BANK CASE STUDY :- SQL

# PROBLEM STATEMENT :-

A bank wants to create a database to manage its customers, accounts, transactions, and loans. They need to track customer information, account balances and types, transaction history, and loan details. The bank wants to be able to perform various queries, such as finding customers with a specific account type, calculating interest earned, and identifying overdue loans.

# ER DIAGRAM :-

# 

# RELATIONSHIPS:-

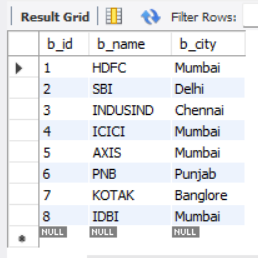
* **Bank has Branches => 1 : N**  
  One Bank can have many Branches but one Branch can not belong to many Banks, so the relationship between Bank and Branch is one to many relationship.
* **Branch maintain Accounts => 1 : N**   
  One Branch can have many Accounts but one Account can not belong to many Branches, so the relationship between Branch and Account is one to many relationship.
* **Branch offer Loans => 1 : N**   
  One Branch can have many Loans but one Loan can not belong to many Branches, so the relationship between Branch and Loan is one to many relationship.
* **Account held by Customers => M : N**   
  One Customer can have more than one Accounts and also One Account can be held by one or more Customers, so the relationship between Account and Customers is many to many relationship.
* **Loan availed by Customer => M : N**   
  (Assume loan can be jointly held by many Customers).   
  One Customer can have more than one Loans and also One Loan can be availed by one or more Customers, so the relationship between Loan and Customers is many to many relationship.

# KEY ENTITIES :-

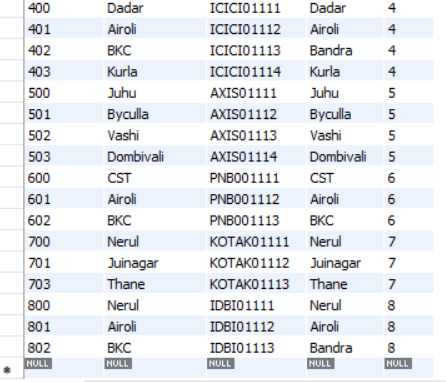
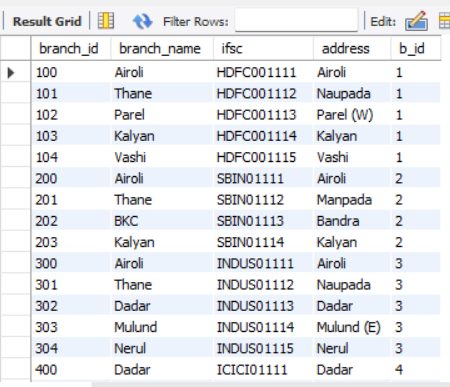
* **Bank Entity :** Attributes of Bank Entity are Bank Name, Code and Address.   
  Code is Primary Key for Bank Entity.
* **Customer Entity :** Attributes of Customer Entity are Customer\_id, Name, Phone Number and Address.   
  Customer\_id is Primary Key for Customer Entity.
* **Branch Entity :** Attributes of Branch Entity are Branch\_id, Name and Address.   
  Branch\_id is Primary Key for Branch Entity.
* **Account Entity :** Attributes of Account Entity are Account\_number, Account\_Type and Balance.   
  Account\_number is Primary Key for Account Entity.
* **Loan Entity :** Attributes of Loan Entity are Loan\_id, Loan\_Type and Amount.   
  Loan\_id is Primary Key for Loan Entity.

# TABLES :-

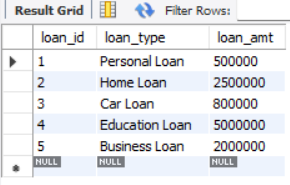
1. BANK :-



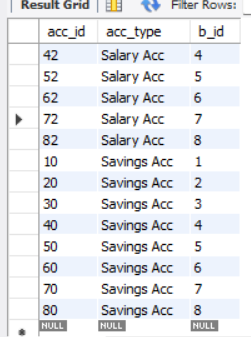
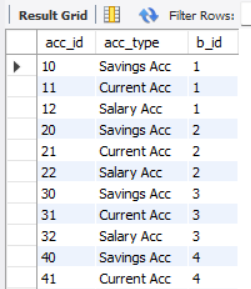
1. BRANCHES :-



