra |
+---------+---------------+-------+-------+------------+-------+
| deptid  | int(11)       | NO    | PRI   | NULL       |       |
| dname   | varchar(10)   | YES   |       | NULL       |       |
| dhead   | varchar(50)   | YES   |       | NULL       |       |
+---------+---------------+-------+-------+------------+-------+
3 rows in set (0.01 sec)
```

8) **ALTER TABLE**: - it is a **DDL** command to work on attributes and constraints. change or alter the structure (schema) of the table

Example table for alter command

```
mysql> desc staff;
+---------+------------------+------+-----+---------+---------+
| Field   | Type             | Null | Key | Default | Extra   |
+---------+------------------+------+-----+---------+---------+
| staffno | int(11)          | YES  |     | NULL    |         |
| sname   | varchar(20)      | YES  |     | NULL    |         |
| dno     | int(11)          | YES  |     | NULL    |         |
+---------+------------------+------+-----+---------+---------+
```

i) **ADD clause**:- can add a new attribute/column , constrain(primary key , foreign key, unique)

a) add new attribute in table

syntax: ALTER TABLE tablename ADD attributename1 datatype constraint;

This command is used to add new column with its datatype, size and constraint

```
mysql> alter table staff add address varchar(50) not null;

mysql> desc staff;
+---------+-----------------+------+-------+----------+-------+
| Field   | Type            | Null | Key   | Default  | Extra |
+---------+-----------------+------+-------+----------+-------+
| staffno | int(11)         | YES  |       | NULL     |       |
| sname   | varchar(20)     | YES  |       | NULL     |       |
| dno     | int(11)         | YES  |       | NULL     |       |
| address | varchar(50)     | NO   |       | NULL     |       |
+---------+-----------------+------+-------+----------+-------+
```

b) **add primary key constraint**

syntax: ALTER TABLE tablename ADD PRIMARY KEY (attribute_name);

This command is used to make the existing column primary key in the table. If the column is not in the table, it will produce an error.

```
mysql> alter table staff add primary key(staffno);
mysql> desc staff;
+---------+-----------------+------+-------+----------+-------+
| Field   | Type            | Null | Key   | Default  | Extra |
+---------+-----------------+------+-------+----------+-------+
| staffno | int(11)         | YES  | PRI   | NULL     |       |
| sname   | varchar(20)     | YES  |       | NULL     |       |
```

```
| dno     | int(11)     | YES |     | NULL |     |
| address | varchar(50) | NO  |     | NULL |     |
+---------+-------------+-----+-----+------+-----+
```

c) **add foreign key constraint**

syntax: ALTER TABLE tablename ADD FOREIGN KEY(attribute name) REFERENCES referenced_table_name (attribute name);

This command is used to make an existing column foreign key that derives its value from the primary key of another table.

```
mysql> alter table staff add foreign key(dno) references dept (deptid);
```

```
mysql> desc staff;
+---------+-------------+-----+-----+---------+-------+
| Field   | Type        | Null| Key | Default | Extra |
+---------+-------------+-----+-----+---------+-------+
| staffno | int(11)     | YES | PRI | NULL    |       |
| sname   | varchar(20) | YES |     | NULL    |       |
| dno     | int(11)     | YES | MUL | NULL    |       |
| address | varchar(50) | NO  |     | NULL    |       |
+---------+-------------+-----+-----+---------+-------+
```

d) **add unique constraint**

syntax: ALTER TABLE tablename ADD UNIQUE (attribute name) ;

This command is used to apply a unique constraint on the existing column of the table.

