ASSIGNMENT 2 -BANKING SYSTEM

```
-use bankingSystem;
-insert into customer(first_name,last_name,dob) values
 ('harry','potter','2002-03-21'),
 ('ronald','weasley','2001-02-10'),
 ('hermione','granger','2002-11-15');
-select*from customer;
-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. Write a SQL query to retrieve the name, account type and email of all customers.
select distinct c.first_name,a.account_type from customer c , account a where c.id=a.customer_id;
/*
harry    savings
harry    current
ronald   current
hermione         zero_balance
hermione         savings
draco    zero_balance
*/
```

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

```
select c.first_name, t.transaction_type from customer c, transaction t, account a where
c.id=a.customer_id and a.id= t.account_id;
```

/*

first_name	transaction_type
harry	deposit
harry	withdrawal
harry	transfer
ronald	deposit
hermione	withdrawal
hermione	transfer

*/

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

```
select account_type,balance= balance+1000 as incresed_balance from account where
account_type='current';
```

-- 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;
```

/*

harry potter

ronald weasley

hermione granger

draco malfoy

*/

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

```
select id as removed_id from account where balance=0 and account_type='savings';
```

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

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

```
select id, balance from account where account_type='current';
```

/*

2	120000
4	150000

*/

-- 8. Write a SQL query to List all current accounts with a balance greater than $1,000.

```
select distinct account_type from account where balance>1000;
/*
        account_type
        savings
        current
        zero_balance
*/
-- 9. Write a SQL query to Retrieve all transactions for a specific account.
select a.account_type, t.transaction_type from account a, transaction t where a.id=t.account_id;
/*
savings deposit
savings withdrawal
current  deposit
zero_balance     withdrawal
current  transfer
savings transfer
*/
-- 10. Calculate the interest accrued on savings accounts based on a given interest rate
SELECT account_id, balance * interest_rate AS interest_accrued
FROM account
WHERE account_type = 'savings';

-- 11. Identify accounts where the balance is less than a specified overdraft limit
SELECT a.id AS account_id,
    a.account_type,
    a.balance
FROM account a
WHERE a.balance < 50000;
/*
5       savings 30000
6       zero_balance     40000
*/
```

```
-- 12. Find customers not living in a specific city
SELECT *
FROM customer
WHERE city != 'mumbai';
-- Task 3
-- Find the average account balance for all customers.
SELECT AVG(balance) AS average_balance
FROM account;
/*
81666.6667
*/
-- Write a SQL query to Retrieve the top 10 highest account balances.
SELECT *FROM account ORDER BY balance DESC LIMIT 10;
/*
4	current	150000	1
2	current	120000	2
3	zero_balance	100000	3
1	savings	50000	1
6	zero_balance	40000	4
5	savings	30000	3
*/

-- 3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.
SELECT SUM(t.amount) AS total_deposits
FROM transaction t
JOIN account a ON t.account_id = a.id
WHERE t.transaction_type = 'deposit'
AND t.transaction_date = '2024-02-02';
/*
20000
*/
-- 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;
    /*
    2000-05-06    2002-11-15
    */

--  5. Write a SQL query to Retrieve transaction details along with the account type.
SELECT
    t.id AS transaction_id,
    t.transaction_type,
    t.amount,
    t.transaction_date,
    a.account_type
FROM
    transaction t
JOIN
    account a ON t.account_id = a.id;
    /*
    1       deposit  10000   2024-02-01       savings
2           withdrawal       5000     2024-02-02       savings
3           deposit  20000   2024-02-02       current
4           withdrawal       8000     2024-02-02       zero_balance
5           transfer 20000   2024-02-01       current
6           transfer 7000    2024-02-05       savings
    */
-- 6. Write a SQL query to Get a list of customers along with their account details.
SELECT
    c.id AS customer_id,
    c.first_name,
```

```
    c.last_name,
    c.dob,
    a.id AS account_id,
    a.account_type,
    a.balance
FROM
    customer c
JOIN
    account a ON c.id = a.customer_id;
/*
1       harry   potter  2002-03-21      1       savings 50000
1       harry   potter  2002-03-21      4       current  150000
2       ronald  weasley 2001-02-10      2       current  120000
3       hermione        granger 2002-11-15      3       zero_balance    100000
3       hermione        granger 2002-11-15      5       savings 30000
4       draco   malfoy  2000-05-06      6       zero_balance    40000
*/
-- 7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.
SELECT
    t.id AS transaction_id,
    c.first_name,
    c.last_name
FROM
    transaction t
JOIN
    account a ON t.account_id = a.id
JOIN
    customer c ON a.customer_id = c.id
WHERE
    a.id = 2;
/*
3       ronald  weasley
```

```
*/
-- 8. Write a SQL query to Identify customers who have more than one account.
SELECT
    c.id AS customer_id,
    c.first_name,
    c.last_name
FROM
    customer c
JOIN
    account a ON c.id = a.customer_id
GROUP BY
    c.id
HAVING
    COUNT(a.id) > 1;
/*
1        harry    potter
3        hermione        granger
*/
-- 9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.
SELECT
    SUM(CASE WHEN t.transaction_type = 'deposit' THEN t.amount ELSE -t.amount END) AS net_transaction_amount
FROM
    transaction t;
    /*
    -10000
    */
-- 10. Write a SQL query to Calculate the average daily balance for each account over a specified period.
SELECT
    account_id,
    AVG(daily_balance) AS average_daily_balance
```

