1. Using any open source data modelling tool (MySQL/Rational Rose/ERwin), design and develop a ER diagram for the following Library database. Specify relevant Referential constraints.

BOOK_AUTHORS

BOOK_ID	AUTHOR_NAME

PUBLISHER

NAME	ADDRESS	PHONE

BOOK_COPIES

BOOK_ID	BRANCH_ID	NO_COPIES

BOOK_LOANS

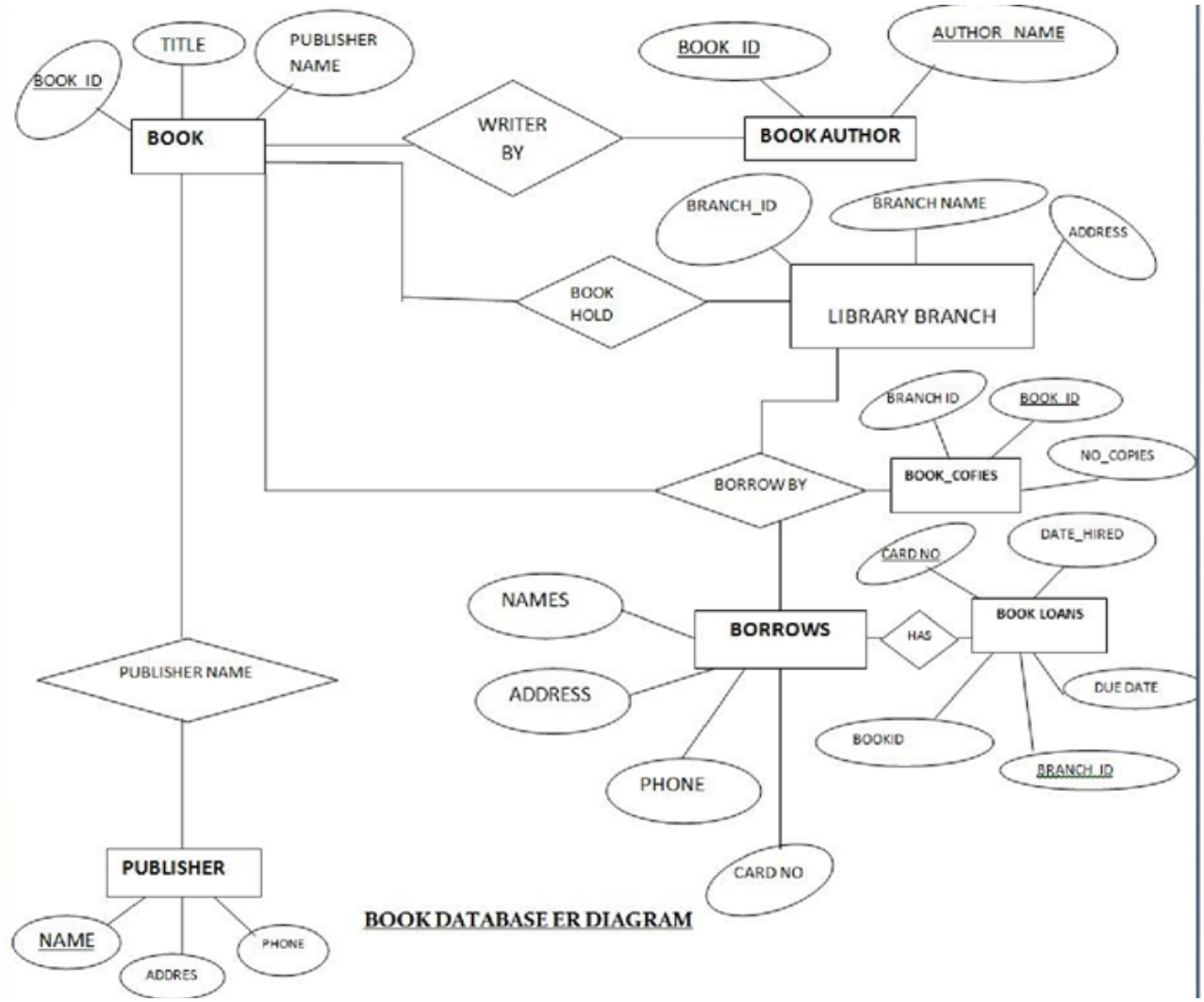
BOOK_ID	BRANCH_ID	CARD_NO	DATE_HIRED	DUE_DATE

LIBRARY_BRANCH

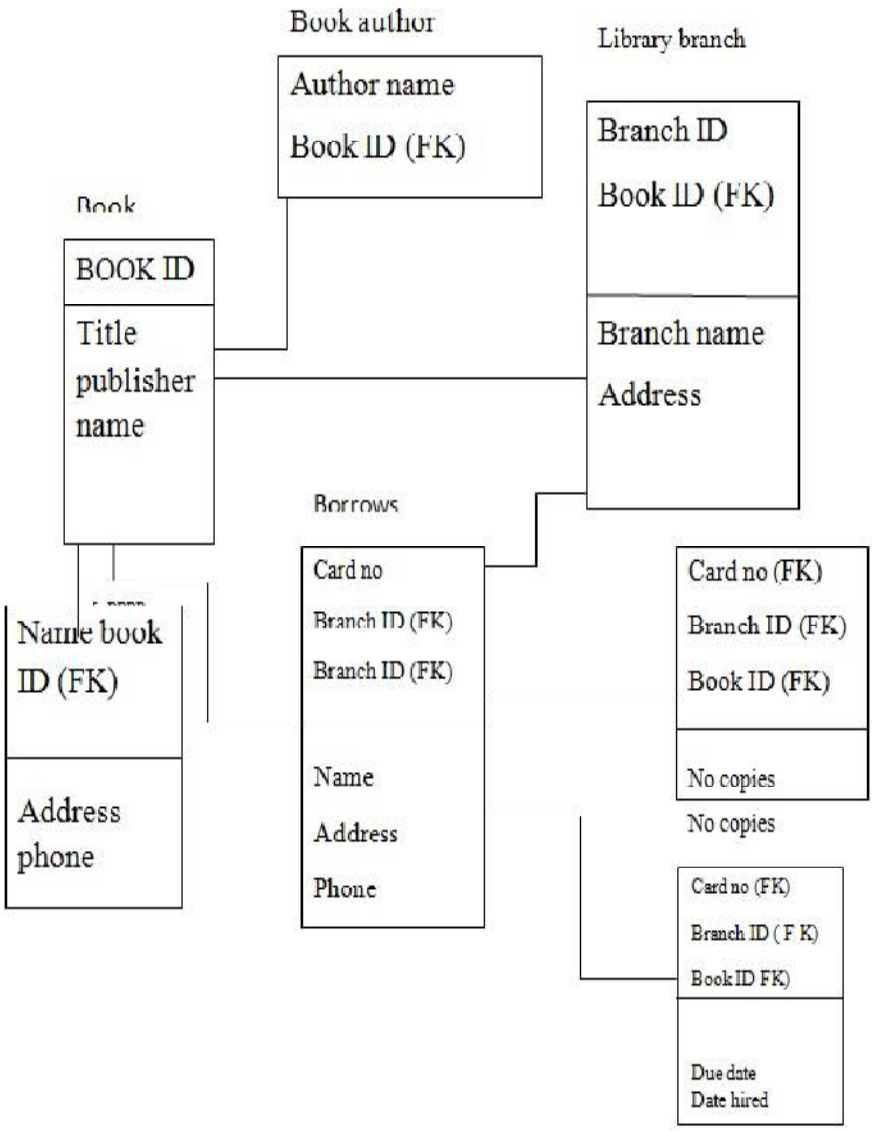
BRANCH_ID	BRANCH_NAME	ADDRESS

BORROWER

CARD_NO	NAME	ADDRESS	PHONE

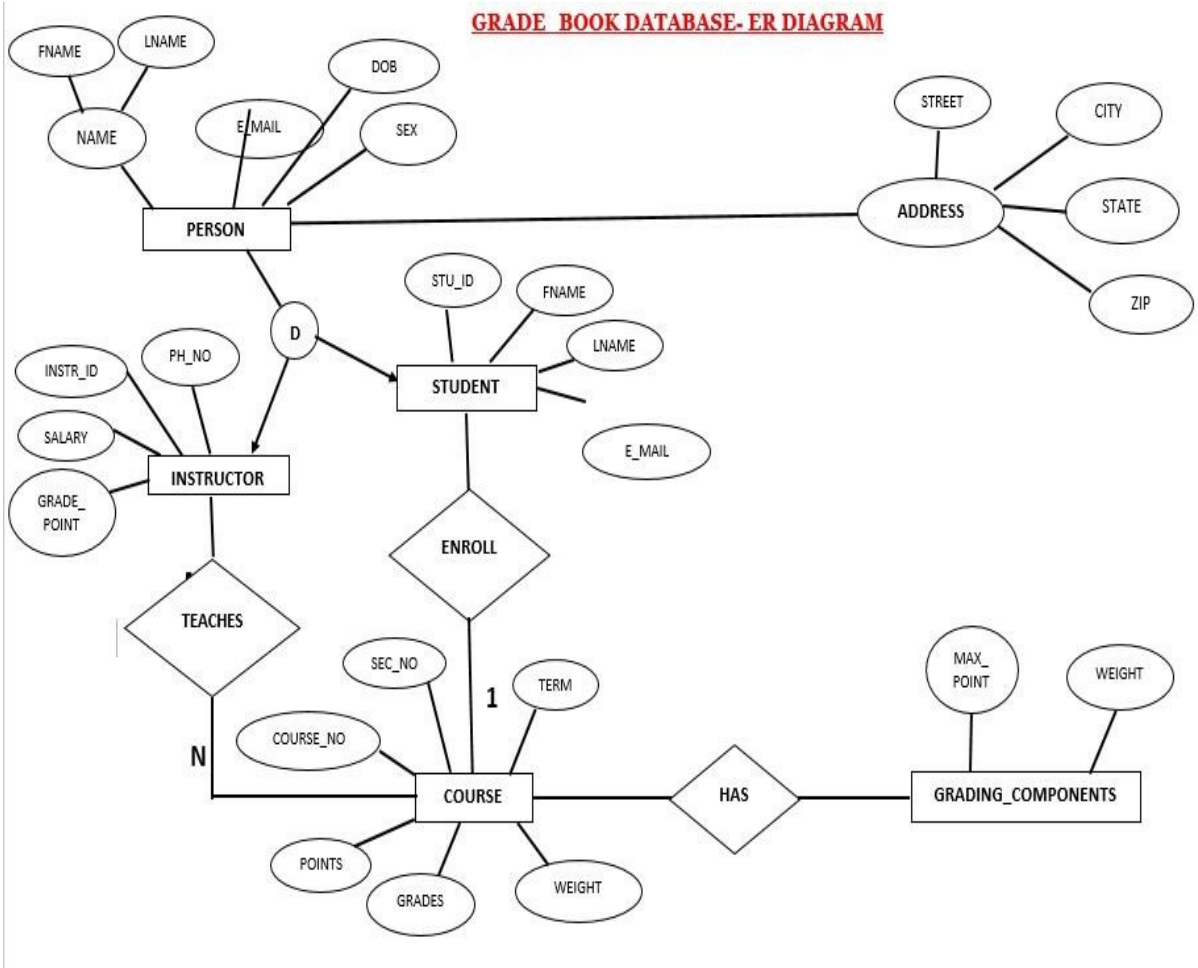


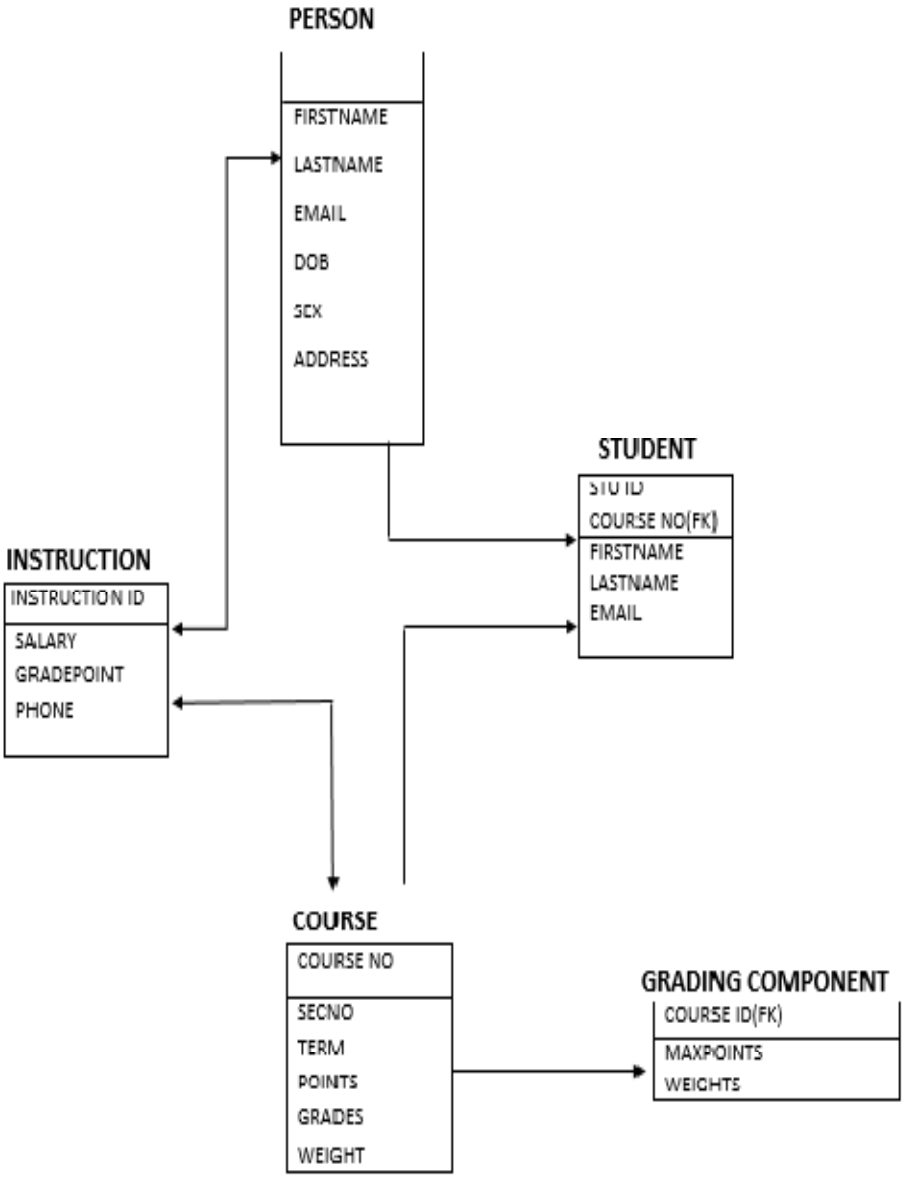
