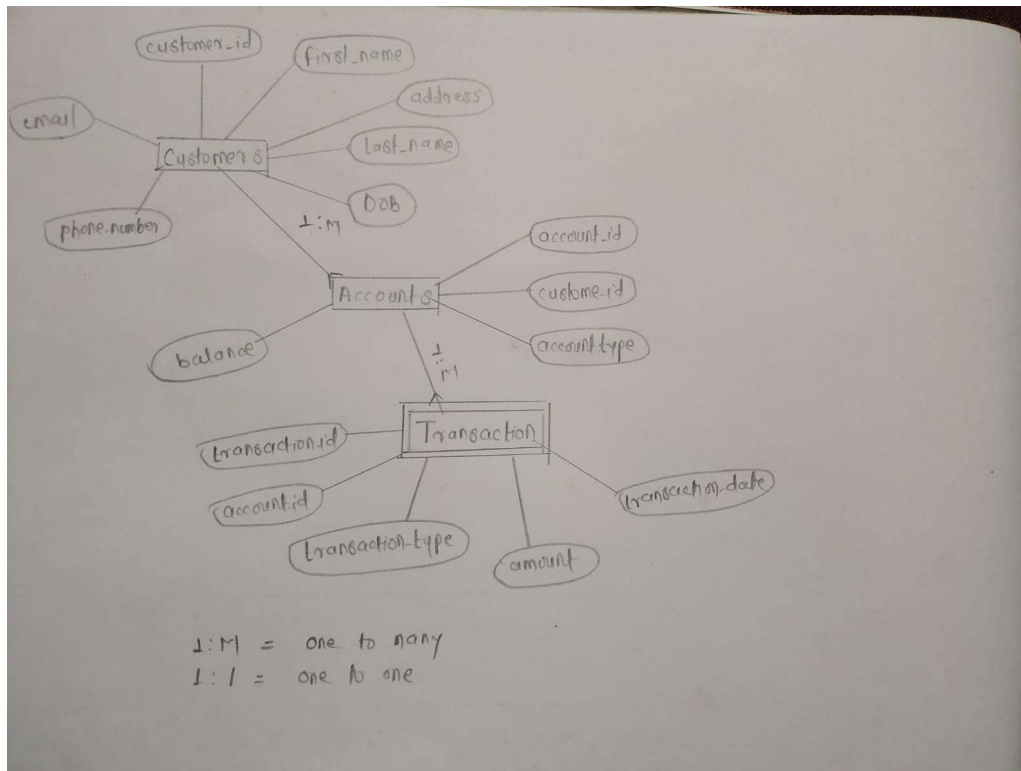Krushnakumar Homraj Patle

Tasks 1: Database Design:

1. Create the database named "HMBank".

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

4. Create an ERD

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

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

• Customers

• Accounts

• Transactions

Krushnakumar Homraj Patle

32 •   DESC Customers;

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

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(50)	NO		NULL	
last_name	varchar(50)	NO		NULL	
DOB	date	YES		NULL	
email	varchar(100)	NO	UNI	NULL	
phone_number	varchar(20)	YES		NULL	
address	varchar(200)	YES		NULL	

33 •   DESC Accounts;

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

Field	Type	Null	Key	Default	Extra
account_id	int	NO	PRI	NULL	auto_increment
customer_id	int	NO	MUL	NULL	
account_type	enum('savings','current','zero_balance')	NO		NULL	
balance	decimal(10,2)	NO		0.00	

34 •   DESC Transactions;

