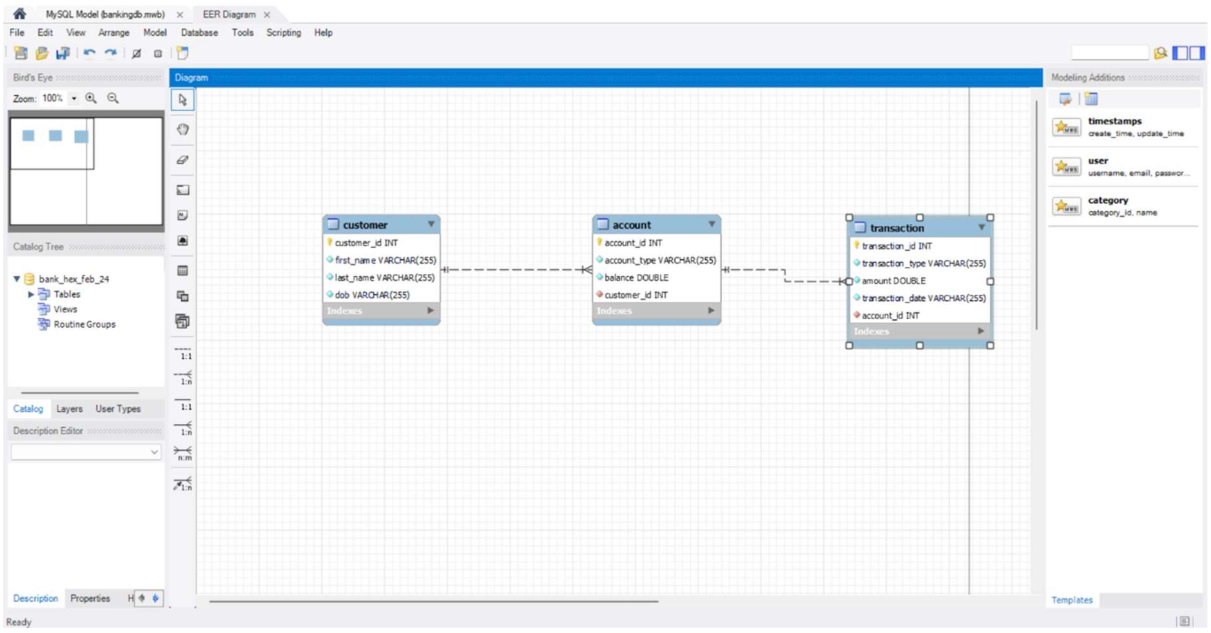
Assignment-2

name	Yalini Shree P V
serial no.	31
Topic	banking system

ER diagram for reference



SQL code with outputs

-- mysql workbench forward engineering

-- ---

-- schema bank_hex_feb_24

-- ---

-- ---

-- schema bank_hex_feb_24

-- ---

create schema if not exists `bank_hex_feb_24` default character set utf8 ;

use `bank_hex_feb_24` ;

