

DBMS LAB CHIT SOLUTIONS

(1) Chit 1 –

```
create database bank;
```

```
use bank;
```

```
create table customer (
```

```
-> c_id int auto_increment primary key,
```

```
-> cname varchar(100),
```

```
-> city varchar(100)
```

```
-> );
```

```
create table account (
```

```
-> c_id int,
```

```
-> acc_type varchar(100),
```

```
-> amount int,
```

```
-> );
```

```
insert into customer(cname, city)
```

```
-> values ('John', 'Nashik'),
```

```
-> ('Seema', 'Aurangabad'),
```

```
-> ('Amita', 'Nagar'),
```

```
-> ('Rakesh', 'Pune'),
```

```
-> ('Samata', 'Nashik'),
```

```
-> ('Ankita', 'Chandwad'),
```

```
-> ('Bhavika', 'Pune'),
```

```
-> ('Deepa', 'Mumbai'),
```

```
-> ('Nitin', 'Nagpur'),
```

```
-> ('Pooja', 'Pune');
```

```
insert into account(c_id, acc_type, amount)
```

```
-> values (1, 'Current', 5000),
```

-> (2, 'Saving', 20000),
-> (3, 'Saving', 70000),
-> (4, 'Saving', 50000),
-> (6, 'Current', 35000),
-> (7, 'Loan', 30000),
-> (8, 'Saving', 50000),
-> (9, 'Saving', 90000),
-> (10, 'Loan', 8000),
-> (11, 'Current', 45000);

QUERIES :

- **Show the cname, Acc_Type, amount information of customer who is having an saving account.**

```
select cname, acc_type, amount  
-> from customer  
-> join account  
-> on customer.c_id = account.c_id  
-> where acc_type = 'Saving';
```

- **Display the data using Natural, left and right join.**

```
select * from customer  
-> join account  
-> on customer.c_id = account.c_id;
```

```
select * from customer  
-> left join account  
-> on customer.c_id = account.c_id;
```

```
select * from customer  
-> right join account  
-> on customer.c_id = account.c_id;
```

- **Display the information of customers living in the same city as of 'pooja'.**

```
select * from customer  
-> where city = (select city from customer where cname = 'Pooja');
```

- **Display the information of account, having less amount than average amount throughout the bank.**

```
select * from account
```

```
-> where amount < (select avg(amount) from account);
```

- **Display the C_id having maximum amount in account.**

```
select c_id from account
```

```
-> where amount = (select max(amount) from account);
```

- **Display the amount and acc_type of those customers whose amount is the minimum amount of that Acc_type.**

```
select amount, acc_type
```

```
-> from account
```

```
-> group by acc_type
```

```
-> having amount = min(amount);
```

- **Display the amount of those accounts whose amount is higher than amount of any saving account amount.**

```
select amount from account
```

```
-> where amount > (select min(amount) from account where acc_type = 'Saving');
```

(2) Chit 12 –

```
create database bank;
```

```
use bank;
```

```
create table branch(
```

```
-> branch_name varchar(100) primary key,
```

```
-> branch_city varchar(100) not null,
```

```
-> assets varchar(100)
```

```
-> );
```

```
create table account(
```

```
-> acc_no int primary key,
```

```
-> branch_name varchar(100) not null,
```

```
-> balance int,
```

```
-> foreign key(branch_name) references branch(branch_name),
```

```
-> );
```

```
create table customer(
```

```
-> cust_name varchar(100) primary key,
```

```
-> cust_street varchar(100) not null,
```

```
-> cust_city varchar(100) not null
```

```
-> );
```

```
create table depositor(
```

```
-> cust_name varchar(100),
```

```
-> acc_no int,
```

```
-> foreign key(cust_name) references customer(cust_name),
```

```
-> foreign key(acc_no) references account(acc_no)
```

```
-> );
```

```
create table loan(
```

```
-> loan_no int primary key,
```

```
-> branch_name varchar(100),
```

```
-> amount int not null,
```

```
-> foreign key(branch_name) references branch(branch_name)
```

```
-> );
```

```
create table borrower(
```

```
-> cust_name varchar(100),
```

```
-> loan_no int,
```

```
-> foreign key(cust_name) references customer(cust_name),
```

```
-> foreign key(loan_no) references loan(loan_no)
```

```
-> );
```

insert into branch

-> values ('Akurdi', 'Pune', 'Cash worth 200000'),
-> ('Nigdi', 'Pune', 'Cash worth 4000000'),
-> ('Andheri', 'Mumbai', 'Cash worth 10000000');

