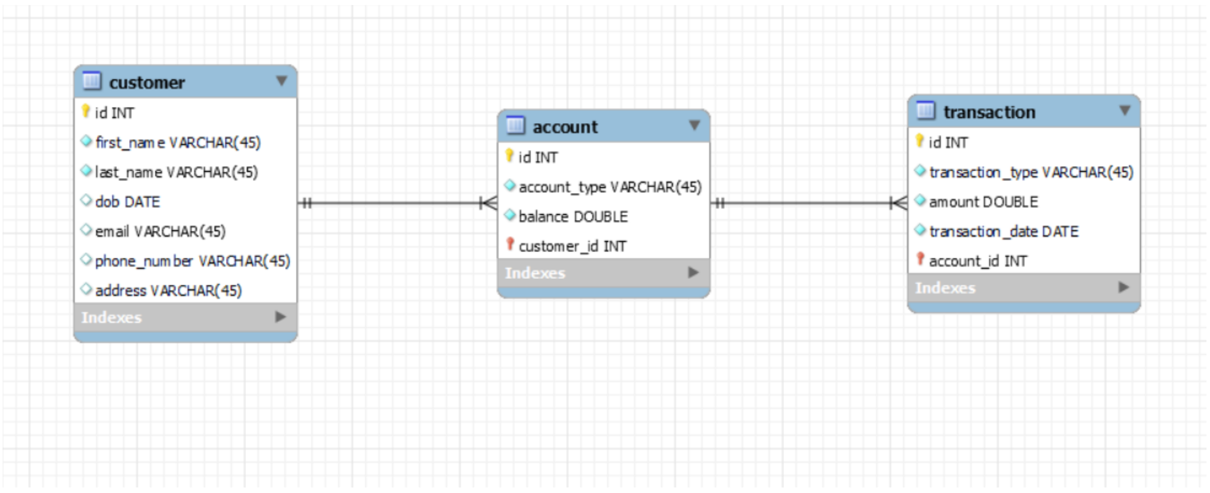
ASSIGNMENT 3

BANKING

ER DIAGRAM:



Task 1: Database Design

-- MySQL Workbench Forward Engineering

-- ---

-- Schema assignment_banking

-- ---

-- ---

-- Schema assignment_banking

-- ---

CREATE SCHEMA IF NOT EXISTS `assignment_banking` DEFAULT CHARACTER SET utf8 ;

USE `assignment_banking` ;

-- ---

-- Table `assignment_banking`.`customer`

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_banking`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `dob` DATE NULL,
  `email` VARCHAR(45) NULL,
  `phone_number` VARCHAR(45) NULL,
  `address` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
COMMENT = '          ';


-- -----------------------------------------------------
-- Table `assignment_banking`.`account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_banking`.`account` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `account_type` VARCHAR(45) NOT NULL,
  `balance` DOUBLE NOT NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`id`, `customer_id`),
  INDEX `fk_account_customer_idx` (`customer_id` ASC) ,
  CONSTRAINT `fk_account_customer`
    FOREIGN KEY (`customer_id`)
    REFERENCES `assignment_banking`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `assignment_banking`.`transaction`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_banking`.`transaction` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `transaction_type` VARCHAR(45) NOT NULL,
  `amount` DOUBLE NOT NULL,
  `transaction_date` DATE NOT NULL,
  `account_id` INT NOT NULL,
  PRIMARY KEY (`id`, `account_id`),
  INDEX `fk_transaction_account1_idx` (`account_id` ASC) ,
  CONSTRAINT `fk_transaction_account1`
    FOREIGN KEY (`account_id`)
    REFERENCES `assignment_banking`.`account` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

