TASK 1: Database Design (Normalisation):

1. Create the database named "HMBank"

QUERY: CREATE DATABASE HMBank;
USE HMBank;

```
CREATE DATABASE HMBank;
USE HMBank;
```

```
Messages
Commands completed successfully.

Completion time: 2023-12-11T10:25:59.3450747+05:30
```

2. Define the schema for the Customers, Accounts, and Transactions tables based on the provided schema.

QUERY:
CREATE TABLE Customers (
customer_id INT PRIMARY KEY,
full_name VARCHAR(100), DOB
DATE,
contact_info VARCHAR(255),
address VARCHAR(255)
);

```
CREATE TABLE Customers (customer_id INT PRIMARY KEY,
        full_name VARCHAR(100),
        DOB DATE,
        contact_info VARCHAR(255),
        address VARCHAR(255));
```

```
Messages
Commands completed successfully.

Completion time: 2023-12-11T10:29:41.3487665+05:30
```

CREATE TABLE Accounts (account_id INT PRIMARY KEY,
customer_id INT, full_name VARCHAR(100), account_type VARCHAR(20), balance DECIMAL(10, 2), DOB DATE,
contact_info VARCHAR(255), address VARCHAR(255),
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

```sql
CREATE TABLE Accounts (account_id INT PRIMARY KEY,
    customer_id INT,
    full_name VARCHAR(100),
    account_type VARCHAR(20),
    balance DECIMAL(10, 2),
    DOB DATE,
    contact_info VARCHAR(255),
    address VARCHAR(255),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id));
```

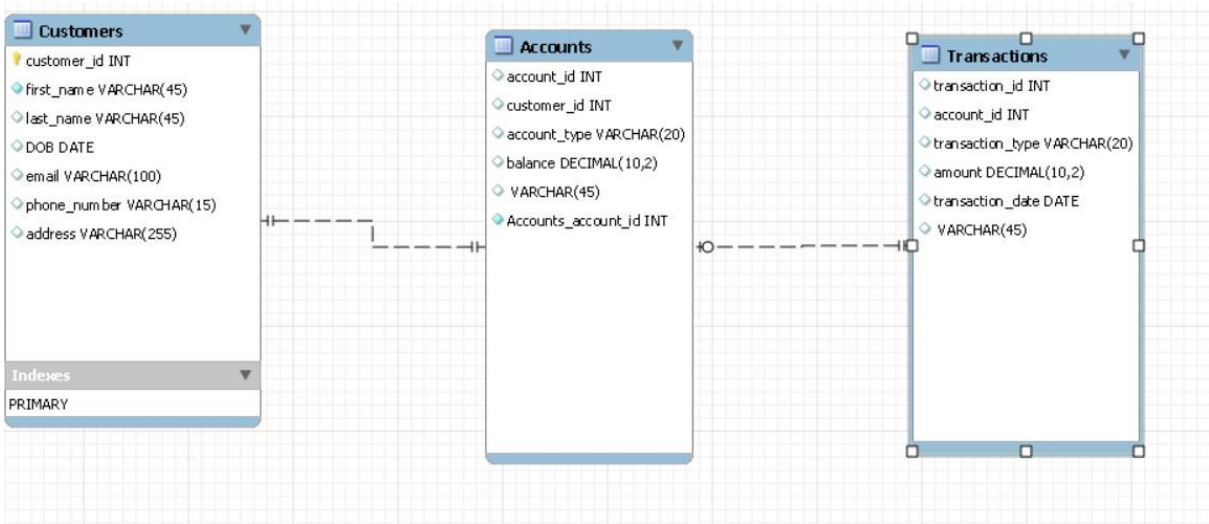
```
200 %
Messages
Commands completed successfully.

Completion time: 2023-12-11T10:30:15.7243409+05:30
```

CREATE TABLE Transactions (transaction_id INT PRIMARY KEY,
account_id INT, full_name VARCHAR(100), transaction_type VARCHAR(20), amount DECIMAL(10, 2), transaction_date DATE, account_type VARCHAR(20), balance DECIMAL(10, 2), DOB DATE,
contact_info VARCHAR(255),
address VARCHAR(255),
FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);