```
mysql> alter table staff add unique (address) ;
mysql> desc staff;
+---------+-------------+-----+-----+---------+-------+
| Field   | Type        | Null| Key | Default | Extra |
+---------+-------------+-----+-----+---------+-------+
| staffno | int(11)     | YES | PRI | NULL    |       |
| sname   | varchar(20) | YES |     | NULL    |       |
| dno     | int(11)     | YES | MUL | NULL    |       |
| address | varchar(50) | NO  | UNI | NULL    |       |
+---------+-------------+-----+-----+---------+-------+
```

ii) **MODIFY clause**: - can change existing attribute datatype, size and constraint

syntax: ALTER TABLE tablename Modify attributename1 newdatatype (new_size) new_constraint;

This command is used to again define existing column datatype, size and constraint

```
mysql> alter table staff modify sname char(40) not null;
+---------+----------------+------+-------+----------+-------+
| Field   | Type           | Null | Key   | Default | Extra |
+---------+----------------+------+-------+----------+-------+
| staffno | int(11)        | YES  | PRI   | NULL    |       |
| sname   | char(40)       | NO   |       | NULL    |       |
| dno     | int(11)        | YES  | MUL   | NULL    |       |
| address | varchar(50)    | NO   | UNI   | NULL    |       |
+---------+----------------+------+-------+----------+-------+
```

```
mysql> alter table staff modify dno integer default 101;

+---------+----------------+------+-------+----------+-------+
| Field   | Type           | Null | Key   | Default | Extra |
+---------+----------------+------+-------+----------+-------+
| staffno | int(11)        | YES  | PRI   | NULL    |       |
| sname   | char(40)       | NO   |       | NULL    |       |
| dno     | int(11)        | YES  | MUL   | 101     |       |
| address | varchar(50)    | NO   | UNI   | NULL    |       |
+---------+----------------+------+-------+----------+-------+
```

iii) **Drop clause :-** used to remove column and primary key

a) Remove an attribute

syntax: ALTER TABLE tablename DROP attributename;

This command is used to remove specific attributes from the table.

```
mysql> alter table staff drop address;
mysql> desc dept;
+---------+----------------+------+-------+----------+-------+
| Field   | Type           | Null | Key   | Default | Extra |
+---------+----------------+------+-------+----------+-------+
| staffno | int(11)        | YES  | PRI   | NULL    |       |
| sname   | char(40)       | NO   |       | NULL    |       |
| dno     | int(11)        | YES  | MUL   | 101     |       |
+---------+----------------+------+-------+----------+-------+
```

b) Remove primary key

syntax: ALTER TABLE tablename DROP PRIMARY KEY;

This command is used to remove the primary key constraint from the table. It will remove constraints only, not columns.

```
mysql> alter table staff drop primary key;
mysql> desc dept;
+---------+----------------+------+-------+----------+-------+
| Field     | Type           | Null | Key   | Default | Extra |
+---------+----------------+------+-------+----------+-------+
| staffno | int(11)        | YES |       | NULL   |       |
| sname   | char(40)       | NO   |       | NULL   |       |
| dno       | int(11)        | YES  | MUL | 101      |       |
+---------+----------------+------+------+----------+-------+
```

iv) Change clause :- used to rename column and change column defination

syntax: ALTER TABLE tablename Change attributename new_name new_datatype (new_size) new_constraint;

```
mysql> alter table staff change sname Name varchar(50);
+---------+----------------+------+-------+----------+-------+
| Field     | Type           | Null | Key   | Default | Extra |
+---------+----------------+------+-------+----------+-------+
| staffno | int(11)        | YES |       | NULL   |       |
| Name    | Varchar(50)  | NO   |       | NULL   |       |
| dno       | int(11)        | YES  | MUL | 101      |       |
+---------+----------------+------+------+----------+-------+
```

9) **Insert into:-** it is a **DML** command to insert data into table(add new row in table)

Syntax: INSERT INTO tablename (attributename1, attributename2……) VALUES (VALUE1, VALUE2…….) ;

```
mysql> insert into dept(dname, deptid) values("HR",101);
mysql > select * from  dept;
+---------+---------+---------+
| deptid | dname | dhead |
+---------+---------+----------+
|   101   | HR      | NULL |
+---------+---------+---------+
```

```
mysql> insert into employee values(48339, "ana", "f",101,
56000,"2020-03-03");
Query OK, 1 row affected (0.02 sec)
```

10) **SELECT:-** to display data of table

Syntax:

```
SELECT columnname1,columnname2
FROM tablename
WHERE condition
ORDER BY columnname
```

Note:- all column :-*

Consider employee table for select command

a)to display details of all employee(all columns and all rows)