-- ---

-- table `bank_hex_feb_24`.`customer`

```
create table if not exists `bank_hex_feb_24`.`customer` (
  `customer_id` int not null auto_increment,
  `first_name` varchar(255) not null,
  `last_name` varchar(255) not null,
  `dob` varchar(255) not null,
  primary key (`customer_id`))
engine = innodb;
-- -----------------------------------------------------
-- table `bank_hex_feb_24`.`account`
-- -----------------------------------------------------
create table if not exists `bank_hex_feb_24`.`account` (
  `account_id` int not null auto_increment,
  `account_type` varchar(255) not null,
  `balance` double not null,
  `customer_id` int not null,
  primary key (`account_id`),
  index `fk_account_customer_idx` (`customer_id` asc) ,
  constraint `fk_account_customer`
    foreign key (`customer_id`)
    references `bank_hex_feb_24`.`customer` (`customer_id`)
    on delete no action
    on update no action)
engine = innodb;
-- -----------------------------------------------------
-- table `bank_hex_feb_24`.`transaction`
-- -----------------------------------------------------
create table if not exists `bank_hex_feb_24`.`transaction` (
  `transaction_id` int not null auto_increment,
  `transaction_type` varchar(255) not null,
  `amount` double not null,
  `transaction_date` varchar(255) not null,
  `account_id` int not null,
  primary key (`transaction_id`),
  index `fk_transaction_account1_idx` (`account_id` asc),
  constraint `fk_transaction_account1`
    foreign key (`account_id`)
    references `bank_hex_feb_24`.`account` (`account_id`)
    on delete no action
    on update no action)
engine = innodb;
use bank_hex_feb_24;

insert into customer(first_name,last_name,dob) values
('harry','potter','2002-03-21'),
('ronald','weasley','2001-02-10'),
('hermione','granger','2002-11-15');

insert into account(account_type,balance,customer_id) values
('savings',50000,1) ,
('current',120000,2) ,
('zero_balance',100000,3),
('current',150000,1) ,
('savings',30000,3);

insert into transaction(transaction_type,amount,transaction_date,account_id)
values
('deposit', 10000, '2024-02-01',1),
('withdrawal', 5000, '2024-02-02',1),
('deposit', 20000, '2024-02-02',2),
('withdrawal', 8000, '2024-02-02',3),
('transfer', 20000, '2024-02-01',4),
('transfer', 7000, '2024-02-05',5);
```

task 2

1. retrieve the name, account type and email of all customers.

```
select c.first_name,a.account_type
from customer c
join account a on c.customer_id = a.customer_id;
```

```
harry    savings
ronald   current
hermione        zero_balance
harry    current
hermione        savings
```

2. list all transactions corresponding to each customer.

```
select c.first_name,t.*
from customer c
join account a on c.customer_id = a.customer_id
join transaction t on a.account_id = t.account_id;
```

```
harry    1       deposit 10000   2024-02-01      1
harry    2       withdrawal      5000    2024-02-02      1
harry    5       transfer 20000  2024-02-01      4
ronald   3       deposit 20000   2024-02-02      2
hermoine        4       withdrawal      8000    2024-02-02      3
hermoine        6       transfer 7000   2024-02-05      5
```

3. increase the balance of a specific account by a certain amount.

```
update account
set balance = balance + 5000
where account_id = 1;
select * from account;
```

```
1        savings 60000   1
2        current 120000 2
3        zero_balance    100000 3
4        current 150000 1
```

```
5          savings 30000   3
```

4. combine first and last names of customers as a full_name.

```
select concat(first_name,' ',last_name)
from customer;
```

```
harry potter
ronald weasley
hermione granger
```

5. remove accounts with a balance of zero where the account type is savings.

```
delete from account
where balance = 0 and account_type = 'savings';
```

6. find customers living in a specific city.

7. get the account balance for a specific account.

```
select balance
from account
where account_id = 1;
```

```
60000
```

8. list all current accounts with a balance greater than $1,000.

```
select *
from account
where account_type = 'current' and balance > 1000;
```

```
2          current 120000 2
4          current 150000 1
```

9. retrieve all transactions for a specific account.

```
select *
from transaction
```

```
where amount=5000;
```

```
2          withdrawal       5000     2024-02-02       1
```

10. calculate the interest accrued on savings accounts based on a given interest rate.

11. identify accounts where the balance is less than a specified overdraft limit.

```
select *
from account
where balance>100000;
```

```
2          current 120000 2
4          current 150000 1
```

12. find customers not living in a specific city.

tasks 3: aggregate functions, having, order by, groupby and joins:

1. write a sql query to find the average account balance for all customers.

```
select avg(balance) as avg_balance
from account;
```

```
90000
```

2. write a sql query to retrieve the top 10 highest account balances.

```
select customer_id, balance
from account
order by balance desc
limit 10;
```

```
1          150000
2          120000
3          100000
1          50000
3          30000
```

3. write a sql query to calculate total deposits for all customers in specific date.

```
select sum(amount) as total_deposits
from transaction
where transaction_type = 'deposit' and transaction_date = '2024-02-01';
```

```
10000
```