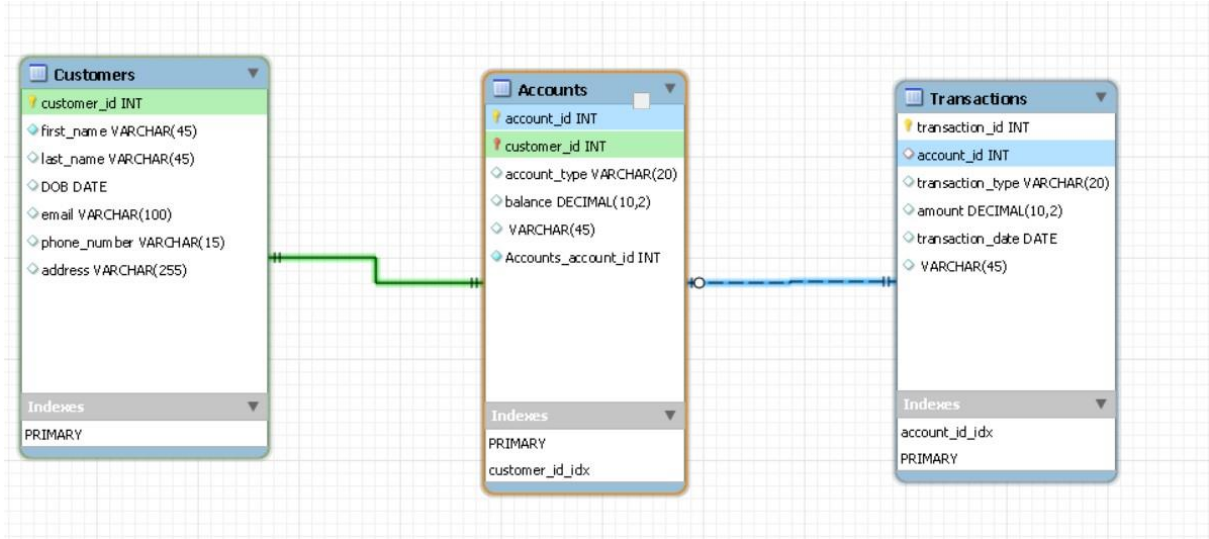
```sql
CREATE TABLE Transactions (transaction_id INT PRIMARY KEY,
    account_id INT,
    full_name VARCHAR(100),
    transaction_type VARCHAR(20),
    amount DECIMAL(10, 2),
    transaction_date DATE,
    account_type VARCHAR(20),
    balance DECIMAL(10, 2),
    DOB DATE,
    contact_info VARCHAR(255),
    address VARCHAR(255),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

```
200 %
Messages
Commands completed successfully.

Completion time: 2023-12-11T10:31:32.9841599+05:30

200 %
```

4. Create an ERD (Entity Relationship Diagram) for the database.



5. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



<u>Task: Data Definition Language (DDL):</u>

1. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

- **Customers**
- **Accounts**
- **Transactions**

Query-

```
-- Customers Table CREATE TABLE
Customers (    customer_id INT
PRIMARY KEY,     first_name
VARCHAR(50),     last_name
VARCHAR(50),     DOB DATE,
email VARCHAR(100),
phone_number VARCHAR(15),
   address VARCHAR(255)
);

-- Accounts Table CREATE TABLE
Accounts (     account_id INT
PRIMARY KEY,
   customer_id INT,     account_type
VARCHAR(20),     balance
DECIMAL(10, 2),
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

-- Transactions Table CREATE TABLE
Transactions (     transaction_id INT
PRIMARY KEY,
   account_id INT,     transaction_type
VARCHAR(20),     amount
DECIMAL(10, 2),     transaction_date
DATE,
   FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

Task 2: Data Manipulation Language (DML):

2. Write SQL queries for the following tasks:

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

```sql
-- Question 1
select customers.first_name,customers.last_name,accounts.account_type , customers.email from customers
inner join accounts on customers.customer_id=accounts.customer_id ;

-- Question 2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	account_type	email
John	Doe	savings	john.doe@example.com
Jane	Smith	current	jane.smith@example.com
Bob	Johnson	savings	bob.johnson@example.com
Alice	Williams	zero_balance	alice.williams@example.com
Charlie	Brown	current	charlie.brown@example.com

2.Write a SQL query to list all transaction corresponding customers.

