Assignment

Banking System - HM_Bank

By ~ **ANSHAY KHARE** *(anshaykhare27@gmail.com)*

Tasks 1: Database Design:

Q1. Create the database named "HMBank"

```
create database HMBank;
use HMBank;
```

Messages

```
Commands completed successfully.

Completion time: 2024-09-23T10:55:04.5112377+05:30
```

Q2. Define the schema for the Customers, Accounts, and Transactions tables based on the provided schema.

Customers Table Schema=>

```
create table Customers(
customer_id INT IDENTITY PRIMARY KEY,
first_name VARCHAR(20) NOT NULL,
last_name VARCHAR(20) NOT NULL,
DOB VARCHAR(10) NOT NULL,
email VARCHAR(30) NOT NULL,
phone_number VARCHAR(10) NOT NULL,
address VARCHAR(30) NOT NULL);
```

Accounts Table Schema=>

```
create table Accounts(
account_id INT IDENTITY PRIMARY KEY,
customer_id INT FOREIGN KEY
REFERENCES Customers(customer_id),
account_type VARCHAR(15) NOT NULL,
balance INT);
```

Transactions Table Schema=>

```
create table Transactions(
transaction_id INT IDENTITY PRIMARY KEY,
account_id INT FOREIGN KEY
REFERENCES Accounts(account_id),
transaction_type VARCHAR(15) NOT NULL,
amount INT,
transaction_date VARCHAR(10) NOT NULL);
```

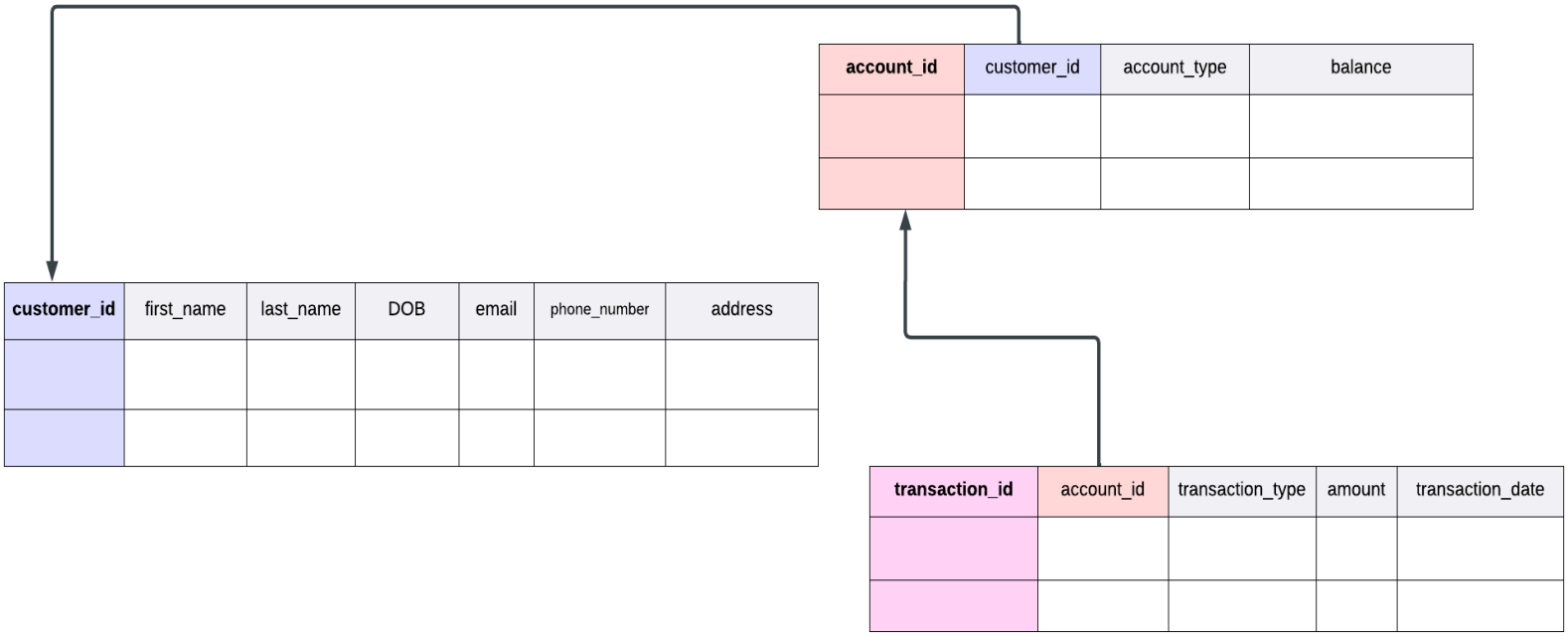
Messages

```
Commands completed successfully.

Completion time: 2024-09-23T11:25:26.2909717+05:30
```