BOOK DATABASE ER DIAGRAM


Book author
Author name
Book ID (FK)
Library branch
Branch ID
Book ID (FK)
Book
BOOK ID
Title
publisher
name
Branch name
Address
Borrows
Card no
Branch ID (FK)
Branch ID (FK)
Name
Address
Phone
Card no (FK)
Branch ID (FK)
Book ID (FK)
No copies
Name book
ID (FK)
Address
phone
No copies
Card no (FK)
Branch ID (F K)
Book ID FK)
Due date
Date hired

2. Consider a GRADE_BOOK database in which instructors within an academic department record point earned by individual students in theory classes. The data requirements are summarised as follows:

- Each student is identified by a unique identifier, first and last name and an email address.
- Each instructor teaches certain courses each term. Each course is identified by a course number, a section number and the term in it is taught. For each course he or she teaches, the instructor specifies the minimum number of points required in order to earn letter grades, A, B, C, D and F. For Ex. 90 points for an A, 80 points for a B, 70 points for a C and so forth.
- Students are enrolled in each course taught by the instructor.
- Each course has a number of grading components (such as mid-term exam, final exam, project and so forth). Each grading component has a maximum number of points (100 or 50) and a weight (20% or 10%). The weight of all the grading components of a course is 100.
- Finally, the instructor records the points earned by each student in each of the grading components in each course. For Ex. The student 1234 earns 84 points for a mid-term exam, grading component of the section 2 course CS 2310 in the fall term of 2009. The mid-term exam grading component may have been defined to have a maximum of 100 points and a weight of 20% of the course grade.