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

Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	auto_increment
account_id	int	NO	MUL	NULL	
transaction_type	enum('deposit','withdrawal','transfer')	NO		NULL	
amount	decimal(10,2)	NO		NULL	
transaction_date	date	NO		NULL	

```
 5  ● ⊖  CREATE TABLE Customers (
 6       |      customer_id INT PRIMARY KEY AUTO_INCREMENT,
 7       |      first_name VARCHAR(50) NOT NULL,
 8       |      last_name VARCHAR(50) NOT NULL,
 9       |      DOB DATE,
10       |      email VARCHAR(100) NOT NULL UNIQUE,
11       |      phone_number VARCHAR(20),
12       |      address VARCHAR(200)
13       └  );
14
15  ● ⊖  CREATE TABLE Accounts (
16       |      account_id INT PRIMARY KEY AUTO_INCREMENT,
17       |      customer_id INT NOT NULL,
18       |      account_type ENUM('savings', 'current', 'zero_balance') NOT NULL,
19       |      balance DECIMAL(10,2) NOT NULL DEFAULT 0,
20       |      FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
21       └  );
22
23  ● ⊖  CREATE TABLE Transactions (
24       |      transaction_id INT PRIMARY KEY AUTO_INCREMENT,
25       |      account_id INT NOT NULL,
26       |      transaction_type ENUM('deposit', 'withdrawal', 'transfer') NOT NULL,
27       |      amount DECIMAL(10,2) NOT NULL,
28       |      transaction_date DATE NOT NULL,
29       |      FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
30       └  );
```

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

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

- Customers
- Accounts
- Transactions

```
36 •    INSERT INTO Customers (first_name, last_name, DOB, email, phone_number, address)
37      VALUES ('John', 'Doe', '1980-01-01', 'john.doe@example.com', '+1234567890', '1 Main Street'),
38             ('Jane', 'Doe', '1985-02-02', 'jane.doe@example.com', '+9876543210', '2 Elm Street'),
39             ('Alice', 'Smith', '1990-03-03', 'alice.smith@example.com', '+0987654321', '3 Oak Street'),
40             ('Bob', 'Johnson', '1995-04-04', 'bob.johnson@example.com', '+1230987654', '4 Maple Street'),
41             ('Mary', 'Williams', '2000-05-05', 'mary.williams@example.com', '+5432109876', '5 Pine Street'),
42             ('David', 'Brown', '2005-06-06', 'david.brown@example.com', '+9876540987', '6 Cedar Street'),
43             ('Jessica', 'Miller', '2010-07-07', 'jessica.miller@example.com', '+0987098765', '7 Ash Street'),
44             ('Thomas', 'Garcia', '2015-08-08', 'thomas.garcia@example.com', '+1239876540', '8 Elm Street'),
45             ('Sarah', 'Jones', '2020-09-09', 'sarah.jones@example.com', '+9870987098', '9 Oak Street'),
46             ('Michael', 'Anderson', '2023-10-10', 'michael.anderson@example.com', '+0987654321', '10 Maple Street');
47
48 •    SELECT * FROM Customers;
```

customer_id	first_name	last_name	DOB	email	phone_number	address
1	John	Doe	1980-01-01	john.doe@example.com	+1234567890	1 Main Street
2	Jane	Doe	1985-02-02	jane.doe@example.com	+9876543210	2 Elm Street
3	Alice	Smith	1990-03-03	alice.smith@example.com	+0987654321	3 Oak Street
4	Bob	Johnson	1995-04-04	bob.johnson@example.com	+1230987654	4 Maple Street
5	Mary	Williams	2000-05-05	mary.williams@example.com	+5432109876	5 Pine Street
6	David	Brown	2005-06-06	david.brown@example.com	+9876540987	6 Cedar Street
7	Jessica	Miller	2010-07-07	jessica.miller@example.com	+0987098765	7 Ash Street
8	Thomas	Garcia	2015-08-08	thomas.garcia@example.com	+1239876540	8 Elm Street
9	Sarah	Jones	2020-09-09	sarah.jones@example.com	+9870987098	9 Oak Street
10	Michael	Anderson	2023-10-10	michael.anderson@example.com	+0987654321	10 Maple Street
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Krushnakumar Homraj Patle

```sql
50 •    INSERT INTO Accounts (customer_id, account_type, balance)
51      VALUES
52      (1, 'savings', 1000.00),
53      (2, 'current', 500.00),
54      (3, 'savings', 2000.00),
55      (4, 'current', 750.00),
56      (5, 'zero_balance', 0.00),
57      (6, 'savings', 3500.00),
58      (7, 'current', 1200.00),
59      (8, 'savings', 4800.00),
60      (9, 'current', 2350.00),
61      (10, 'zero_balance', 0.00);
62 •    SELECT * FROM Accounts;
63
```