```
mysql> select * from employee;
```

eid	name	gender	deptid	date_of_Join	city	salary
4563	ravi	m	101	2002-12-11	patna	53000
41350	paras	m	103	2018-07-01	delhi	52000
42369	arabh	m	101	2020-01-03	patna	53000
42374	alisha	f	102	2020-01-05	pune	50000
47865	saurav	m	104	2022-01-29	sonpur	45000
47869	aisika	f	103	2021-04-20	delhi	55000
48356	praveen	m	107	2022-01-27	patna	45000
65897	aman	m	107	2021-10-23	patna	56000
72346	harsh	m	104	2021-07-25	hajipur	NULL
78346	gitanjali	f	104	2021-12-23	hajipur	57000
78456	gita	f	102	2021-10-23	agra	65000

b) to display details only few columns in specified order

```
mysql> select city, name, deptid from employee;
```

city	name	deptid
patna	ravi	101
delhi	paras	103
patna	arabh	101
pune	alisha	102
sonpur	saurav	104
delhi	aisika	103
patna	praveen	107
patna	aman	107
hajipur	harsh	104
hajipur	gitanjali	104
agra	gita	102

NOTE:- You can display any column in any order . You can give a new name (alias name) for display. You can apply a mathematical operation on an integer column.

```
mysql> select eid, name , date_of_Join as doj, salary*12 as
annual_salary from employee;
+--------+-------------+---------------+-------------------+
| eid    | name        | doj           | annual_salary     |
+--------+-------------+---------------+-------------------+
|  4563 | ravi        | 2002-12-11 |      636000      |
| 41350 | paras       | 2018-07-01 |      624000      |
| 42369 | arabh       | 2020-01-03 |      636000      |
| 42374 | alisha      | 2020-01-05 |      600000      |
| 47865 | saurav      | 2022-01-29 |      540000      |
| 47869 | aisika      | 2021-04-20 |      660000      |
| 48356 | praveen     | 2022-01-27 |      540000      |
| 65897 | aman        | 2021-10-23 |      672000      |
| 72346 | harsh       | 2021-07-25 |       NULL       |
| 78346 | gitanjali   | 2021-12-23 |      684000      |
| 78456 | gita        | 2021-10-23 |      780000      |
+---------+-------------+---------------+-------------------+
```

c) Where clause: - It is used to apply conditions on rows. If a row fulfils that condition then only it will be selected.

i) Relational operator (=,<>,>=,<=,<,>)

where column_name **operator** "value"

example :-

a) to display details of female employee

```
mysql> select * from employee where gender="f";
```

```
+---------+------------+----------+---------+----------------+-----------+----------+
| eid     | name       | gender | deptid | date_of_Join  | city      | salary   |
+---------+------------+---------+---------+----------------+-----------+----------+
| 42374 | alisha     | f       |   102  | 2020-01-05  | pune     |  50000 |
| 47869 | aisika     | f       |   103  | 2021-04-20  | delhi    |  55000 |
| 78346 | gitanjali  | f       |   104  | 2021-12-23  | hajipur  |  57000 |
| 78456 | gita       | f       |   102  | 2021-10-23  | agra     |  65000 |
+---------+------------+---------+---------+----------------+-----------+----------+
```

b) to display employee id, name and salary of employee earning more than 55000

```
mysql> select eid , name, salary from employee where salary>55000;
+--------+-----------+----------+
| eid    | name      | salary   |
+--------+-----------+----------+
| 65897 | aman      |  56000 |
| 78346 | gitanjali |  57000 |
| 78456 | gita      |  65000 |
+--------+-----------+----------+
```

c) to display employee name department id and city of employee not living in Patna