INSERTION

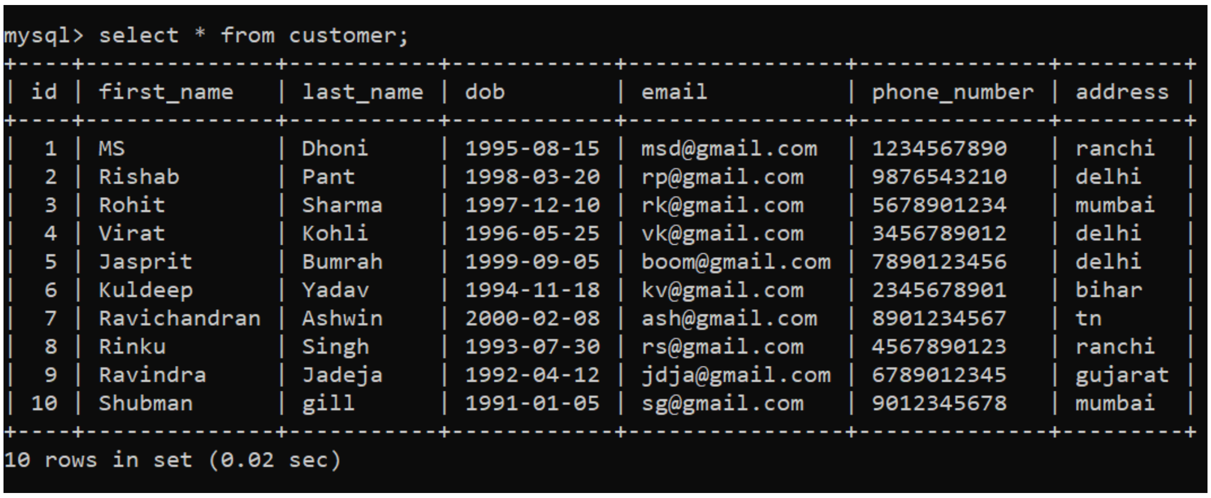
```
--- customer insertion
INSERT INTO customer (id,first_name, last_name, dob, email, phone_number,address)
VALUES
    (1,'MS', 'Dhoni', '1995-08-15', 'msd@gmail.com', '1234567890','ranchi'),
    (2,'Rishab', 'Pant', '1998-03-20', 'rp@gmail.com', '9876543210','delhi'),
    (3,'Rohit', 'Sharma', '1997-12-10', 'rk@gmail.com', '5678901234','mumbai'),
    (4,'Virat', 'Kohli', '1996-05-25', 'vk@gmail.com', '3456789012','delhi'),
    (5,'Jasprit', 'Bumrah', '1999-09-05', 'boom@gmail.com', '7890123456','delhi'),
    (6,'Kuldeep', 'Yadav', '1994-11-18', 'kv@gmail.com', '2345678901','bihar'),
```

(7,'Ravichandran', 'Ashwin', '2000-02-08', 'ash@gmail.com', '8901234567','tn'),

(8,'Rinku', 'Singh', '1993-07-30', 'rs@gmail.com', '4567890123','ranchi'),

(9,'Ravindra', 'Jadeja', '1992-04-12', 'jdja@gmail.com', '6789012345','gujarat'),

(10,'Shubman', 'gill', '1991-01-05', 'sg@gmail.com', '9012345678','mumbai');



```
mysql> select * from customer;
+----+--------------+-----------+------------+----------------+--------------+---------+
| id | first_name   | last_name | dob        | email          | phone_number | address |
+----+--------------+-----------+------------+----------------+--------------+---------+
|  1 | MS           | Dhoni     | 1995-08-15 | msd@gmail.com  | 1234567890   | ranchi  |
|  2 | Rishab       | Pant      | 1998-03-20 | rp@gmail.com   | 9876543210   | delhi   |
|  3 | Rohit        | Sharma    | 1997-12-10 | rk@gmail.com   | 5678901234   | mumbai  |
|  4 | Virat        | Kohli     | 1996-05-25 | vk@gmail.com   | 3456789012   | delhi   |
|  5 | Jasprit      | Bumrah    | 1999-09-05 | boom@gmail.com | 7890123456   | delhi   |
|  6 | Kuldeep      | Yadav     | 1994-11-18 | kv@gmail.com   | 2345678901   | bihar   |
|  7 | Ravichandran | Ashwin    | 2000-02-08 | ash@gmail.com  | 8901234567   | tn      |
|  8 | Rinku        | Singh     | 1993-07-30 | rs@gmail.com   | 4567890123   | ranchi  |
|  9 | Ravindra     | Jadeja    | 1992-04-12 | jdja@gmail.com | 6789012345   | gujarat |
| 10 | Shubman      | gill      | 1991-01-05 | sg@gmail.com   | 9012345678   | mumbai  |
+----+--------------+-----------+------------+----------------+--------------+---------+
10 rows in set (0.02 sec)
```