```
FROM
    (
        SELECT
            account_id,
            DATE(transaction_date) AS transaction_date,
            SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE -amount END) AS
daily_balance
        FROM
            transaction
        GROUP BY
            account_id,
            DATE(transaction_date)
    ) AS daily_balances
GROUP BY
    account_id;
    /*
    1       2500
2           20000
3           -8000
4           -20000
5           -7000
    */
-- 11.Calculate the total balance for each account type:
SELECT
    account_type,
    SUM(balance) AS total_balance
FROM
    account
GROUP BY
    account_type;
    /*
    current          270000
savings 80000
```

```
zero_balance     140000
   */
-- 12.Identify accounts with the highest number of transactions order by descending order:
SELECT
   account_id,
   COUNT(*) AS transaction_count
FROM
   transaction
GROUP BY
   account_id
ORDER BY
   COUNT(*) DESC;
   /*
   1      2
2         1
3         1
4         1
5         1
   */
-- 13.List customers with high aggregate account balances, along with their account types:
SELECT
   c.id AS customer_id,
   c.first_name,
   c.last_name,
   a.account_type,
   SUM(a.balance) AS aggregate_balance
FROM
   customer c
JOIN
   account a ON c.id = a.customer_id
GROUP BY
   c.id, a.account_type
```

```
HAVING
    SUM(a.balance) >  30000 ;
-- 14.Identify and list duplicate transactions based on transaction amount, date, and account:
SELECT
    transaction_type,
    amount,
    transaction_date,
    account_id,
    COUNT(*) AS duplicate_count
FROM
    transaction
GROUP BY
    transaction_type, amount, transaction_date, account_id
HAVING
    COUNT(*) > 1;
-- TASK 4
-- 1.Retrieve the customer(s) with the highest account balance:
SELECT
    id, first_name, last_name
FROM
    customer
WHERE
    id = (
        SELECT
            customer_id
        FROM
            account
        ORDER BY
            balance DESC
        LIMIT 1
    );
-- 2.Calculate the average account balance for customers who have more than one account
```

```
SELECT
    AVG(balance) AS average_balance
FROM
    (
        SELECT
            customer_id, COUNT(*) AS account_count
        FROM
            account
        GROUP BY
            customer_id
        HAVING
            COUNT(*) > 1
    ) AS multi_account_customers
JOIN
    account ON multi_account_customers.customer_id = account.customer_id;
    /*
    82500.0000
    */
-- 3.Retrieve accounts with transactions whose amounts exceed the average transaction amount.
SELECT
    a.id AS account_id,
    a.account_type,
    a.balance,
    t.amount AS transaction_amount
FROM
    account a
JOIN
    transaction t ON a.id = t.account_id
WHERE
    t.amount > (SELECT AVG(amount) FROM transaction);
    /*
    2       current  120000 20000
```

```
4          current  150000 20000
   */
-- 4.. Identify customers who have no recorded transactions.
SELECT
   id AS customer_id,
   first_name,
   last_name
FROM
   customer
WHERE
   id NOT IN (SELECT DISTINCT customer_id FROM transaction);

-- 5. Calculate the total balance of accounts with no recorded transactions.
SELECT
   SUM(balance) AS total_balance_no_transactions
FROM
   account
WHERE
   id NOT IN (SELECT DISTINCT account_id FROM transaction);
   /*
   40000
   */

-- 6. Retrieve transactions for accounts with the lowest balance.
SELECT
   t.id AS transaction_id,
   t.transaction_type,
   t.amount,
   t.transaction_date,
   a.id AS account_id,
   a.account_type,
   a.balance
```

```
FROM
    transaction t
JOIN
    account a ON t.account_id = a.id
WHERE
    a.balance = (SELECT MIN(balance) FROM account);
/*
6           transfer 7000     2024-02-05        5           savings 30000
*/
-- 7. Identify customers who have accounts of multiple types.
SELECT
    c.id AS customer_id,
    c.first_name,
    c.last_name
FROM
    customer c
JOIN
    account a ON c.id = a.customer_id
GROUP BY
    c.id
HAVING
    COUNT(DISTINCT account_type) > 1;
    /*
    1       harry     potter
3           hermione          granger
    */

-- 8. Calculate the percentage of each account type out of the total number of accounts.
SELECT
    account_type,
    COUNT(*) AS account_count,
    ROUND((COUNT(*) * 100.0) / (SELECT COUNT() FROM account), 2) AS percentage
```

```
FROM
    account
GROUP BY
    account_type;



-- 9. Retrieve all transactions for a customer with a given customer_id.
SELECT
    t.id AS transaction_id,
    t.transaction_type,
    t.amount,
    t.transaction_date,
    t.transaction_date,
    a.account_type,
    a.balance
FROM
    transaction t
JOIN
    account a ON t.account_id = a.id
WHERE
    a.customer_id = 2;
/*
3           deposit  20000   2024-02-02        2024-02-02        current  120000
*/
-- 10. Calculate the total balance for each account type, including a subquery within the SELECT clause.
SELECT
    account_type,
    (SELECT SUM(balance) FROM account WHERE account_type = a.account_type) AS total_balance
FROM
    account a
GROUP BY
```

```
    account_type;
    /*
    current          270000
savings 80000
zero_balance     140000
    */
```