PERSON
FIRSTNAME
LASTNAME
EMAIL
DOB
SEX
ADDRESS
STUDENT
STU ID
COURSE NO(FK)
FIRSTNAME
LASTNAME
EMAIL
INSTRUCTION
INSTRUCTION ID
SALARY
GRADEPOINT
PHONE
COURSE
COURSE NO
SECNO
TERM
POINTS
GRADES
WEIGHT
GRADING COMPONENT
COURSE ID(FK)
MAXPOINTS
WEIGHTS

3. CREATE THE FOLLOWING TABLES FOR A COMPANY DATABASE

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPER_SSN	DNO

create table employee (fname varchar(10), minit varchar(10), lname varchar(10), ssn varchar(10), bdate date, address varchar(30), sex char(10), salary number(10), super_ssn char(10), dno number(10));

desc employee;

DEPARTMENT

DNAME	DNUMBER	MGR_SSN	MGR_START_DATE

create table department (dname varchar(15), dnumber number(10) , mgr_ssn varchar(10), mgr_start_date date);

desc department;

DEPT_LOCATIONS

DNUMBER	DLOCATION

create table dept_locations(dnumber number(10), dlocation varchar(20));

desc dept_locations;

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

create table project (pname varchar(20), pnumber number(10), plocation varchar(20), dnum number(10));

desc project;

WORKS_ON

ESSN	PNO	HOURS

create table works_on (essn varchar(10), pno number(10), hours number(10));

desc works_on;

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

create table dependent (essn char(10), dependent_name varchar(20), sex char(10), bdate date, relationship varchar(10));

desc dependent;

4. ILLUSTRATE THE USE OF CONSTRAINTS

NOT NULL
PRIMARY
KEY UNIQUE
CHECK
DEFAULT
REFERENCES

NOT NULL CONSTRAINT:

The NOT NULL constraint enforces a column to NOT accept NULL values.
The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

EMPLOYEE

FNAME	SSN	ADDRESS	SALARY	DNO

create table employee (fname varchar(10) **not null**, ssn number(10), address varchar(30), salary number(10), dno number(10));

desc employee;

Inserting values into the table

insert into employee values('raju',10,'koppal',5000,5);
insert into employee values(15,'hospet',7000,4);

PRIMARY KEY CONSTRAINT:

The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values. A primary key column cannot contain NULL values.
Most tables should have a primary key, and each table can have only ONE primary key.

create table employee(fname varchar(10), ssn number(10), address varchar(30), salary number(10), dno number(10) **primary key**);

desc employee;

Inserting values into the table

insert into employee values('raju',10,'koppal',5000,5);
insert into employee values('ravi',15,'hospet',7000,5);

UNIQUE CONSTRAINT:

The UNIQUE constraint uniquely identifies each record in a database table.
The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.
A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.
Note that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

create table employee (fname varchar(10), ssn number(10) **unique**, address varchar(30), salary number(10), dno number(10));

desc employee;

Inserting values into the table

insert into employee values('raju',123,'koppal',5000,5);
insert into employee values('ravi',123,'hospet',7000,6);

REFERENCES CONSTRAINT:

1. **DEPARTMENT**

DNAME	DNO

create table department(dname varchar(20),dno number(10) **primary key**);

desc department;

Inserting values into the table

insert into department values('testing',1);

insert into department values('debugging',2);

select * from department;

2. **DEPT_LOCATION**

DNUMBER	DLOCATION

create table dept_location(dnum number(10) **references** department(dno),dlocation varchar(20));

desc dept_location;

Inserting values into the table

insert into dept_location values(1,'koppal');

insert into dept_location values(3,'hospet');

CHECK CONSTRINT:

The CHECK constraint is used to limit the value range that can be placed in a column.
If you define a CHECK constraint on a single column it allows only certain values for this column.

create table student (regno varchar(10), name varchar(20), age int **check** (age>=18), city varchar(20));

desc student;

Inserting values into the table

insert into student values('137cs1901','ravi',20,'hospet');
insert into student values('137cs1902','ramya',16,'koppal');

DEFAULT CONSTRAINT: The DEFAULT constraint is used to insert a default value into a column.
The default value will be added to all new records, if no other value is specified.

create table student (regno varchar(10), name varchar(20), age int, city varchar(20) **default** 'koppal');

desc student;

Inserting values into the table

insert into student values('137cs1901','ravi',20,'hospet');

insert into student(regno, name, age) values('137cs1902','ramya',21);

insert into student(regno, name, age) values('137cs1903','raju',20);

select * from student;