--- account insertion

insert into account(account_type,balance,customer_id)values

('savings',50000,1),

('current',120000,2),

('zero_balance',100000,3),

('savings',50000,4),

('savings',500000,5),

('savings',20000,6),

('savings',30000,7),

('savings',40000,8),

('savings',70000,9),

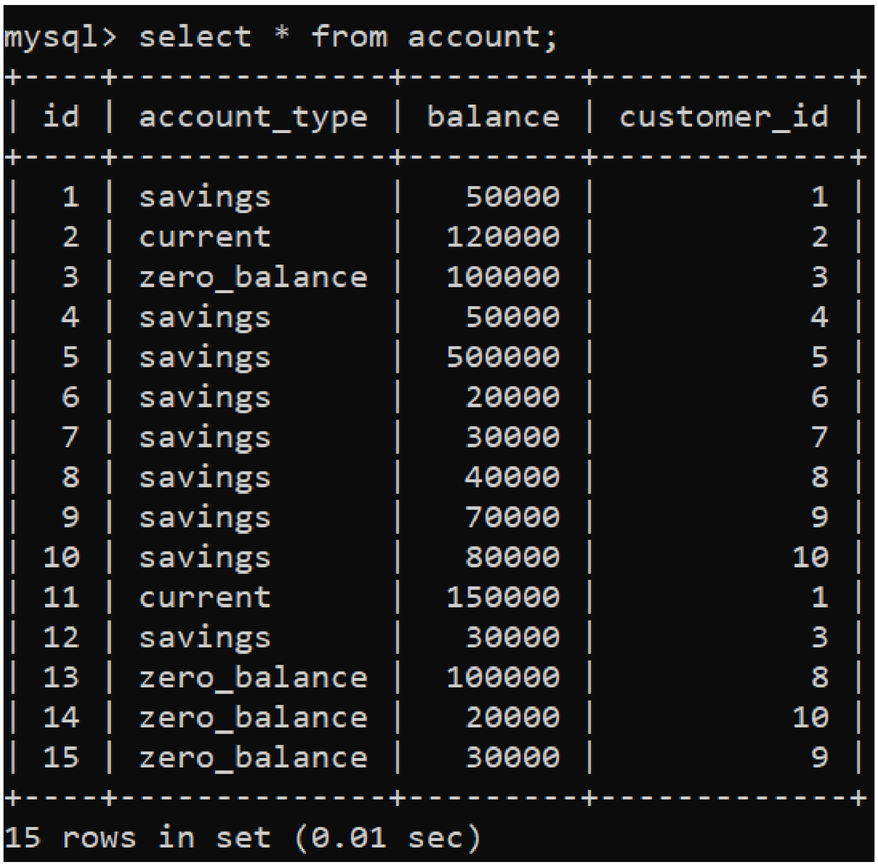
('savings',80000,10),

('current',150000,1),

('savings',30000,3),

('zero_balance',100000,8),

('zero_balance',20000,10),



```
mysql> select * from account;
+----+--------------+---------+-------------+
| id | account_type | balance | customer_id |
+----+--------------+---------+-------------+
|  1 | savings      |   50000 |           1 |
|  2 | current      |  120000 |           2 |
|  3 | zero_balance |  100000 |           3 |
|  4 | savings      |   50000 |           4 |
|  5 | savings      |  500000 |           5 |
|  6 | savings      |   20000 |           6 |
|  7 | savings      |   30000 |           7 |
|  8 | savings      |   40000 |           8 |
|  9 | savings      |   70000 |           9 |
| 10 | savings      |   80000 |          10 |
| 11 | current      |  150000 |           1 |
| 12 | savings      |   30000 |           3 |
| 13 | zero_balance |  100000 |           8 |
| 14 | zero_balance |   20000 |          10 |
| 15 | zero_balance |   30000 |           9 |
+----+--------------+---------+-------------+
15 rows in set (0.01 sec)
```

('zero_balance',30000,9);

--- transaction insertion

insert into transaction(transaction_type,amount,transaction_date,account_id)

values

('deposit',10000,'2024-02-01',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),