```sql
SELECT
    Transactions.transaction_id,
    Customers.first_name,
    Customers.last_name,
    Transactions.transaction_type,
    Transactions.amount,
    Transactions.transaction_date
FROM
    Transactions
JOIN
    Accounts ON Transactions.account_id = Accounts.account_id
JOIN
    Customers ON Accounts.customer_id = Customers.customer_id;


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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transaction_id	first_name	last_name	transaction_type	amount	transaction_date
1	John	Doe	deposit	500.00	2023-01-01
2	Jane	Smith	withdrawal	200.00	2023-01-02
3	Bob	Johnson	transfer	1000.00	2023-01-03
4	Alice	Williams	deposit	300.00	2023-01-04
5	Charlie	Brown	withdrawal	150.00	2023-01-05

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

```sql
-- Question 3 Write a SQL query to increase the balance of a specific account by a certain amount.
UPDATE Accounts
SET balance = balance + 500.00
WHERE account_id = 1;

select * from accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_id	customer_id	account_type	balance
1	1	savings	2000.00
2	2	current	500.00
3	3	savings	2000.00
4	4	zero_balance	0.00
5	5	current	1000.00
6	6	savings	2500.00

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

```sql
-- Question 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
    Customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

full_name
John Doe
Jane Smith
Bob Johnson
Alice Williams
Charlie Brown
Eva Davis

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

```sql
-- Question 5 Write a SQL query to remove accounts with a balance of zero where the account type is saving
Delete  FROM accounts
WHERE balance = 0 AND account_type = 'savings';

select * from accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_id	customer_id	account_type	balance
1	1	savings	2000.00
2	2	current	500.00
3	3	savings	2000.00
4	4	current	0.00
5	5	current	1000.00
6	6	savings	2500.00

Result Grid

Form Editor

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

```
683  -- Question 6 Write a SQL query to Find customers living in a specific city.
684  SELECT * FROM Customers1
685  WHERE city = 'Mumbai';
686
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name	DOB	email	phone_number	address	city
1	John	Doe	1990-05-15	john.doe@example.com	1234567890	123 Main St	Mumbai
19	Rachel	Ward	1988-06-23	rachel.ward@example.com	3456789012	1616 Elm St	Mumbai
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

Query:

SELECT balance FROM Accounts WHERE account_id = 102
SELECT *FROM Accounts

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

SELECT balance FROM Accounts WHERE account_id = 102
SELECT *FROM Accounts
```

90 %

Results | Messages

	balance
1	1500.00

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

```sql
--Write a SQL query to List all current accounts with a balance greater than $1,0
SELECT account_id, account_type , balance,customer_id
FROM Accounts WHERE balance > 1000
```

90 %

Results | Messages

	account_id	account_type	balance	customer_id
1	101	savings	5000.00	1
2	102	current	1500.00	1
3	103	savings	8000.00	2
4	104	current	2500.00	2
5	105	savings	6000.00	3
6	106	current	2000.00	3
7	107	savings	7000.00	4
8	108	current	3000.00	5
9	109	savings	9000.00	5
10	110	savings	4000.00	6
11	111	current	5000.00	7
12	112	savings	12000.00	8
13	113	current	8000.00	9

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

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

SELECT
    Transactions.transaction_id,
    Transactions.account_id,
    Transactions.transaction_type,
    Transactions.amount,
    Transactions.transaction_date
FROM
    Transactions
JOIN
    Accounts ON Transactions.account_id = Accounts.account_id
WHERE
    Accounts.account_id = 105
```

90 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	1007	105	deposit	1500.00	2023-07-25

10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate 5 percent.

```sql
-- a SQL query to Calculate the interest accrued on savings accounts based on a

SELECT
    account_id,
    account_type,
    balance,
    (balance * 0.05) AS interest_accrued
FROM
    Accounts
WHERE
    account_type = 'savings';
```

90 %

Results | Messages

	account_id	account_type	balance	interest_accrued
1	101	savings	5000.00	250.0000
2	103	savings	8000.00	400.0000
3	105	savings	6000.00	300.0000
4	107	savings	7000.00	350.0000
5	109	savings	9000.00	450.0000
6	110	savings	4000.00	200.0000
7	112	savings	12000.00	600.0000

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

```sql
--Write a SQL query to Identify accounts where the balance is less than a spec
SELECT
    account_id,
    account_type,
    balance
FROM
    Accounts
WHERE
    balance < 5000;
```

Results | Messages

	account_id	account_type	balance
1	102	current	1500.00
2	104	current	2500.00
3	106	current	2000.00
4	108	current	3000.00
5	110	savings	4000.00