insert into customer

-> values ('Raj', 'Ravet', 'Pune'),
-> ('Shivam', 'Punawale', 'Pune'),
-> ('Aditya', 'Akurdi', 'Pune'),
-> ('Abhishek', 'Nigdi', 'Pune');

insert into account

-> values (111, 'Akurdi', 20000),
-> (112, 'Nigdi', 30000),
-> (113, 'Andheri', 50000),
-> (114, 'Andheri', 6000),
-> (115, 'Akurdi', 40000),
-> (116, 'Nigdi', 70000),
-> (117, 'Akurdi', 65000),
-> (118, 'Akurdi', 7400);

insert into depositor

-> values ('Abhishek', 111),
-> ('Aditya', 112),
-> ('Raj', 116);

insert into loan

-> values (1, 'Akurdi', 15000),
-> (2, 'Andheri', 200000),
-> (3, 'Akurdi', 20000),
-> (4, 'Akurdi', 4000),
-> (5, 'Nigdi', 50000),
-> (6, 'Akurdi', 1400),
-> (7, 'Nigdi', 1450);

insert into borrower

-> values ('Raj', 1),
-> ('Shivam', 3),
-> ('Aditya', 5);

QUERIES

- Find the names of all branches in loan relation.

select distinct branch_name from loan;

- **Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.**

select loan_no from loan

-> where branch_name = 'Akurdi' and amount > 12000;

- **Find all customers who have a loan from bank. Find their names, loan_no and loan amount.**

select cust_name, borrower.loan_no, amount as loan_amount

-> from borrower

-> join loan

-> on borrower.loan_no = loan.loan_no;

- **List all customers in alphabetical order who have loan from Akurdi branch.**

select customer.cust_name, cust_street, cust_city from customer

-> join borrower

-> on customer.cust_name = borrower.cust_name

-> join loan

-> on borrower.loan_no = loan.loan_no

-> where branch_name = 'Akurdi';

- **Find all customers who have an account or loan or both at bank.**

select customer.cust_name, cust_street, cust_city

-> from customer

-> join depositor

-> on customer.cust_name = depositor.cust_name;

select customer.cust_name, cust_street, cust_city

-> from customer

-> join borrower

-> on customer.cust_name = borrower.cust_name;

select customer.cust_name, cust_street, cust_city

-> from customer

-> join depositor

-> on customer.cust_name = depositor.cust_name;

-> join borrower

-> on customer.cust_name = borrower.cust_name;

- **Find average account balance at Akurdi branch.**

select avg(balance) as avg_balance

-> from account

-> where branch_name = 'Akurdi';

- **Find the average account balance at each branch**

select branch_name, avg(balance) as avg_balance

-> from account

-> group by branch_name;

- **Find no. of depositors at each branch.**

select branch_name, count(*) as no_of_depositors

-> from depositor

-> join account

-> on depositor.acc_no = account.acc_no

-> group by branch_name;

- **Find the branches where average account balance > 12000.**

select branch_name from account

-> where (select avg(balance) from account) > 12000

-> group by branch_name;

- **Find number of tuples in customer relation.**

select count(*) from customer;

- **Delete all loans with loan amount between 1300 and 1500**

delete from loan

-> where amount between 1300 and 1500;

(3) Chit 13 –

```
create database emp_db;  
use emp_db;
```

```
create table jobs(  
  -> job_id int primary key,  
  -> job_desc varchar(100)  
  -> );
```

```
create table employees(  
  -> employee_id int primary key,  
  -> first_name varchar(100),  
  -> last_name varchar(100),  
  -> job_id int,  
  -> salary int,  
  -> foreign key(job_id) references jobs(job_id)  
  -> );
```

Insert some data in both the tables and try updating & deleting some values an error will occur which is asked in the question