5. DATA MANIPULATION : INSERTING VALUES INTO A TABLE

EMPLOYEE

```
insert into employee values('john','t','smith','677','12-jan-87','gptkoppal','m',50000,'677',5);
insert into employee values('franklin','t','smile','679','12-jun-88','hospet','m',30000,'679',5);
insert into employee values('pallavi','c','keshav','680','19-jun-91','ballari','f',25000,'680',4);
insert into employee values('santhosh','v','kumar','685','19-dec-86','gptkoppal','m',55000,'685',3);
insert into employee values('sudha','s','rani','686','26-jan-88','gpt koppal','f',65000,'686',2);
insert into employee values('suneeth','s','kumar','690','26-apr-88','gpt koppal','f',15000,'690',1);
```

```
select * from employee;
```

DEPARTMENT

```
insert into department values('research',5,'12345677,'05-aug-2010');
insert into department values('research',5,'12345677','05-aug-2010');
insert into department values('administration',4,'12345679','15-sep-2011');
insert into department values('design',3,'12345680','15-sep-2013');
insert into department values('analysis',1,'12345685','12-jul-2012');
insert into department values('maintainance',2,'12345686','12-jul-2016');
```

```
select * from department;
```

DEPT_LOCATIONS

```
insert into dept_locations values(1,'koppal');
insert into dept_locations values(2,'hospet');
insert into dept_locations values(3,'hubli');
insert into dept_locations values(4,'ballari');
```

```
select * from dept_locations;
```

WORKS_ON

```
insert into works_on values('690',1,40);
insert into works_on values('686',2,45);
insert into works_on values('685',3,40);
insert into works_on values('680',4,40);
insert into works_on values('679',5,40);
```

select * from works_on;

DEPENDENT

```
insert into dependent values('677','alice','f','05-aug-2010','daughter');
insert into dependent values('679','akshara','f','04-jul-2010','daughter');
insert into dependent values('680','anjan','m','05-jan-2018','son');
insert into dependent values('690','nandini','f','05-aug-2010','sister');
insert into dependent values('686','rathnamma','f','24-aug-1966','mother');
insert into dependent values('685','pallavi','f','19-jun-1991','spouse');
```

select * from dependent;

PROJECT

```
insert into project values('product x',1,'kgf',5);
insert into project values('product xy',2,'kolar',4);
insert into project values('product xyz',3,'bangalore',5);
insert into project values('computerization',4,'kolar',4);
insert into project values('synthesis',5,'kgf',1);
```

select * from project;

6. ILLUSTRATE THE USE OF SELECT STATEMENT

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

SELECT statement can be depicted in two ways, as follows

Syntax1

SELECT *column_name1 ,column_name2* FROM *table_name*;

Syntax2

SELECT * FROM *table_name*;

Create employee table

create table employee(name varchar(10),salary number(10), ssn number(10), dno number(10));

desc employee;

Insert records into the table

insert into employee values('ram',20000,2222, 5);

insert into employee values('ramu',10000,2223,3);

insert into employee values('ravi',25000, 2224,4);

insert into employee values('raju',30000,2225,1);

Retrieve name, ssn from EMPLOYEE table

select name, ssn from employee;

Retrieve name and salary from EMPLOYEE table

select name, salary from employee;

Retrieve name, dno from EMPLOYEE table

select name, dno from employee;

Retrieve name, salary,ssn from EMPLOYEE table

select name, salary, ssn from employee;

Retrieve all fields from EMPLOYEE table

select * from employee;

7. CONDITIONAL RETRIEVAL -WHERE CLAUSE

The WHERE clause is used to filter records obtained from the SELECT statement according to the condition applied. The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax
SELECT *column_name, column_name*
FROM *table_name*
WHERE *column_name = operator value*;

Create employee table

create table employee(name varchar(10),salary number(10),ssn number(10),dno number(10));

desc employee;

Insert records into the table

insert into employee values('ram',20000,101,1);

insert into employee values('raju',10000,102,2);

insert into employee values('ankit,25000,103,2);

insert into employee values('arun, 30000,104,3);

insert into employee values('seetha',50000,105,3);

select * from employee;

Retrieve employee details where ssn is 102

select name, salary, ssn from employee where ssn=102;

Retrieve employee details where salary is less than or equal to 40000

select name, ssn, salary from employee where salary<=40000;

Retrieve employee details where dno=3

select name, ssn, dno from employee where dno=3;

Retrieve employee details where salary is greater than 50000

select * from employee where salary>=50000;

Retrieve employee details whose name is seetha

select * from employee where name='seetha';

8. QUERY SORTED-ORER BY CLAUSE

The ORDER BY keyword is used to sort the result-set by one or more columns either in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default.
To sort the records in a descending order, you can use the DESC keyword.