Task 3: Aggregate functions, GroupBy and Joins:

1. Write a SQL query to Find the average account balance for all customers.

```sql
--Write a SQL query to Find the average account balance for all customers.
SELECT
    AVG(balance) AS average_account_balance
FROM
    Accounts;
```

90 %

Results | Messages

	average_account_balance
1	5615.384615

2. Write a SQL query to Retrieve the top 10 highest account balances.

```sql
--  Write a SQL query to Retrieve the top 10 highest account balances
SELECT top 10
    account_id,
    account_type,
    balance
FROM
    Accounts
ORDER BY
    balance DESC ;
```

90 %

Results | Messages

	account_id	account_type	balance
1	112	savings	12000.00
2	109	savings	9000.00
3	103	savings	8000.00
4	113	current	8000.00
5	107	savings	7000.00
6	105	savings	6000.00
7	101	savings	5000.00
8	111	current	5000.00
9	110	savings	4000.00
10	108	current	3000.00

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

```sql
--Write a SQL query to Calculate Total Deposits for All Customers on a specific date.
SELECT
    t.transaction_date,
    SUM(t.amount) AS total_deposits
FROM
    Transactions t
JOIN
    Accounts a ON t.account_id = a.account_id
WHERE
    t.transaction_type = 'deposit'
    AND t.transaction_date = '2023-01-10'
GROUP BY
    t.transaction_date;

select * from Transactions
```

Results | Messages

	transaction_date	total_deposits
1	2023-01-10	1000.00

4. Write a SQL query to Find the Oldest and Newest Customers.

a. Find the oldest customer.

b. Find the newest customer.

```sql
--Write a SQL query to Find the Oldest and Newest Customers.
--Find the oldest customer
SELECT top 1 *
FROM customers
ORDER BY DOB ASC



--Find the newest customer

SELECT TOP 1 *
FROM customers
ORDER BY customer_id DESC;
```

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address
1	1	John	Doe	1980-05-15	john.doe@email.com	555-1234	123 Main St

	customer_id	first_name	last_name	DOB	email	phone_number	address
1	10	Daniel	Miller	1983-02-14	daniel.miller@email.com	555-1098	707 Cedar St

5.Write a SQL query to Retrieve transaction details along with the account type.
Query-

```sql
--Write a SQL query to Retrieve transaction details along with the account type
SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date,
    a.account_type
FROM
    Transactions t
JOIN
    Accounts a ON t.account_id = a.account_id;
```

99 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date	account_type
1	1001	101	deposit	1000.00	2023-01-10	savings
2	1002	101	withdrawal	500.00	2023-02-15	savings
3	1003	102	deposit	2000.00	2023-03-20	current
4	1004	102	withdrawal	100.00	2023-04-05	current
5	1005	103	deposit	3000.00	2023-05-12	savings
6	1006	104	withdrawal	800.00	2023-06-18	current
7	1007	105	deposit	1500.00	2023-07-25	savings
8	1008	106	withdrawal	200.00	2023-08-30	current
9	1009	107	deposit	2500.00	2023-09-15	savings
10	1010	108	withdrawal	1200.00	2023-10-20	current
11	1011	109	deposit	1800.00	2023-11-22	savings
12	1012	110	withdrawal	700.00	2023-12-28	savings
13	1013	111	deposit	3500.00	2023-01-02	current

6.Write a SQL query to Get a list of customers along with their account details.
7.Write a SQL query to Retrieve transaction details along with customer information for a

```sql
--Write a SQL query to Get a list of customers along with their account details
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    c.DOB,
    c.email,
    c.phone_number,
    c.address,
    a.account_id,
    a.account_type,
    a.balance
FROM
    Customers c
JOIN
    Accounts a ON c.customer_id = a.customer_id;
```