Result Grid | Filter Rows: | Edit: | Export

account_id	customer_id	account_type	balance
1	1	savings	1000.00
2	2	current	500.00
3	3	savings	2000.00
4	4	current	750.00
5	5	zero_balance	0.00
6	6	savings	3600.00
7	7	current	1200.00
8	8	savings	4800.00
9	9	current	2350.00
10	10	zero_balance	0.00
NULL	NULL	NULL	NULL

```
64 ●     INSERT INTO Transactions (account_id, transaction_type, amount, transaction_date)
65       VALUES
66       (1, 'deposit', 100.00, '2023-10-26'),
67       (2, 'withdrawal', 50.00, '2023-10-27'),
68       (3, 'transfer', 25.00, '2023-10-28'),
69       (4, 'deposit', 75.00, '2023-10-29'),
70       (5, 'withdrawal', 20.00, '2023-10-30'),
71       (6, 'transfer', 15.00, '2023-10-31'),
72       (7, 'deposit', 125.00, '2023-11-01'),
73       (8, 'withdrawal', 35.00, '2023-11-02'),
74       (9, 'transfer', 40.00, '2023-11-03'),
75       (10, 'deposit', 150.00, '2023-11-04');
76
77 ●     SELECT * FROM Transactions;
78
```

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

transaction_id	account_id	transaction_type	amount	transaction_date
1	1	deposit	100.00	2023-10-26
2	2	withdrawal	50.00	2023-10-27
3	3	transfer	25.00	2023-10-28
4	4	deposit	75.00	2023-10-29
5	5	withdrawal	20.00	2023-10-30
6	6	transfer	15.00	2023-10-31
7	7	deposit	125.00	2023-11-01
8	8	withdrawal	35.00	2023-11-02
9	9	transfer	40.00	2023-11-03
10	10	deposit	150.00	2023-11-04
NULL	NULL	NULL	NULL	NULL

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

```
79 ●   SELECT c.first_name, c.last_name, c.email, a.account_type
80     FROM Customers c
81     INNER JOIN Accounts a ON c.customer_id = a.customer_id;
82
```

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

first_name	last_name	email	account_type
John	Doe	john.doe@example.com	savings
Jane	Doe	jane.doe@example.com	current
Alice	Smith	alice.smith@example.com	savings
Bob	Johnson	bob.johnson@example.com	current
Mary	Williams	mary.williams@example.com	zero_balance
David	Brown	david.brown@example.com	savings
Jessica	Miller	jessica.miller@example.com	current
Thomas	Garcia	thomas.garcia@example.com	savings
Sarah	Jones	sarah.jones@example.com	current
Michael	Anderson	michael.anderson@example.com	zero_balance

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

```
83 •   SELECT t.transaction_type, t.amount, t.transaction_date, c.first_name, c.last_name
84     FROM Transactions t
85     INNER JOIN Accounts a ON t.account_id = a.account_id
86     INNER JOIN Customers c ON a.customer_id = c.customer_id;
```

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

transaction_type	amount	transaction_date	first_name	last_name
deposit	100.00	2023-10-26	John	Doe
withdrawal	50.00	2023-10-27	Jane	Doe
transfer	25.00	2023-10-28	Alice	Smith
deposit	75.00	2023-10-29	Bob	Johnson
withdrawal	20.00	2023-10-30	Mary	Williams
transfer	15.00	2023-10-31	David	Brown
deposit	125.00	2023-11-01	Jessica	Miller
withdrawal	35.00	2023-11-02	Thomas	Garcia
transfer	40.00	2023-11-03	Sarah	Jones
deposit	150.00	2023-11-04	Michael	Anderson