```
mysql> select name, deptid, city from employee where city<>"patna";
+-----------+----------+----------+
| name      | deptid   | city     |
+-----------+----------+----------+
| paras     |   103    | delhi    |
| alisha    |   102    | pune     |
| saurav    |   104    | sonpur   |
| aisika    |   103    | delhi    |
| harsh     |   104    | hajipur  |
| gitanjali |   104    | hajipur  |
| gita      |   102    | agra     |
+-----------+----------+----------+
```

ii) Logical operator (and , or, not)

a) where condition1 **and** condition2 :-true is both condition are true

b) where condition1 or condition2:- true at least one condition is true

c) **where not**(condition)

c) to display employee id, name and salary of employee earning from 50000 to 60000 both value included

```
mysql> select eid , ename, salary from employee where salary
between 50000 and 60000;
+---------+---------+-----------+
| eid     | ename   | salary    |
+---------+---------+-----------+
| 48339   | ana     |  56000    |
+---------+---------+-----------+
1 row in set (0.04 sec)
```

d) to display employee id, name and deptid male employee of 103 department

```
mysql> select eid , ename, deptid from employee where deptid=103
and gender="m";
+-------+-------+--------+
| eid   | ename | deptid |
+-------+-------+--------+
| 41350 | paras |   103  |
| 42350 | rana  |   103  |
+-------+-------+--------+
2 rows in set (0.01 sec)
```

e) to display employee id, name and deptid of employees of 101 or 102 department

```
mysql> select eid , ename, deptid from employee where deptid=102
or deptid=101;
+--------+---------+--------+
| eid    | ename  | deptid |
+--------+---------+--------+
| 42369 | arabh  |   101 |
| 42374 | alisha  |   102 |
| 48339 | ana    |   101 |
+---------+---------+--------+
3 rows in set (0.00 sec)
```

f) to display employee id, name of employee present in a list of employee

```
mysql> select eid , ename from employee where ename
in("ana","mina","ajay");
+-------+-------+
| eid   | ename |
+-------+-------+
| 41250 | mina  |
| 48339 | ana   |
+-------+-------+
2 rows in set (0.00 sec)
```

g) to display employee id, name of employee whose name ending at “a”

```
mysql> select eid , ename from employee where ename like "%a";
+-------+--------+
| eid   | ename  |
+-------+--------+
| 41250 | mina   |
| 42350 | rana   |
| 42374 | alisha |
| 48339 | ana    |
+-------+--------+
4 rows in set (0.00 sec)
```

h) to display employee id, name and date of join of employee whose join in 2018

```
mysql> select eid , ename from employee where doj like "2018%";
+--------+---------+----------------+
| eid    | ename | doj            |
+--------+---------+----------------+
| 41250 | mina  | 2018-09-01 |
| 41350 | paras | 2018-07-01 |
| 42350 | rana  | 2018-01-03 |
```

```
+---------+--------+----------------+
3 rows in set, 1 warning (0.00 sec)
```

ix) to update a particular data in table

Syntax: UPDATE tablename SET COLUNMANE=VALUE

```
mysql> update employee set salary=salary+500;
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

x) to delete data from table

```
mysql> delete from employee where ename="ana";
Query OK, 1 row affected (0.02 sec)
```

xi) to display data in a particular order

```
mysql> select eid, ename, deptid from employee order by ename;
+--------+----------+---------+
| eid      | ename  | deptid |
+--------+----------+---------+
| 42374 | alisha   |    102 |
| 48339 | ana      |    101 |
| 42369 | arabh   |    101 |
| 41250 | mina    |    103 |
| 41350 | paras   |    103 |
| 42350 | rana   |   103 |
+-------+--------+--------+
6 rows in set (0.00 sec)
```

xii) to apply aggregate function

```
 mysql> select max(salary) from employee;
+-------------+
| max(salary) |
+-------------+
|      65500 |
+-------------+
1 row in set (0.00 sec)

mysql> select min(salary) from employee where gender="f";
+-------------+
| min(salary) |
+-------------+
|      25500 |
+-------------+
```

```
1 row in set (0.00 sec)

mysql> select sum(salary) from employee where deptid=103;
+-------------+
| sum(salary) |
+-------------+
|      81500 |
+-------------+
1 row in set (0.00 sec)

mysql> select avg(salary) from employee where gender="m";
+-------------+
| avg(salary) |
+-------------+
|  40500.0000 |
+-------------+
1 row in set (0.00 sec)

mysql> select count(*) from employee where gender="m";
+----------+
| count(*) |
+----------+
|       3 |
+----------+
1 row in set (0.00 sec)
```