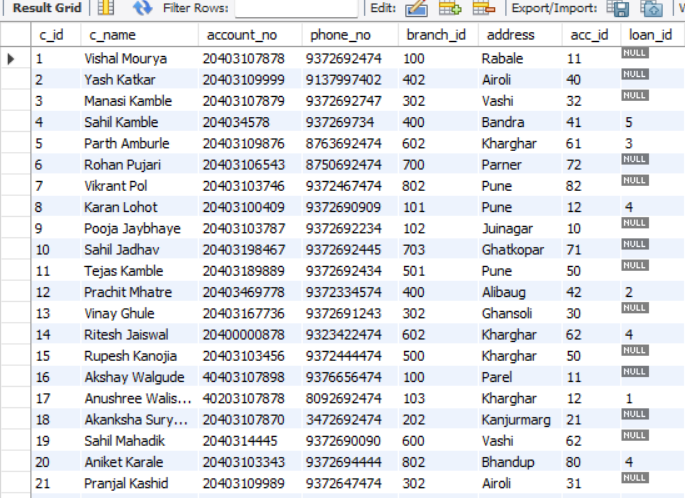
1. LOAN :-

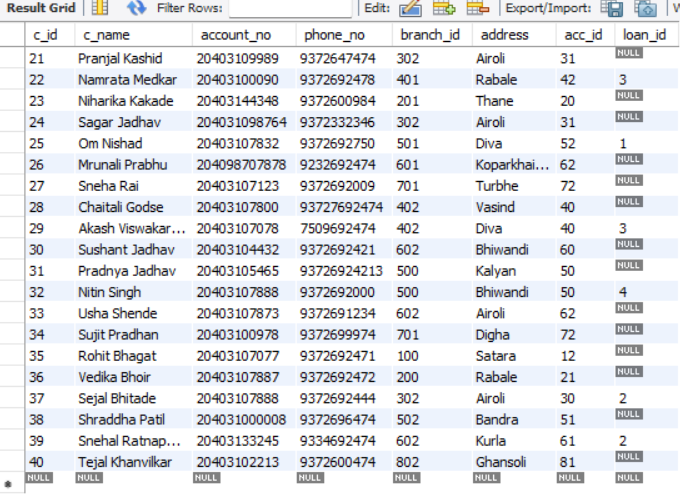


1. ACCOUNTS :-



1. CUSTOMER :-





# QUERRIES :-

/\*================ QUERRY TO SELECT DATABASE =========================\*/

use bank ;

/\*================ QUERRY TO VIEW STRUCTURE OF TABLE =========================\*/

Desc Bank ;

Desc Branch ;

Desc account ;

Desc loan ;

Desc customer ;

/\*================ QUERRY TO VIEW TABLE DATA =========================\*/

select \* from Bank ;

select \* from Branch ;

select \* from account ;

select \* from loan ;

select \* from customer ;

/\*================ QUERRY ALTER THE TABLE DATA =========================\*/

alter table account rename to Account ;

alter table Bank change column b\_location b\_city varchar(15) ;

alter table Bank modify b\_city varchar(15) default('Mumbai') ;

/\*================ QUERRY UPDATE THE TABLE DATA =========================\*/

update loan set loan\_amt = 2500000 where loan\_id = 2 ;

update customer set loan\_id = 4 where c\_id =14 ;

update Bank set b\_loacation = "Kolkata" where b\_name = "IDBI" ;

update customer set address = "Airoli" where c\_name="Pranjal Kashid" ;

/\*================ WHERE CLAUSE FOR SEARCH CONDITION =========================\*/

select \* from Bank where b\_name = "KOTAK" ;

select c\_id ,c\_name , account\_no , branch\_id , address from customer where c\_id = 21 ;

select b\_name as Name , b\_id from Bank where b\_city = "Mumbai";

/\*================ OPERATORS =========================\*/

/\* Arithmatic operator\*/

select loan\_id , loan\_type , loan\_amt + 50000 as Updated\_Amt from loan where loan\_id = 1 ;

/\*Comparison operator\*/

select loan\_type from loan where loan\_amt > 700000 ;

select c\_name from customer where address = "Airoli" ;

select c\_name from customer where address != "Airoli" ;

select c\_name from customer where address <> "Kalyan" ;

/\* Logical Oprator\*/

select \* from customer where address = "Kalyan" and loan\_id = 3 ;

select \* from customer where address = "Thane" or loan\_id = 3 ;