Q3. Create an ERD (Entity Relationship Diagram) for the database.

Entity Relationship Diagram for tables named :
Customer , Accounts and Transaction



Q4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Primary Key and Foreign Key constraints were already created while creating the schema of the following tables.

Q5. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

- Customers
- Accounts
- Transactions

Already did the same while creating the schema of tables.

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

Q1) Insert at least 10 sample records into each of the following tables.

- Customers
- Accounts
- Transactions

Inserting Records into Customers Table=>

```
INSERT INTO Customers(first_name, last_name, DOB, email, phone_number, address)
VALUES
    ('John', 'Doe', '1990-01-01', 'johndoe@example.com', '1234567890', '123 Main St'),
    ('Jane', 'Smith', '1995-02-02', 'janesmith@example.com', '9876543210', '456 Main St'),
    ('Alice', 'Johnson', '1985-03-03', 'alicejohnson@example.com', '5678901234', '789 Main St'),
    ('Bob', 'Williams', '1970-04-04', 'bobwilliams@example.com', '4567890123', '101 Main St'),
    ('Charlie', 'Brown', '1965-05-05', 'charliebrown@example.com', '3456789012', '102 Main St'),
    ('David', 'Lee', '1950-06-06', 'davidlee@example.com', '2345678901', '103 Main St'),
    ('Emily', 'Wilson', '1945-07-07', 'emilywilson@example.com', '1234567890', '104 Main St'),
    ('Frank', 'Taylor', '1940-08-08', 'franktaylor@example.com', '9876543210', '105 Main St'),
    ('Grace', 'Miller', '1935-09-09', 'gracemiller@example.com', '5678901234', '106 Main St'),
    ('Henry', 'Baker', '1930-10-10', 'henrybaker@example.com', '4567890123', '107 Main St');
```

Inserting Records into Accounts Table=>

```
INSERT INTO Accounts (customer_id, account_type, balance)
VALUES
    (1, 'Savings', 1000),
    (1, 'Current', 500),
    (2, 'Savings', 2000),
    (3, 'Current', 1500),
    (4, 'Savings', 3000),
    (4, 'Current', 2500),
    (6, 'Savings', 4000),
    (7, 'Current', 3500),
    (7, 'Savings', 5000),
    (9, 'Current', 4500);
```

Inserting Records into Transactions Table=>

```
INSERT INTO Transactions
(account_id, transaction_type, amount, transaction_date)
VALUES
    (1, 'Deposit', 500.00, '2024-09-01'),
    (1, 'Withdrawal', 200.00, '2024-09-02'),
    (2, 'Deposit', 1000.00, '2024-09-03'),
    (2, 'Withdrawal', 500.00, '2024-09-04'),
    (2, 'Deposit', 1500.00, '2024-09-05'),
    (3, 'Withdrawal', 750.00, '2024-09-06'),
    (4, 'Deposit', 2000.00, '2024-09-07'),
    (7, 'Withdrawal', 1000.00, '2024-09-08'),
    (9, 'Deposit', 2500.00, '2024-09-09'),
    (9, 'Withdrawal', 1250.00, '2024-09-10');
```

Messages

```
(10 rows affected)

(10 rows affected)

(10 rows affected)

Completion time: 2024-09-23T11:48:03.3122410+05:30
```

Q2) Write SQL queries for the following tasks:

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

```
select Customers.first_name, Accounts.account_type,
       Customers.email
From Customers left join Accounts
ON Customers.customer_id = Accounts.customer_id;
```

Results | Messages

	first_name	account_type	email
1	John	Savings	johndoe@example.com
2	John	Current	johndoe@example.com
3	Jane	Savings	janesmith@example.com
4	Alice	Current	alicejohnson@example.com
5	Bob	Savings	bobwilliams@example.com
6	Bob	Current	bobwilliams@example.com
7	Charlie	NULL	charliebrown@example.com
8	David	Savings	davidlee@example.com
9	Emily	Current	emilywilson@example.com
10	Emily	Savings	emilywilson@example.com
11	Frank	NULL	franktaylor@example.com
12	Grace	Current	gracemiller@example.com
13	Henry	NULL	henrybaker@example.com

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

```
select Customers.first_name, Transactions.transaction_type,
       Transactions.amount, Transactions.transaction_date
From Transactions left join Accounts
ON Transactions.account_id = Accounts.account_id
left join Customers
ON Customers.customer_id = Accounts.customer_id;
```

Results Messages

	first_name	transaction_type	amount	transaction_date
1	John	Deposit	500	2024-09-01
2	John	Withdrawal	200	2024-09-02
3	John	Deposit	1000	2024-09-03
4	John	Withdrawal	500	2024-09-04
5	John	Deposit	1500	2024-09-05
6	Jane	Withdrawal	750	2024-09-06
7	Alice	Deposit	2000	2024-09-07
8	David	Withdrawal	1000	2024-09-08
9	Emily	Deposit	2500	2024-09-09
10	Emily	Withdrawal	1250	2024-09-10

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

```
update Accounts
set balance = balance + 2000
where account_id = 4;
```

Messages

```
(1 row affected)

Completion time: 2024-09-23T12:02:42.3290625+05:30
```

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

```
select
CONCAT(Customers.first_name, ' ', Customers.last_name) As full_name
from Customers
```

Results Messages

	full_name
1	John Doe
2	Jane Smith
3	Alice Johnson
4	Bob Williams
5	Charlie Brown
6	David Lee
7	Emily Wilson
8	Frank Taylor
9	Grace Miller
10	Henry Baker

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

```
delete from Accounts
where balance=0
and account_type='Saving';
```

Messages

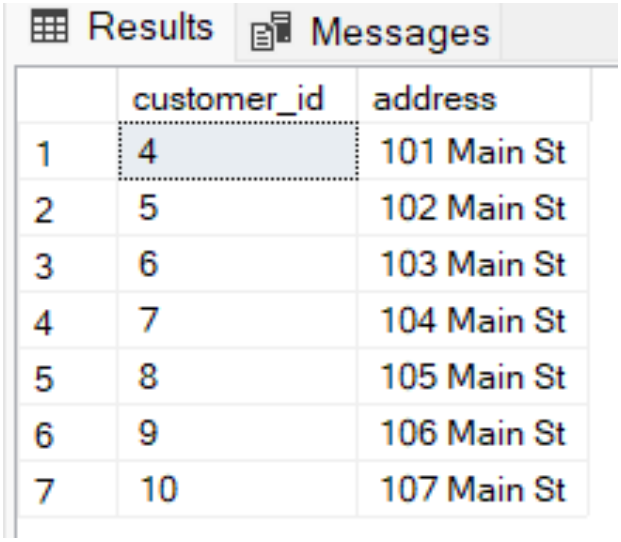
```
(0 rows affected)

Completion time: 2024-09-23T12:20:26.0143787+05:30
```

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

```
--In our data entries, assuming customers with address (100 Main st. - 109 Main st.) are residing in same city

select customer_id,address From Customers
where address like '10_ Main St';
```