xiii) to use group by clause

a) to display count of employee of each gender

```
mysql> select gender, count(*) from employee group by gender;
+----------+-----------+
| gender  | count(*) |
+----------+-----------+
| f           |       3      |
| m          |       3      |
+----------+------------+
2 rows in set (0.07 sec)
```

b) to display deptid and sum of salary of those department where number of employee is more than two

```
mysql> select deptid, sum(salary) from employee group by deptid
having count(*)> 2;
+--------+-----------------+
| deptid | sum(salary) |
```

```
+--------+----------------+
|   103 |     81500      |
+--------+----------------+
1 row in set (0.01 sec)
```

==========

==========

JOINS : CARTESIAN PRODUCT ON TWO TABLES, EQUI-JOIN AND NATURAL JOIN

Cartesian product (X)/cross joint:- is denoted by X symbol. Let's say we have two tables R1 and R2 then the Cartesian product of these two tables (R1 X R2) would combine each row of first table R1 with the each row of second table R2.

Cardinality of resultant table=cardinality of R1 * cardinality of R2

Degree of resultant table =degree of R1 +degree of R2

R1:- dance

```
+-----+-------------+
| tid | name        |
+-----+-------------+
| 100 | kavita      |
| 101 | riya        |
| 102 | arabh       |
+-----+-------------+
```

R2:- student

```
+-----+------+------+
| sid | name | id   |
+-----+------+------+
|   1 | arya |  100 |
|   2 | anu  |  102 |
+-----+------+------+
```

This query will display Cartesian product of two table

mysql> select * from dance , student;

```
+-----+--------+-----+------+------+
| tid | name   | sid | name | id   |
+-----+--------+-----+------+------+
| 100 | kavita |   1 | arya |  100 |
| 100 | kavita |   2 | anu  |  102 |
| 101 | riya   |   1 | arya |  100 |
| 101 | riya   |   2 | anu  |  102 |
| 102 | arabh  |   1 | arya |  100 |
```

| 102 | arabh | 2 | anu | 102 |
+-----+--------+-----+------+------+

JOIN

Join – JOIN operation combines tuples from two tables on specified conditions.

4. EQUI JOIN:

- ✓ The Join, in which columns are compared for equality, is called Equi-join.
- ✓ In equi-join, all the columns from joining table appear in the output even if they are identical.
- ✓ Using other comparison operators (such as <) disqualifies a join as an equi-join.

Example of an explicit equi join:

```
SELECT * FROM emp JOIN dept  ON emp.Dno = dept.Dno;
```

Example of an implicit equi join:

```
SELECT * FROM emp, dept WHERE emp.Dno = dept.Dno;
```

Output: The above query will produce the following output:

Eno	Dno	Ename	Job	Sal	Dno	Dname	Loc
196	10	ANKIT	SALSEMAN	1250	10	ACCOUNTING	RAJKOT
134	20	SIYA	CLERK	1100	20	RESEARCH	AHEMDABAD
182	30	RADHIKA	ANALYST	3200	30	SALSE	BARODA

6. NATURAL JOIN:

- ✓ A natural join is a type of equi-join where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables.
- ✓ The resulting joined table contains only one column for each pair of equally named columns.

Example of an explicit natural join:

```
SELECT * FROM emp NATURAL JOIN dept;
```

With NATURAL JOIN clause, you need not specify Join Condition

Example of an implicit natural join:

```
SELECT * FROM emp.*, dname, loc
  FROM emp, dept
  WHERE emp.Dno = dept.Dno;
```

See this time no duplicate column

Output: The above query will produce the following output:

Eno	Ename	Job	Sal	Dno	Dname	Loc
196	ANKIT	SALSEMAN	1250	10	ACCOUNTING	RAJKOT
134	SIYA	CLERK	1100	20	RESEARCH	AHEMDABAD
182	RADHIKA	ANALYST	3200	30	SALSE	BARODA