/\*Range Operator\*/

select \* from customer where c\_id between 21 and 35 ;

/\*================ CLAUSES =========================\*/

/\*Distinct Clause \*/

select distinct address from customer ;

/\* IN Clause \*/

select \* from loan where loan\_id in (1 , 3, 5) ;

select \* from customer where address in ("Kalyan" , "Airoli", "BKC");

/\*ORDER BY Clause \*/

select \* from customer order by c\_name asc ;

select \* from customer order by c\_name desc ;

/\*LIMIT Claude \*/

select \* from loan order by loan\_amt desc limit 2 ;

select \* from loan order by loan\_amt desc limit 1,2 ;

/\* Like Clause \*/

select c\_name from customer where c\_name like ("P%") ;

select c\_name from customer where c\_name like ("\_\_a%") ;

select c\_name from customer where c\_name like ("\_a%") ;

select c\_name from customer where c\_name like ("%as%") ;

/\*============= INBUILD FUNCTIONS =======================\*/

/\* String function \*/

select concat(c\_name,'\_',address) as Cust\_Details from customer ;

select upper (c\_name) as Names from customer;

select lower(c\_name )as Names from customer ;

select replace (b\_city ,'Chennai','Chennai\_') from Bank ;

select length(address) as len from customer ;

/\*Substring Functiom\*/

select substring('Good\_Morning', 5,13) as result ;

/\*Date Function\*/

select date\_format(now(), '%D %M %Y') as result;

/\*===== AGGREGATE FUNCTION (GROUP BY CLAUSE)-- count, min, max, avg, sum ====\*/

select count(c\_name) from customer group by loan\_id having loan\_id = 3 ;

select min(loan\_amt) from loan group by loan\_amt ;

select max(loan\_amt) from loan group by loan\_amt ;

select avg(loan\_amt) as avg\_amt from loan group by loan\_amt ;

/\*======================== JOINS =======================================\*/

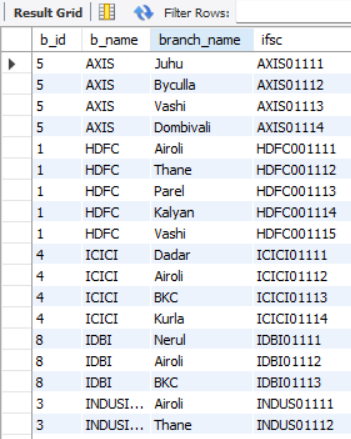
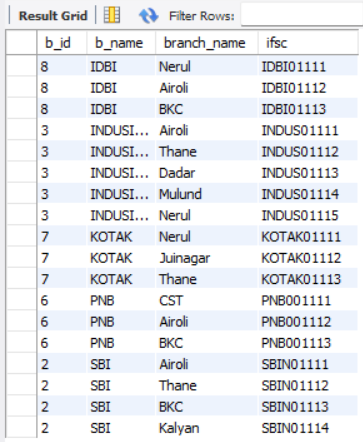
/\* Q1. Write a querry to create a table that joins the Bank And Branch tables which displays bankid , Bank name and its Branches and IFSC Code. \*/

select Bank.b\_id , Bank.b\_name , Branch.branch\_name , Branch.ifsc from Bank

INNER JOIN Branch

WHERE Bank.b\_id = Branch.b\_id ;

OUTPUT:-

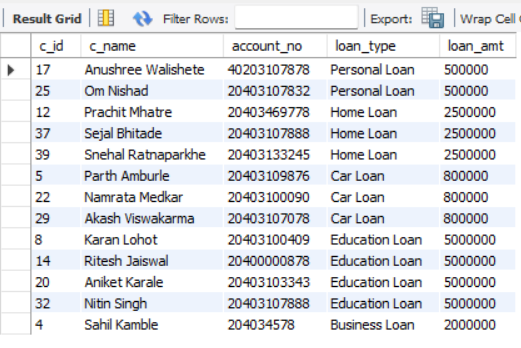
/\*Q2. Write a querry to display the Customers who have availed Loans from their respective Banks . Querry must return cust id , cust name , cust Account no ,Loan type , loan amt . \*/

SELECT customer.c\_id, customer.c\_name , customer.account\_no , loan.loan\_type , loan.loan\_amt from customer

INNER JOIN loan

WHERE customer.loan\_id=loan.loan\_id ;