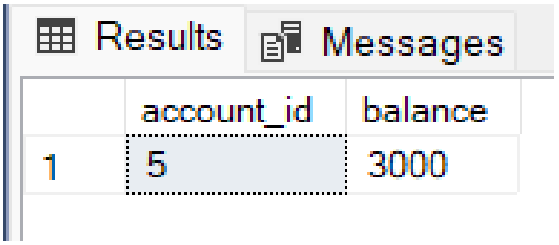


Results Messages

	customer_id	address
1	4	101 Main St
2	5	102 Main St
3	6	103 Main St
4	7	104 Main St
5	8	105 Main St
6	9	106 Main St
7	10	107 Main St

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

```
select account_id, balance From Accounts
where account_id = 5;
```



Results Messages

	account_id	balance
1	5	3000

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

```
select account_id,balance From Accounts
where balance>1000 and account_type='Current';
```

Results Messages

	account_id	balance
1	4	5500
2	6	2500
3	8	3500
4	10	4500

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

```
select * from Transactions
where account_id= 2;
```

Results Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	3	2	Deposit	1000	2024-09-03
2	4	2	Withdrawal	500	2024-09-04
3	5	2	Deposit	1500	2024-09-05

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

```
declare @rate int = 5;
select *,((balance * @rate)/100) AS Interest
from Accounts
```

Results Messages

	account_id	customer_id	account_type	balance	Interest
1	1	1	Savings	1000	50
2	2	1	Current	500	25
3	3	2	Savings	2000	100
4	4	3	Current	5500	275
5	5	4	Savings	3000	150
6	6	4	Current	2500	125
7	7	6	Savings	4000	200
8	8	7	Current	3500	175
9	9	7	Savings	5000	250
10	10	9	Current	4500	225

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

```
declare @overdraft_limit int = 2600;
select * From Accounts
where balance<@overdraft_limit;
```

Results Messages

	account_id	customer_id	account_type	balance
1	1	1	Savings	1000
2	2	1	Current	500
3	3	2	Savings	2000
4	6	4	Current	2500

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

--In our data entries, assuming customers with address (100 Main st. - 109 Main st.) are residing in same city

```
select * From Customers
where address not like '10_ Main St';
```

Results | Messages

	customer_id	first_name	last_name	DOB	email	phone_number	address
1	1	John	Doe	1990-01-01	johndoe@example.com	1234567890	123 Main St
2	2	Jane	Smith	1995-02-02	janesmith@example.com	9876543210	456 Main St
3	3	Alice	Johnson	1985-03-03	alicejohnson@example.com	5678901234	789 Main St

Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

Q1. Write a SQL query to Find the average account balance for all customers.

```
select customer_id ,
       AVG(balance) AS "Average Balance"
from Accounts
group by customer_id;
```

Results Messages

	customer_id	Average Balance
1	1	750
2	2	2000
3	3	5500
4	4	2750
5	6	4000
6	7	4250
7	9	4500

Q2. Write a SQL query to Retrieve the top 10 highest account balances.

```
select top 10 customer_id, balance from Accounts
order by balance desc;
```

Results Messages

	customer_id	balance
1	3	5500
2	7	5000
3	9	4500
4	6	4000
5	7	3500
6	4	3000
7	4	2500
8	2	2000
9	1	1000
10	1	500

Q3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

```
declare @transactionDate Varchar(10) = '2024-09-09';
select SUM(amount) AS TOTAL_DEPOSIT from Transactions
where transaction_type = 'Deposit'
and transaction_date = @transactionDate;
```

Results Messages

	TOTAL_DEPOSIT
1	2500

Q4. Write a SQL query to Find the Oldest and Newest Customers.

```sql
--Query to find oldest customer
select top 1 customer_id,DOB from Customers
order by DOB;
--Query to find newest/youngest customer
select top 1 customer_id,DOB from Customers
order by DOB desc;
```

Results Messages

	customer_id	DOB
1	10	1930-10-10

	customer_id	DOB
1	2	1995-02-02

Q5. Write a SQL query to Retrieve transaction details along with the account type.

```sql
select t.*,a.account_type from Transactions t
left join Accounts a
On t.account_id = a.account_id;
```