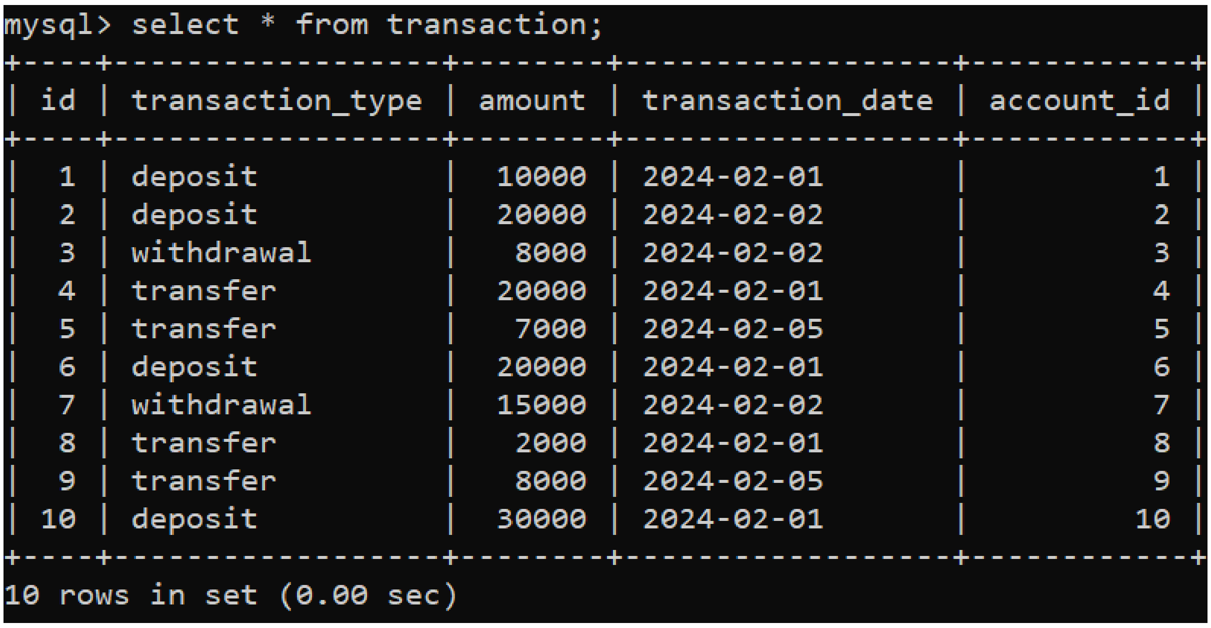
('deposit',20000,'2024-02-01',6),

('withdrawal',15000,'2024-02-02',7),

('transfer',2000,'2024-02-01',8),

('transfer',8000,'2024-02-05',9),

('deposit',30000,'2024-02-01',10);



```
mysql> select * from transaction;
+----+------------------+--------+------------------+------------+
| id | transaction_type | amount | transaction_date | account_id |
+----+------------------+--------+------------------+------------+
|  1 | deposit          |  10000 | 2024-02-01       |          1 |
|  2 | deposit          |  20000 | 2024-02-02       |          2 |
|  3 | withdrawal       |   8000 | 2024-02-02       |          3 |
|  4 | transfer         |  20000 | 2024-02-01       |          4 |
|  5 | transfer         |   7000 | 2024-02-05       |          5 |
|  6 | deposit          |  20000 | 2024-02-01       |          6 |
|  7 | withdrawal       |  15000 | 2024-02-02       |          7 |
|  8 | transfer         |   2000 | 2024-02-01       |          8 |
|  9 | transfer         |   8000 | 2024-02-05       |          9 |
| 10 | deposit          |  30000 | 2024-02-01       |         10 |
+----+------------------+--------+------------------+------------+
10 rows in set (0.00 sec)
```

Tasks 2: Select, Where, Between, AND, LIKE:

-- 1. Write a SQL query to retrieve the name, account type and email of all customers.

```
select concat(c.first_name," ",c.last_name) as name,a.account_type,c.email
from customer c,account a
where c.id=a.customer_id;
```

-- 2. Write a SQL query to list all transaction corresponding customer.

```
select concat(c.first_name," ",c.last_name) as name, t.* from
customer c, transaction t ,account a
where a.customer_id=c.id and a.id=t.account_id;
```

-- 3. Write a SQL query to increase the balance of a specific account by a certain amount.

```
update account set balance=balance+5000 where id=6;
```

-- 4. Write a SQL query to Combine first and last names of customers as a full_name.

```
select concat(first_name," ",last_name) as full_name from customer;
```

5. Write a SQL query to remove accounts with a balance of zero where the account type is savings.

```
insert into account(account_type,balance,customer_id) values ('savings',0,9);
delete from account where balance=0 and account_type='savings';
```

-- 6. Write a SQL query to Find customers living in a specific city.

```
select * from customer where address='delhi';
```

-- 7. Write a SQL query to Get the account balance for a specific account.

```
select id,balance from account where id=5;
```

-- 8. Write a SQL query to List all savings accounts with a balance greater than $100,000.

```
select * from account where balance>100000 and account_type='savings';
```

-- 9. Write a SQL query to Retrieve all transactions for a specific account.

```
select * from transaction where account_id=4;
```

10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.

```
select id,balance*2.5 as interest from account where account_type='savings';
```

11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.

```
select * from account where balance<20000;
```

-- 12. Write a SQL query to Find customers not living in a specific city.

```
select * from customer where address!='delhi';
```

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 customer_id,avg(customer_id) as average_account_balance from account
group by customer_id;
```

-- 2. Write a SQL query to Retrieve the top 5highest account balances.