(4) Chit 17 –

```
create database library;  
use library;
```

```
create table library_branch(  
  -> branch_id int primary key,  
  -> branch_name varchar(100),  
  -> address varchar(100)  
  -> );
```

```
create table publisher(  
  -> name varchar(100) primary key,  
  -> address varchar(100),  
  -> phone int  
  -> );
```

```
create table book(  
  -> book_id int primary key,  
  -> title varchar(100),  
  -> publisher_name varchar(100),  
  -> pub_year year(4),  
  -> foreign key(publisher_name) references publisher(name)  
  -> );
```



```
create table book_authors(  
  -> book_id int,  
  -> author_name varchar(100),  
  -> foreign key(book_id) references book(book_id) on delete cascade  
  -> );
```

```
create table book_copies(  
  -> book_id int,  
  -> branch_id int,  
  -> no_of_copies int,  
  -> foreign key(book_id) references book(book_id) on delete cascade,  
  -> foreign key(branch_id) references library_branch(branch_id)  
  -> );
```

```
create table book_lending(  
  -> book_id int,  
  -> branch_id int,  
  -> card_no int,  
  -> date_out date,  
  -> due_date date,  
  -> foreign key(book_id) references book(book_id) on delete cascade,  
  -> foreign key(branch_id) references library_branch(branch_id)  
  -> );
```

```
insert into library_branch  
  -> values (201, 'LIB-A', 'Akurdi'),  
  -> (202, 'LIB-B', 'Nigdi');
```

```
insert into publisher  
  -> values ('X', 'Ravet', 93465),  
  -> ('Y', 'Pimpri', 79931),  
  -> ('Z', 'Chinchwad', 79343);
```

```
insert into book  
  -> values(1, 'C', 'X', 2004),  
  -> (2, 'Java', 'Y', 2006),  
  -> (3, 'Python', 'Z', 2008);
```

```
insert into book_authors  
  -> values(1, 'A'),  
  -> (2, 'B'),  
  -> (3, 'C');
```

```
insert into book_copies  
  -> values (1, 201, 500),  
  -> (1, 202, 1000),  
  -> (2, 201, 300),
```

-> (2, 202, 100),

-> (3, 201, 800);

insert into book_lending

-> values(1, 201, 111, '2022-02-25', '2022-03-25'),

-> (2, 202, 222, '2022-09-13', '2022-10-13');

QUERIES

- **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch,etc.**

select book.book_id, title, publisher_name, author_name, branch_name, no_of_copies

-> from book

-> join book_authors on book.book_id = book_authors.book_id

-> join book_copies on book.book_id = book_copies.book_id

-> join library_branch on book_copies.branch_id = library_branch.branch_id;

- **Get the particulars of borrowers who have borrowed from Jan 2017 to Jun2017**

select title, branch_name, card_no, date_out, due_date

-> from book_lending

-> join book on book_lending.book_id = book.book_id

-> join library_branch on book_lending.branch_id = library_branch.branch_id

-> where date_out between '2017-01-01' and '2017-06-30';

- **Delete a book name “Databases” from BOOK table.**

delete from book where title = 'Databases';

- **Print total number of books as yearwise.**

select count(*) as total_no_of_books, pub_year

-> from book

-> group by pub_year

-> order by pub_year;

- **Create a view of all books and its number of copies that are currently available in the Library**

create view books as

```
-> select title, sum(no_of_copies) as copies
-> from book
-> join book_copies on book.book_id = book_copies.book_id
-> group by book.book_id;
```

```
select * from books;
```

(5) Chit 18 –

```
create database emp_db;
```

```
use emp_db;
```

```
create table employee(
```

```
-> eid int auto_increment primary key,
-> ename varchar(100),
-> address varchar(100),
-> salary int,
-> commision int
-> );
```

```
create table project(
```

```
-> prno int primary key,
-> addr varchar(100)
-> );
```

```
insert into employee(ename, address, salary, commision)
```

```
-> values ('Amit', 'Pune', 35000, 5000),
-> ('Sneha', 'Pune', 25000, null),
-> ('Savita', 'Nashik', 28000, 2000),
-> ('Pooja', 'Mumbai', 19000, null),
-> ('Sagar', 'Mumbai', 25000, 3000);
```

```
insert into project
```

```
-> values (10, 'Mumbai'),
-> (20, 'Pune'),
-> (30, 'Jalgaon');
```

QUERIES

- **Find different locations from where employees belong to?**

select distinct address from employee;

- **What is maximum and minimum salary?**

select max(salary), min(salary) from employee;

- **Display the content of employee table according to the ascending order of salary amount.**

select * from employee order by salary;

- **Find the name of employee who lived in Nashik or Pune city.**

select ename, address from employee
-> where address in ('Nashik', 'Pune');

- **Find the name of employees who does not get commission.**

select ename from employee
-> where commission is null;

- **Change the city of Amit to Nashik.**

update employee set address = 'Nashik'
-> where ename = 'Amit';

select * from employee;

- **Find the information of employees whose name starts with 'A'.**

select * from employee where ename like 'A%';