Results Messages

	transaction_id	account_id	transaction_type	amount	transaction_date	account_type
1	1	1	Deposit	500	2024-09-01	Savings
2	2	1	Withdrawal	200	2024-09-02	Savings
3	3	2	Deposit	1000	2024-09-03	Current
4	4	2	Withdrawal	500	2024-09-04	Current
5	5	2	Deposit	1500	2024-09-05	Current
6	6	3	Withdrawal	750	2024-09-06	Savings
7	7	4	Deposit	2000	2024-09-07	Current
8	8	7	Withdrawal	1000	2024-09-08	Savings
9	9	9	Deposit	2500	2024-09-09	Savings
10	10	9	Withdrawal	1250	2024-09-10	Savings

Q6. Write a SQL query to Get a list of customers along with their account details.

```sql
select c.first_name, c.last_name, a.* from Accounts a
left join Customers c
on a.customer_id = c.customer_id;
```

Results Messages

	first_name	last_name	account_id	customer_id	account_type	balance
1	John	Doe	1	1	Savings	1000
2	John	Doe	2	1	Current	500
3	Jane	Smith	3	2	Savings	2000
4	Alice	Johnson	4	3	Current	5500
5	Bob	Williams	5	4	Savings	3000
6	Bob	Williams	6	4	Current	2500
7	David	Lee	7	6	Savings	4000
8	Emily	Wilson	8	7	Current	3500
9	Emily	Wilson	9	7	Savings	5000
10	Grace	Miller	10	9	Current	4500

Q7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.

```sql
select t.*,c.customer_id,c.first_name from Transactions t
left join Accounts a on t.account_id = a.account_id
left join Customers c on a.customer_id = c.customer_id
where a.account_id = 2;
```

Results Messages

	transaction_id	account_id	transaction_type	amount	transaction_date	customer_id	first_name
1	3	2	Deposit	1000	2024-09-03	1	John
2	4	2	Withdrawal	500	2024-09-04	1	John
3	5	2	Deposit	1500	2024-09-05	1	John

Q8. Write a SQL query to Identify customers who have more than one account.

```sql
SELECT c.customer_id, c.first_name, c.last_name,
       COUNT(c.customer_id) AS account_count
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(c.customer_id) > 1;
```

Results Messages

	customer_id	first_name	last_name	account_count
1	1	John	Doe	2
2	4	Bob	Williams	2
3	7	Emily	Wilson	2

Q9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

```sql
select (
       (select sum(amount) from Transactions
        where transaction_type = 'Deposit') -
       (select sum(amount) from Transactions
        where transaction_type = 'Withdrawal')
) AS total_difference
```

Results Messages

	total_difference
1	3800

Q10. Write a SQL query to Calculate the average daily balance for each account over a specified period.

```
declare @days int = 20;
select *, (balance/@days) AS avg_daily_balance
From Accounts;
```

Results Messages

	account_id	customer_id	account_type	balance	avg_daily_balance
1	1	1	Savings	1000	50
2	2	1	Current	500	25
3	3	2	Savings	2000	100
4	4	3	Current	5500	275
5	5	4	Savings	3000	150
6	6	4	Current	2500	125
7	7	6	Savings	4000	200
8	8	7	Current	3500	175
9	9	7	Savings	5000	250
10	10	9	Current	4500	225

Q11. Calculate the total balance for each account type.

```
select account_type, sum(balance) as total_balance
from Accounts
group by account_type
```

Results Messages

	account_type	total_balance
1	Current	16500
2	Savings	15000

Q12. Identify accounts with the highest number of transactions order by descending order.

```
select account_id, COUNT(account_id) as 'No. of transactions'
from Transactions
group by account_id
order by COUNT(account_id) desc;
```

Results Messages

	account_id	No. of transactions
1	2	3
2	1	2
3	9	2
4	3	1
5	4	1
6	7	1

Q13. List customers with high aggregate account balances, along with their account types.