Syntax
SELECT *column_name*, *column_name*
FROM *table_name*
ORDER BY *column_name*
ASC|DESC, *column_name* ASC|DESC;

Create employee table

create table employee(name varchar(10),salary number(10),ssn number(10),dno number(10));

desc employee;

Insert records into the table

insert into employee values('ram',20000,101,1);

insert into employee values('raju',10000,102,5);

insert into employee values('ankit,25000,104,2);

insert into employee values('arun, 30000,103,4);

insert into employee values('seetha',50000,105,3);

select * from employee;

Retrieve employee names order by fname

select name, salary, ssn from employee order by name;

Retrieve employee names order by ssn

select name, salary, ssn from employee order by ssn;

Retrieve employee details order by salary

select name, salary, ssn from employee order by salary;

select name, salary, ssn from employee order by salary asc;

select name, salary, ssn from employee order by salary desc;

Retrieve employee details order by dno

select name, dno, ssn from employee order by dno;

select name, dno, ssn from employee order by dno asc;

select name, dno, ssn from employee order by dno desc;

9. GROUPING THE RESULT OF QUERY - GROUP BY CLAUSE AND HAVING CLAUSE

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns. Aggregate functions often need an added GROUP BY statement.

Syntax

SELECT column_name, aggregate_function(column_name)

FROM table_name

WHERE column_name operator value GROUP BY column_name;

Create employee table

create table employee(name varchar(10),salary number(10),dno number(10));

desc employee;

Insert the records into the table

insert into employee values('roopa',35000,1);

insert into employee values('roumya',45000,1);

insert into employee values('ravi',40000,2);

insert into employee values('raju',50000,2);

insert into employee values('ramya',55000,3);

insert into employee values('soumya',30000,3);

select * from employee;

Select maximum salary group by dno

select dno, max(salary) from employee group by dno;

Select maximum salary group by dno having count greater than 1

select dno, max(salary) from employee group by dno having count(*)>1;

Select average of salary group by dno

select dno, avg(salary) from employee group by dno;

Select dno, sum of salary and count group by dno

select dno, count(*), sum(salary) from employee group by dno;

Select dno, average of salary group by dno and having dno=1

select dno, avg(salary) from employee group by dno having dno=1;

Select dno, sum of salary group by dno and having dno=2

select dno, sum(salary) from employee group by dno having dno=2;

10. AGGREGATE FUNCTIONS IN SQL (COUNT, SUM, MAX, MIN, AVG)

SQL aggregate functions return a single value, calculated from values in a column.

AVG() - Returns the average value

COUNT() - Returns the number of rows FIRST() - Returns the first value

MAX() - Returns the largest value

MIN() - Returns the smallest value

SUM() - Returns the sum.

Create employee table

create table employee(name varchar(10),salary number(10),dno number(10));

desc employee;

Insert the records into the table

insert into employee values('roopa',15000,50);

insert into employee values('roumya',20000,50);

insert into employee values('ravi',25000,50);

insert into employee values('raju',50000,70);

insert into employee values('ramya',55000,30);

insert into employee values('soumya',30000,30);

select * from employee;

Compute the total salary of all employees

select sum(salary) from employee;

Compute the max and min salary of department 50

select max(salary),min(salary) from employee where dno=50;

Compute the average salary of all employees

select avg(salary) from employee;

Count the total number of employees

select count(*) from employee;

Compute the total ,min, max, avg, sum salary of all employees

select count(*),min(salary),max(salary),avg(salary),sum(salary) from employee;

11. SQL operators (And, Or, Not, In, Between, Like)

Create employee table

create table employee(fname varchar(10), lname varchar(10),salary number(10), ssn number(10), dno number(10));

desc employee;

Insert the records into the table

insert into employee values('roopa','rao'15000,10,50);

insert into employee values('sushma','rani'20000,11,50);

insert into employee values('aniketh','raj'25000,12,50);

insert into employee values('raju','kumar'50000,13,70);

insert into employee values('ankitha','seth'55000,14,30);

insert into employee values('soumya','singh'30000,15,30);

select * from employee;

Retrieve the employees based on salary is 10000 and 30000

select * from employee where salary=25000 and dno=50;

Retrieve employee details whose dno is 50 or 70

select * from employee where dno=50 or dno=70;