99 %

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address	account_id	account_type	balance
1	1	John	Doe	1980-05-15	john.doe@email.com	555-1234	123 Main St	101	savings	5000.00
2	1	John	Doe	1980-05-15	john.doe@email.com	555-1234	123 Main St	102	current	1500.00
3	2	Jane	Smith	1992-08-22	jane.smith@email.com	555-5678	456 Oak St	103	savings	8000.00
4	2	Jane	Smith	1992-08-22	jane.smith@email.com	555-5678	456 Oak St	104	current	2500.00
5	3	Alice	Johnson	1985-11-08	alice.johnson@email.com	555-9876	789 Pine St	105	savings	6000.00
6	3	Alice	Johnson	1985-11-08	alice.johnson@email.com	555-9876	789 Pine St	106	current	2000.00
7	4	Bob	Williams	1990-04-25	bob.williams@email.com	555-4321	101 Cedar St	107	savings	7000.00
8	5	Eva	Davis	1982-09-18	eva.davis@email.com	555-8765	202 Elm St	108	current	3000.00
9	5	Eva	Davis	1982-09-18	eva.davis@email.com	555-8765	202 Elm St	109	savings	9000.00
10	6	Michael	Lee	1995-06-30	michael.lee@email.com	555-3456	303 Maple St	110	savings	4000.00
11	7	Samantha	Jones	1988-12-12	samantha.jones@email.com	555-6543	404 Birch St	111	current	5000.00
12	8	David	Taylor	1987-03-05	david.taylor@email.com	555-7890	505 Oak St	112	savings	12000.00
13	9	Sophia	Brown	1993-07-28	sophia.brown@email.com	555-2109	606 Pine St	113	current	8000.00

specific account.

```sql
--Write a SQL query to Retrieve transaction details along with customer information for a specific account

SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date,
    c.customer_id,
    c.first_name,
    c.last_name,
    c.DOB,
    c.email,
    c.phone_number,
    c.address
FROM
    Transactions t
JOIN
    Accounts a ON t.account_id = a.account_id
JOIN
    Customers c ON a.customer_id = c.customer_id
WHERE
    t.account_id = 105
```

99 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date	customer_id	first_name	last_name	DOB	email	phone_number	address
1	1007	105	deposit	1500.00	2023-07-25	3	Alice	Johnson	1985-11-08	alice.johnson@email.com	555-9876	789 Pine St

8.Write a SQL query to Identify customers who have more than one account.

```sql
--Write a SQL query to Identify customers who have more than one account.
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    COUNT(a.account_id) AS num_accounts
FROM
    Customers c
JOIN
    Accounts a ON c.customer_id = a.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name
HAVING
    COUNT(a.account_id) > 1;
```

99 %

Results | Messages

	customer_id	first_name	last_name	num_accounts
1	1	John	Doe	2
2	2	Jane	Smith	2
3	3	Alice	Johnson	2
4	5	Eva	Davis	2

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

```sql
--Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals

SELECT
    account_id,
    SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE 0 END) AS total_deposits,
    SUM(CASE WHEN transaction_type = 'withdrawal' THEN amount ELSE 0 END) AS total_withdrawals,
    SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE -amount END) AS difference
FROM
    Transactions
GROUP BY
    account_id;
```

99 %

Results | Messages

	account_id	total_deposits	total_withdrawals	difference
1	101	1000.00	500.00	500.00
2	102	2000.00	100.00	1900.00
3	103	3000.00	0.00	3000.00
4	104	0.00	800.00	-800.00
5	105	1500.00	0.00	1500.00
6	106	0.00	200.00	-200.00
7	107	2500.00	0.00	2500.00
8	108	0.00	1200.00	-1200.00
9	109	1800.00	0.00	1800.00
10	110	0.00	700.00	-700.00
11	111	3500.00	0.00	3500.00

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

```sql
--Write a SQL query to Calculate the average daily balance for each account over a specified period.

SELECT
    account_id,
    AVG(balance) AS average_daily_balance
FROM
    Accounts
WHERE
    DATEDIFF(DAY, (SELECT MIN(transaction_date) FROM Transactions WHERE account_id = Accounts.account_id), GETDATE()) >= 0
GROUP BY
    account_id;
```

99 %

Results | Messages

	account_id	average_daily_balance
1	101	5000.000000
2	102	1500.000000
3	103	8000.000000
4	104	2500.000000
5	105	6000.000000
6	106	2000.000000
7	107	7000.000000
8	108	3000.000000
9	109	9000.000000
10	111	5000.000000

11. Calculate the total balance for each account type.

```sql
--Calculate the total balance for each account type.
    SELECT
    account_type,
    SUM(balance) AS total_balance
FROM
    Accounts
GROUP BY
    account_type;
```

99 %

Results | Messages

	account_type	total_balance
1	current	22000.00
2	savings	51000.00

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

```sql
--Identify accounts with the highest number of transactions ordered by descending

SELECT
    a.account_id,
    COUNT(t.transaction_id) AS transaction_count
FROM
    Accounts a
JOIN
    Transactions t ON a.account_id = t.account_id
GROUP BY
    a.account_id
ORDER BY
    transaction_count DESC;
```