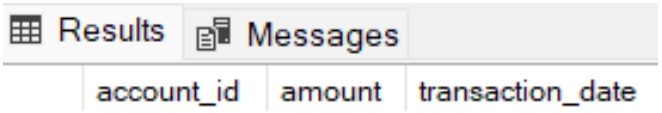
```
declare @highBalance int = 2000;
select customer_id,balance,account_type from Accounts
where balance>@highBalance;
```

Results Messages

	customer_id	balance	account_type
1	3	5500	Current
2	4	3000	Savings
3	4	2500	Current
4	6	4000	Savings
5	7	3500	Current
6	7	5000	Savings
7	9	4500	Current

Q14. Identify and list duplicate transactions based on transaction amount, date, and account.

```
select distinct account_id, amount, transaction_date
from Transactions
group by account_id, amount, transaction_date
Having COUNT(*)>1;
```



(We get an empty record because we don't have any duplicate entries in our data)

Tasks 4: Subquery and its type:

Q1. Retrieve the customer(s) with the highest account balance.

```
select * from Accounts
where balance = (select MAX(balance) from Accounts);
```

Results Messages

	account_id	customer_id	account_type	balance
1	4	3	Current	5500

Q2. Calculate the average account balance for customers who have more than one account.

```
select customer_id, AVG(balance) from Accounts
group by customer_id
having COUNT(customer_id)>1;
```

Results Messages

	customer_id	(No column name)
1	1	750
2	4	2750
3	7	4250

Q3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.

```sql
select * from Transactions
where amount> (select AVG(amount) from Transactions)
```

Results Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	5	2	Deposit	1500	2024-09-05
2	7	4	Deposit	2000	2024-09-07
3	9	9	Deposit	2500	2024-09-09
4	10	9	Withdrawal	1250	2024-09-10

Q4. Identify customers who have no recorded transactions.

```sql
select a.customer_id from Accounts a left join Transactions t
on a.account_id = t.account_id
group by t.account_id,a.customer_id
having COUNT(t.account_id) = 0;
```

Results Messages

	customer_id
1	4
2	7
3	9

Q5. Calculate the total balance of accounts with no recorded transactions.

```sql
select a.customer_id,a.balance from Accounts a left join Transactions t
on a.account_id = t.account_id
group by t.account_id,a.customer_id,a.balance
having COUNT(t.account_id) = 0;
```

Results Messages

	customer_id	balance
1	4	2500
2	4	3000
3	7	3500
4	9	4500

Q6. Retrieve transactions for accounts with the lowest balance.

```sql
select t.* from Transactions t join Accounts a
on t.account_id = a.account_id
where a.balance = (select MIN(balance) from Accounts)
```

Results Messages

	transaction_id	account_id	transaction_type	amount	transaction_date
1	3	2	Deposit	1000	2024-09-03
2	4	2	Withdrawal	500	2024-09-04
3	5	2	Deposit	1500	2024-09-05

Q7. Identify customers who have accounts of multiple types.

```sql
select customer_id from Accounts
group by customer_id
having COUNT(customer_id)>1
```

Results Messag

	customer_id
1	1
2	4
3	7

Q8. Calculate the percentage of each account type out of the total number of accounts.

```
select account_type,
       (((COUNT(account_type)*100)/(select COUNT(account_id)
                                    from Accounts)))
       AS percentage from Accounts
group by account_type
```

Results Messages

	account_type	percentage
1	Current	50
2	Savings	50

Q9. Retrieve all transactions for a customer with a given customer_id.

```
declare @customerID int = 1;
select c.customer_id,t.* from Transactions t join Accounts a
on a.account_id = t.account_id
join Customers c on c.customer_id = a.customer_id
where c.customer_id= @customerID;
```

Results Messages

	customer_id	transaction_id	account_id	transaction_type	amount	transaction_date
1	1	1	1	Deposit	500	2024-09-01
2	1	2	1	Withdrawal	200	2024-09-02
3	1	3	2	Deposit	1000	2024-09-03
4	1	4	2	Withdrawal	500	2024-09-04
5	1	5	2	Deposit	1500	2024-09-05

Q10. Calculate the total balance for each account type, including a subquery within the SELECT clause.

```
select account_type,SUM(balance) from Accounts
group by account_type
```

Results Messages

	account_type	(No column name)
1	Current	16500
2	Savings	15000

----------X----------