Retrieve employee details whose dno is not 50

select * from employee where not dno=50;

Retrieve employee details whose salary is in 15000, 20000, 30000

select * from employee where salary in(15000,20000,30000);

Retrieve the employees based on salary between 10000 and 30000

select * from employee where salary between 10000 and 30000;

Retrieve all employees whose fname starts with letter S.

select fname, lname from employee where fname like 's%';

Retrieve all employees whose lname starts with letter R.

select fname, lname from employee where lname like 'r%';

12. Query multiple tables using JOIN operation.

Create employee table

create table employee(name varchar(10),salary number(10),ssn number(10),dno number(10));

desc employee;

Insert records into the table

insert into employee values('ram',20000,101,18);

insert into employee values('raju',10000,102,19);

insert into employee values('ankit,25000,104,20);

insert into employee values('arun, 30000,103,21);

insert into employee values('seetha',50000,105,22);

select * from employee;

Create department table

create table department (dname varchar(15), dnumber number(10));

desc departtment;

insert into department values('testing', 18);

insert into department values('coding', 19);

insert into department values('debugging', 20);

insert into department values('production', 30);

insert into department values('costing', 19);

select * from department;

EQUIJOIN:

select name, salary, dnumber from employee, department where dno=dnumber;

NON EQUIJOIN:

select name, salary, dnumber from employee, department where dno>dnumber;

OUTER JOIN

LEFT OUTER JOIN:

select dno,name, dnumber from employee left outer join department on (dno=dnumber);

RIGHT OUTER JOIN:

select dno,name, dnumber from employee right outer join department on (dno=dnumber);

FULL OUTER JOIN:

select dno,name, dnumber from employee full outer join department on (dno=dnumber);

13. PERFORM UPDATE, ALTER, DELETE, DROP OPERATIONS ON TABLES

UPDATE

The UPDATE statement is used to update existing records in a table.

Syntax

UPDATE table_name SET column1=value1, column2=value2,...

WHERE some_column=some_value;

ALTER TABLE

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

Syntax

To add a column in a table, use the following syntax:

ALTER TABLE table_name ADD column_name datatype

To modify a column in a table, use the following syntax

ALTER TABLE table_name MODIFY column_name datatype

To delete a column in a table, use the following syntax

ALTER TABLE table_name DROP COLUMN column_name

DELETE

The DELETE statement is used to delete rows in a table.

Syntax

DELETE FROM table_name WHERE some_column=some_value;

DROP

Indexes, tables, and databases can easily be deleted / removed with the DROP statement.

Syntax:

The DROP TABLE statement is used to delete a table.

DROP TABLE table_name;

The DROP DATABASE statement is used to delete a database.

DROP DATABASE database_name;

Create employee table

create table employee(name varchar(10),salary number(10),ssn number(10));

desc employee;

Insert the records into the table

insert into employee values('roopa',10000,11);

insert into employee values('suma',15000,12);

insert into employee values('ravi',20000,13);

insert into employee values('raju',25000,14);

insert into employee values('ramya',35000,15);

insert into employee values('rahitya',30000,16);

select * from employee;

UPDATE

update employee set salary=50000 where dno=14;

select * from employee;

ALTER

alter table employee add address varchar(20);

desc employee;

alter table employee modify name varchar(20);

desc employee;

alter table employee drop column address;

desc employee;

DELETE

delete from employee where ssn=15;

select * from employee;

delete from employee;

select * from employee;

DROP

drop table employee;

14. Illustrate the use of CREATE VIEW command and manipulating

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
We can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

Syntax:
CREATE VIEW view_name AS SELECT column_name(s)
FROM table_name
WHERE condition

A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

Create employee table

create table employee(name varchar(10),ssn number(10),age number(10),salary number(10));

desc employee;

Insert the records into the table

insert into employee values('roopa',150,18,5000);

insert into employee values('suma',151,19,1000);

insert into employee values('ravi',152,20,5000);

insert into employee values('raju',153,30,2000);

select * from employee;

Create a View emp

create view emp as select name,ssn,salary from employee;

desc emp;

select * from emp;

Operations on Views

update emp set name="praveen" where ssn=30;

insert into emp values('ravi',28,18000);

insert into emp values('arju',29,19000);

select * from emp;

delete from emp where ssn=29;

select * from emp;

delete from emp;