4. write a sql query to find the oldest and newest customers.

```
select min(dob) as oldest_customer_dob, max(dob) as newest_customer_dob
from customer;
```

```
2001-02-10      2002-11-15
```

5. write a sql query to retrieve transaction details along with the account type.

```
select t.*, a.account_type
from transaction t
join account a on t.account_id = a.account_id;
```

```
1       deposit 10000   2024-02-01      1       savings
2       withdrawal      5000    2024-02-02      1       savings
3       deposit 20000   2024-02-02      2       current
4       withdrawal      8000    2024-02-02      3       zero_balance
5       transfer 20000  2024-02-01      4       current
6       transfer 7000   2024-02-05      5       savings
```

6. write a sql query to get a list of customers along with their account details.

```
select c.*, a.*
from customer c
join account a on c.customer_id = a.customer_id;
```

```
1       harry   potter  2002-03-21      1       savings 50000   1
2       ronald  weasley 2001-02-10      2       current 120000 2
3       hermione        granger 2002-11-15      3       zero_balance    100000 3
1       harry   potter  2002-03-21      4       current 150000 1
3       hermione        granger 2002-11-15      5       savings 30000   3
```

7. write a sql query to retrieve transaction details along with customer information for a specific account.

```
select t.*, c.first_name, c.last_name
from transaction t
join account a on t.account_id = a.account_id
join customer c on a.customer_id = c.customer_id
where t.account_id = 1;
```

```
1    deposit 10000    2024-02-01    1    harry    potter
2    withdrawal    5000    2024-02-02    1    harry    potter
```

8. write a sql query to identify customers who have more than one account.

```
select customer_id, count(*) as num_accounts
from account
group by customer_id
having count(*) > 1;
```

```
1    2
3    2
```

9. write a sql query to calculate the difference in transaction amounts between deposits and withdrawals.

10. write a sql query to calculate the average daily balance for each account over a specified period.

11. calculate the total balance for each account type.

12. identify accounts with the highest number of transactions order by descending order.

13. list customers with high aggregate account balances, along with their account types.

14. identify and list duplicate transactions based on transaction amount, date, and account

task 4: subquery and its type:

1. retrieve the customer(s) with the highest account balance.

```
select c.first_name, c.last_name, a.balance

from customer c

join account a on c.customer_id = a.customer_id

where a.balance = (select max(balance) from account);
```

```
harry    potter   150000
```

2. calculate the average account balance for customers who have more than one account.

```
select avg(balance) as avg_balance

from account

where customer_id in (

   select customer_id

   from account

   group by customer_id

   having count(*) > 1

);
```

```
82500
```

3. retrieve accounts with transactions whose amounts exceed the average transaction amount.

```
select a.*

from account a

join transaction t on a.account_id = t.account_id

where t.amount > (

   select avg(amount)

   from transaction

);
```

```
2        current 120000 2

4        current 150000 1
```

4. identify customers who have no recorded transactions.

```
select c.*
from customer c
where c.customer_id not in (
   select distinct customer_id
   from transaction
);
```

nill

5. calculate the total balance of accounts with no recorded transactions.

```
select sum(balance) as total_balance_no_transactions
from account
where account_id not in (
   select distinct account_id
   from transaction
);
```

nill

6. retrieve transactions for accounts with the lowest balance.

```
select t.*
from transaction t
join (
   select account_id, min(balance) as min_balance
   from account
) a on t.account_id = a.account_id
where t.amount = a.min_balance;
```

7. identify customers who have accounts of multiple types.

```
select customer_id
from account
group by customer_id
```

```
having count(distinct account_type) > 1;
```

```
1
3
```

8. calculate the percentage of each account type out of the total number of accounts.

9. retrieve all transactions for a customer with a given customer_id.

```
select *
from transaction
where account_id in (
   select account_id
   from account
   where customer_id = 1
);
```

```
1        deposit 10000   2024-02-01      1
2        withdrawal      5000    2024-02-02      1
5        transfer 20000  2024-02-01      4
```

10. calculate the total balance for each account type, including a subquery within the select clause.