- **Find the count of staff from Mumbai.**

select count(*) from employee where address = 'Mumbai';

- **Find the count of staff from each city**

select count(*) as count_of_staff, address
-> from employee
-> group by address;

- **Find the address from where employees are belonging as well as where projects are going on.**

select ename, prno, address from employee
 -> join project on employee.address = project.addr;

- **Find city wise minimum salary.**

select address, min(salary) from employee
 -> group by address;

- **Find city wise maximum salary having maximum salary greater than 26000**

select address, max(salary) as max_salary
 -> from employee
 -> group by address
 -> having max(salary) > 26000;

- **Delete the employee who is having salary greater than 30,000.**

delete from employee where salary > 30000;
 select * from employee;

(6) Chit 19 –

create database emp_db;

use emp_db;

create table emp(

-> eno int auto_increment primary key,
 -> ename varchar(30) not null,
 -> address varchar(100) default 'Nashik',
 -> salary int not null,
 -> joindate date
 ->);

alter table emp auto_increment = 101;

QUERIES

- **After table creation add field - Post in the emp table.**

alter table emp add post varchar(100);

- **Insert some data in emp table. Create Index on Ename field of employee table.**

```
insert into emp(ename, address, salary, joindate, post)
```

```
-> values ('Amit', 'Pune', 25000, '2017-05-23', 'HR'),
```

```
-> ('Sneha', 'Pune', 35000, '2018-11-07', 'Manager'),
```

```
-> ('Savita', 'Nashik', 28000, '2019-06-13', 'Technical Head'),
```

```
-> ('Pooja', 'Mumbai', 19000, '2020-08-09', 'Public Relations Head'),
```

```
-> ('Sagar', 'Mumbai', 25000, '2021-09-17', 'Marketing Head');
```

```
create index emp_name on emp(ename);
```

- **Create View on employee table to show only Ename and Salary.**

```
create view emp_data as
```

```
-> select ename, salary from emp;
```

```
select * from emp_data;
```

(7) Chit 23 –

```
create database hospital;
```

```
use hospital;
```

```
create table physician(
```

```
-> reg_no int auto_increment primary key,
```

```
-> name varchar(100),
```

```
-> tel_no int,
```

```
-> city varchar(100)
```

```
-> );
```

```
create table patient(
```

```
-> p_name varchar(100) primary key,
```

```
-> street varchar(100),
```

```
-> city varchar(100)
```

```
-> );
```

```
create table visit(
```

```
-> p_name varchar(100),
```

```
-> reg_no int,
```

```
-> date_of_visit date,
```

```
-> fee int,
```

```
-> foreign key(p_name) references patient(p_name),
```

```
-> foreign key(reg_no) references physician(reg_no)
```

->);

insert into physician(name, tel_no, city)

-> values ('Amit', 45697, 'Pune'),

-> ('Raju', 47964, 'Mumbai'),

-> ('Ashish', 23974, 'Nashik'),

-> ('Chitra', 17794, 'Nagpur');

insert into patient

-> values ('Sneha', 'Ravet', 'Pune'),

-> ('Savita', 'Lalbaug', 'Mumbai'),

-> ('Pooja', 'Pimpri', 'Pune'),

-> ('Sagar', 'Koregaon', 'Mumbai');

insert into visit

-> values ('Sneha', 1, '2017-07-13', 300),

-> ('Savita', 2, '2017-07-13', 500),

-> ('Pooja', 3, '2021-03-23', 400),

-> ('Sagar', 4, '2022-06-14', 1000);

QUERIES

- **Find the name and city of patients who visited a physician on 13 July 2017.**

select patient.p_name, city from patient

-> join visit on patient.p_name = visit.p_name

-> where date_of_visit = '2017-07-13';

- **Get the name of the physician and the total no. of Patients visited him. Give the details of date wise fees collected at clinic.**

select name, count(*) as total_no_of_patients

-> from physician

-> join visit on physician.reg_no = visit.reg_no

-> group by visit.reg_no;

select date_of_visit, sum(fee) as total_fees

-> from visit

-> group by date_of_visit;

- **Show details of all visitors details.**

select patient.p_name, street, city

-> from patient

-> join visit on patient.p_name = visit.p_name;

- **Create view for visitor who visited in year 2021 to 2022.**

create view visitors as

-> select * from visit

-> where date_of_visit between '2021-01-01' and '2022-12-31';

select * from visitors;

- **Create index on p_name**

create index patient_name

-> on patient(p_name);