```
select * from account
order by balance desc limit 0,5;
```

-- 3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

```
select * from transaction
where transaction_date='2024-02-01' and transaction_type='deposit';
```

```sql
-- 4. Write a SQL query to Find the Oldest and Newest Customers.
(select first_name,last_name,dob, 'oldest_customer' as status from customer order by dob asc
limit
0,1)
union all
(select first_name,last_name,dob,'newest_customer' as status from customer order by dob
desc
limit 0,1);

-- 5. Write a SQL query to Retrieve transaction details along with the account type.
select t.*,a.account_type from account a,transaction t
where a.id=t.account_id;

-- 6. Write a SQL query to Get a list of customers along with their account details.
select c.first_name,c.last_name,a.account_type,a.balance from account a,customer c
where c.id=a.customer_id;

7. Write a SQL query to Retrieve transaction details along with customer information for a
specific account.
select c.first_name,c.last_name,a.account_type,a.balance,t.transaction_type,
t.transaction_date, t.amount from account a,customer c,transaction t
where c.id=a.customer_id and a.id=t.account_id;

-- 8. Write a SQL query to Identify customers who have more than one account.
select c.first_name,c.last_name, count(c.id) as no_of_accounts
from customer c,account a
where c.id=a.customer_id
group by c.id
having count(c.id)>1;
```

9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

```
select (select sum(amount) from transaction where transaction_type='deposit') -
(select sum(amount) from transaction where transaction_type='withdrawal')
as difference_in_transaction;
```

10. Write a SQL query to Calculate the average daily balance for each account over a specified

period.

```
SELECT id, AVG(balance) AS avg_daily_balance
FROM account
GROUP BY id;

select a.id,avg(a.balance) as avg_balance
from account a join transaction t on t.account_id=a.id
where transaction_date between '2024-02-01' and '2024-02-12'
group by account_id;
```

-- 11. Calculate the total balance for each account type.

```
select account_type, sum(balance) as balance from account group by account_type;
```

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

```
select account_id ,count(account_id) as frequency from transaction
group by account_id
order by frequency desc;
```

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

```
select c.first_name,c.last_name,a.balance,a.account_type
from customer c ,account a
where c.id=a.customer_id
order by balance desc limit 0,1;
```

```
-- 14. Identify and list duplicate transactions based on transaction amount, date, and account.

select amount,transaction_date,account_id,count(*)

from transaction

group by amount,transaction_date,account_id

having count(*)>1;
```

Tasks 4: Subquery and its type

```
-- 1. Retrieve the customer(s) with the highest account balance.

select first_name,last_name

from customer where id =(select customer_id from account

 where balance=(select max(balance) from account));


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

select customer_id,avg(balance) as avg_balance

from account

group by customer_id

having count(customer_id)>1;


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

select account_id from transaction

where amount>(select avg(amount) from transaction);


-- 4. Identify customers who have no recorded transactions.

select first_name,last_name from

customer where id not in(select customer_id from account

where id in (select account_id from transaction));
```

-- 5. Calculate the total balance of accounts with no recorded transactions.

```
SELECT SUM(balance) AS total_balance

FROM account

WHERE id NOT IN (SELECT DISTINCT account_id FROM transaction);-- 6. Retrieve transactions for accounts with the lowest balance.

SELECT *

FROM transaction

WHERE account_id IN (SELECT id FROM account ORDER BY balance ASC );
```

-- 7. Identify customers who have accounts of multiple types.

```
SELECT customer_id

FROM account

GROUP BY customer_id

HAVING COUNT(DISTINCT account_type) > 1;
```

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

```
select account_type, count(*) AS account_count,count(*) / (

select count(*) from account) * 100 as percentage

from account

group by account_type;
```

-- 9. Retrieve all transactions for a customer with a given customer_id.

```
select transaction.* from transaction where account_id in (Select id from account where

customer_id = 2);
```

10. Calculate the total balance for each account type, including a subquery within the SELECT clause.

```
select account_type, sum(balance)

from account

group by account_type;
```