OUTPUT:-



/\*======================= SUB QUERRIES ===============================\*/

/\* Q3. Write a querry to find the Loan id of customers whose loan amount is greater than the loan amont of customer with customer id =12 \*/

select loan\_id from loan where loan\_id > any (select loan\_id from customer where c\_id = 12) ;

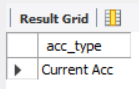
OUTPUT:-



/\*Q4. Write a querry to find the Type of Account of the customer having accoun no = 20403109989 \*/

select acc\_type from account where acc\_id in (select acc\_id from customer where account\_no = 20403109989 );

OUTPUT:-



/\*========================= VIEWS =====================================\*/

/\*Q5. Write a querry to create a view as Account\_Details which displays the name of custome , phone number , account type ,Bank name.\*/

Create view Account\_Details as

select customer.c\_name , customer.phone\_no , account.acc\_type , Bank.b\_name from customer

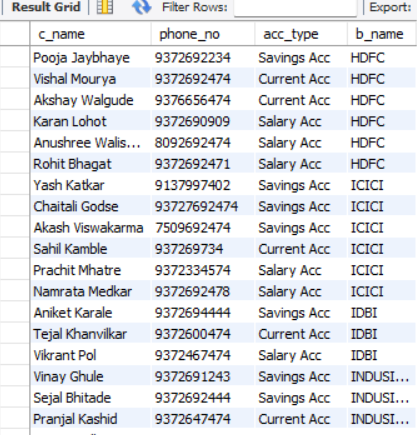
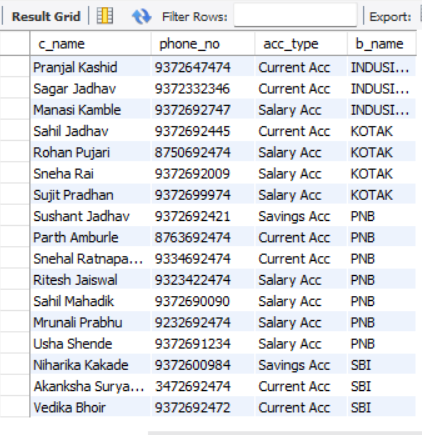
INNER JOIN account

ON customer.acc\_id = account.acc\_id

INNER JOIN Bank

ON account.b\_id = Bank.b\_id ;

OUTPUT:-

/\*======================== CURSOR =====================================\*/

/\*Q6. Create a Curse to Display the Loan Amount Remark such that if Loan Amount is between 5L to 8L then dispay ' Not high ' , if Loan Amount is between 8L to 15L then dispay ' Moderate ' , if Loan Amount is between 15L to 20L then dispay ' high ' if Loan Amount is Greater than 20L then dispay ' Very high ' \*/

delimiter $$

create procedure Loan\_Reminder ()

begin

declare cur cursor for select loan\_amt from loan ;

open cur ;

if loan\_amt >= 500000 or loan\_amt <=800000 then

Select "Loan Amt is Not High" as Loan\_Remark ;

elseif loan\_amt > 800000 or loan\_amt <1500000 then

Select "Loan Amt is Moderate" as Loan\_Remark ;

elseif loan\_amt >= 1500000 or loan\_amt <2000000 then

Select "Loan Amt is High" as Loan\_Remark ;

else

Select "Loan Amt is Very High" as Loan\_Remark ;

end if;

close cur ;

end $$

delimiter ;

/\*======================== TRIGGERS ===============================\*/

/\*Q7. Write a querry that reflects the creation of new account along with its date and time \*/

create table logtable ( action varchar (30) , datetime datetime ) ;

delimiter $$

create trigger Acc\_Created after insert on customer

for each row

begin

insert into logtable(action , datetime ) values ("Account\_Created" , now()) ;

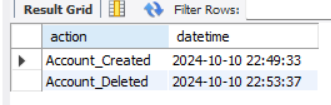
end $$

delimiter ;

Insert into customer values (42 , "Krutika Khanvilkar" , 20403100367, 9372690765 , 301 , "Ghansoli" , 31 , null ) ;

select \* from logtable ;

OUTPUT:-



/\*Q8. Write a querry that reflects the Deletion of existing account along with its date and time \*/

delimiter $$

create trigger Acc\_Deleted after delete on customer

for each row

begin

insert into logtable(action , datetime ) values ("Account\_Deleted" , now()) ;

end $$

delimiter ;

delete from customer where c\_id = 42 ;

select \* from logtable ;

OUTPUT:-