99 %

Results | Messages

	account_id	transaction_count
1	101	2
2	102	2
3	103	1
4	104	1
5	105	1
6	106	1
7	107	1
8	108	1
9	109	1
10	110	1
11	111	1

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

```sql
--List customers with high aggregate account balances, along with their account types.
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    c.DOB,
    c.email,
    c.phone_number,
    c.address,
    a.account_type,
    SUM(a.balance) AS aggregate_balance
FROM
    Customers c
JOIN
    Accounts a ON c.customer_id = a.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name, c.DOB, c.email, c.phone_number, c.address, a.account_type
ORDER BY
    aggregate_balance DESC;
```

99 %

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address	account_type	aggregate_balance
1	8	David	Taylor	1987-03-05	david.taylor@email.com	555-7890	505 Oak St	savings	12000.00
2	5	Eva	Davis	1982-09-18	eva.davis@email.com	555-8765	202 Elm St	savings	9000.00
3	9	Sophia	Brown	1993-07-28	sophia.brown@email.com	555-2109	606 Pine St	current	8000.00
4	2	Jane	Smith	1992-08-22	jane.smith@email.com	555-5678	456 Oak St	savings	8000.00
5	4	Bob	Williams	1990-04-25	bob.williams@email.com	555-4321	101 Cedar St	savings	7000.00
6	3	Alice	Johnson	1985-11-08	alice.johnson@email.com	555-9876	789 Pine St	savings	6000.00
7	1	John	Doe	1980-05-15	john.doe@email.com	555-1234	123 Main St	savings	5000.00
8	7	Samantha	Jones	1988-12-12	samantha.jones@email.com	555-6543	404 Birch St	current	5000.00
9	6	Michael	Lee	1995-06-30	michael.lee@email.com	555-3456	303 Maple St	savings	4000.00
10	5	Eva	Davis	1982-09-18	eva.davis@email.com	555-8765	202 Elm St	current	3000.00
11	2	Jane	Smith	1992-08-22	jane.smith@email.com	555-5678	456 Oak St	current	2500.00
12	3	Alice	Johnson	1985-11-08	alice.johnson@email.com	555-9876	789 Pine St	current	2000.00
13	1	John	Doe	1980-05-15	john.doe@email.com	555-1234	123 Main St	current	1500.00

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

```sql
--Identify and list duplicate transactions based on transaction amount, date, and account
SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date
FROM
    Transactions t
JOIN (
    SELECT
        account_id,
        amount,
        transaction_date
    FROM
        Transactions
    GROUP BY
        account_id, amount, transaction_date
    HAVING
        COUNT(*) > 1
) AS duplicates ON t.account_id = duplicates.account_id
                AND t.amount = duplicates.amount
                AND t.transaction_date = duplicates.transaction_date
ORDER BY
    t.account_id, t.amount, t.transaction_date, t.transaction_id;
```

99 %

Results | Messages

transaction_id	account_id	transaction_type	amount	transaction_date

Task 4: Subquery

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

```sql
--Retrieve the customer(s) with the highest account balance.
WITH RankedCustomers AS (
    SELECT
        c.customer_id,
        c.first_name,
        c.last_name,
        c.DOB,
        c.email,
        c.phone_number,
        c.address,
        a.account_id,
        a.account_type,
        a.balance,
        RANK() OVER (ORDER BY a.balance DESC) AS balance_rank
    FROM
        Customers c
    JOIN
        Accounts a ON c.customer_id = a.customer_id
)
SELECT
    customer_id,
    first_name,
    last_name,
    DOB,
    email,
    phone_number,
    address,
    account_id,
    account_type,
    balance
FROM
    RankedCustomers
WHERE
    balance_rank = 1;
```

68 %

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address	account_id	account_type	balance
1	8	David	Taylor	1987-03-05	david.taylor@email.com	555-7890	505 Oak St	112	savings	12000.00

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

```sql
--Calculate the average account balance for customers who have more than one account

WITH CustomerAccountCounts AS (
    SELECT
        c.customer_id,
        COUNT(a.account_id) AS num_accounts
    FROM
        Customers c
    JOIN
        Accounts a ON c.customer_id = a.customer_id
    GROUP BY
        c.customer_id
    HAVING
        COUNT(a.account_id) > 1
)
SELECT
    AVG(a.balance) AS average_balance
FROM
    Accounts a
JOIN
    CustomerAccountCounts cac ON a.customer_id = cac.customer_id;
```

99 %

Results | Messages

	average_balance
1	4625.000000

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

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

SELECT a.account_id, a.customer_id, t.transaction_id, t.amount
FROM accounts a
JOIN transactions t ON a.account_id = t.account_id
WHERE t.amount > (SELECT AVG(amount) FROM transactions);
```

99 %

Results | Messages

	account_id	customer_id	transaction_id	amount
1	102	1	1003	2000.00
2	103	2	1005	3000.00
3	105	3	1007	1500.00
4	107	4	1009	2500.00
5	109	5	1011	1800.00
6	111	7	1013	3500.00

4. Identify customers who have no recorded transactions.

```sql
--Identify customers who have no recorded transactions.

SELECT customer_id, first_name, last_name, DOB, email, phone_number, address
FROM Customers
WHERE customer_id NOT IN (SELECT c.customer_id FROM Customers c
    LEFT JOIN accounts a ON c.customer_id = a.customer_id
    LEFT JOIN transactions t ON a.account_id = t.account_id
    WHERE t.transaction_id IS NOT NULL
);
```

99 %

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address
1	8	David	Taylor	1987-03-05	david.taylor@email.com	555-7890	505 Oak St
2	9	Sophia	Brown	1993-07-28	sophia.brown@email.com	555-2109	606 Pine St
3	10	Daniel	Miller	1983-02-14	daniel.miller@email.com	555-1098	707 Cedar St

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

```sql
--Calculate the total balance of accounts with no recorded transactions

    SELECT accounts.BALANCE , transactions.ACCOUNT_ID       FROM accounts,transactions
    where accounts.account_id = transactions.account_id       AND
    transactions.transaction_type is null
```

Results | Messages

BALANCE	ACCOUNT_ID

6.Retrieve transactions for accounts with the lowest balance

QUERY:

```sql
--Retrieve transactions for accounts with the lowest balance

WITH LowestBalanceAccounts AS (
    SELECT TOP 1
        account_id,
        balance
    FROM
        Accounts
    ORDER BY
        balance ASC
)
SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date
FROM
    Transactions t
JOIN
    LowestBalanceAccounts lba ON t.account_id = lba.account_id;
```

99 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	1003	102	deposit	2000.00	2023-03-20
2	1004	102	withdrawal	100.00	2023-04-05

7.Identify customers who have accounts of multiple types

QUERY:

```sql
--Identify customers who have accounts of multiple types

WITH LowestBalanceAccounts AS (
    SELECT TOP 1
        account_id,
        balance
    FROM
        Accounts
    ORDER BY
        balance ASC
)
SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date
FROM
    Transactions t
JOIN
    LowestBalanceAccounts lba ON t.account_id = lba.account_id;
```

99 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	1003	102	deposit	2000.00	2023-03-20
2	1004	102	withdrawal	100.00	2023-04-05

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

```sql
SELECT
    account_type,
    COUNT(*) AS num_accounts,
    CAST(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER () AS DECIMAL(5, 2)) AS percentage
FROM
    Accounts
GROUP BY
    account_type;
```

99 %

Results | Messages

	account_type	num_accounts	percentage
1	current	6	46.15
2	savings	7	53.85

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

```sql
--Retrieve all transactions for a customer with a given customer_id.
SELECT
    t.transaction_id,
    t.account_id,
    t.transaction_type,
    t.amount,
    t.transaction_date
FROM
    Transactions t
JOIN
    Accounts a ON t.account_id = a.account_id
WHERE
    a.customer_id = 1;
```

99 %

Results | Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	1001	101	deposit	1000.00	2023-01-10
2	1002	101	withdrawal	500.00	2023-02-15
3	1003	102	deposit	2000.00	2023-03-20
4	1004	102	withdrawal	100.00	2023-04-05

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

```sql
--Calculate the total balance for each account type, including a subquery within the SELECT clause.
SELECT
    account_type,
    (SELECT SUM(balance) FROM Accounts a2 WHERE a2.account_type = a.account_type) AS total_balance
FROM
    Accounts a
GROUP BY
    account_type;
```

99 %

Results | Messages

	account_type	total_balance
1	current	22000.00
2	savings	51000.00