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

```
88 •   UPDATE Accounts
89     SET balance = balance + 100.00
90     WHERE account_id = 6;
91
```

5. Write a SQL query to Combine first and last names of customers as a full name.

```
92 •   SELECT CONCAT(c.first_name, ' ', c.last_name) AS full_name
93     FROM Customers c;
```

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

full_name
John Doe
Jane Doe
Alice Smith
Bob Johnson
Mary Williams
David Brown
Jessica Miller
Thomas Garcia
Sarah Jones
Michael Anderson

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

```
DELETE FROM Accounts
WHERE account_type = 'savings' AND balance = 0;
```

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

```
98 •   SELECT *
99     FROM Customers
100    WHERE address LIKE '%Maple%';
```

	customer_id	first_name	last_name	DOB	email	phone_number	address
▶	4	Bob	Johnson	1995-04-04	bob.johnson@example.com	+1230987654	4 Maple Street
	10	Michael	Anderson	2023-10-10	michael.anderson@example.com	+0987654321	10 Maple Street
●	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
102 •   SELECT balance
103     FROM Accounts
104     WHERE account_id = 4;
```

	balance
▶	750.00
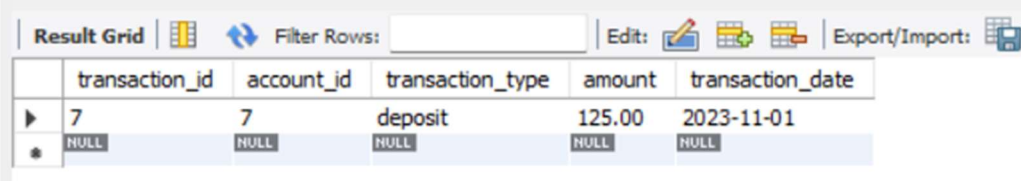
9. Write a SQL query to List all current accounts with a balance greater than $1,000.

```
106 •   SELECT *
107     FROM Accounts
108     WHERE account_type = 'current' AND balance > 1000.00;
109
```

	account_id	customer_id	account_type	balance
▶	7	7	current	1200.00
	9	9	current	2350.00
●	NULL	NULL	NULL	NULL

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

```
110 •   SELECT *
111     FROM Transactions
112     WHERE account_id = 7;
```

	transaction_id	account_id	transaction_type	amount	transaction_date
▶	7	7	deposit	125.00	2023-11-01
*	NULL	NULL	NULL	NULL	NULL

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

```
114 •   SELECT account_id, balance * interest_rate AS interest
115     FROM Accounts
116     WHERE account_type = 'savings';
```

transaction_id	account_id	transaction_type	amount	transaction_date

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

```
118 •   SELECT *
119     FROM Accounts
120     WHERE balance < 0;
121
```

	account_id	customer_id	account_type	balance
*	NULL	NULL	NULL	NULL

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

```
122 •   SELECT *
123     FROM Customers
124     WHERE address NOT LIKE '%Maple%';
```

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

	customer_id	first_name	last_name	DOB	email	phone_number	address
▶	1	John	Doe	1980-01-01	john.doe@example.com	+1234567890	1 Main Street
	2	Jane	Doe	1985-02-02	jane.doe@example.com	+9876543210	2 Elm Street
	3	Alice	Smith	1990-03-03	alice.smith@example.com	+0987654321	3 Oak Street
	5	Mary	Williams	2000-05-05	mary.williams@example.com	+5432109876	5 Pine Street
	6	David	Brown	2005-06-06	david.brown@example.com	+9876540987	6 Cedar Street
	7	Jessica	Miller	2010-07-07	jessica.miller@example.com	+0987098765	7 Ash Street
	8	Thomas	Garcia	2015-08-08	thomas.garcia@example.com	+1239876540	8 Elm Street
	9	Sarah	Jones	2020-09-09	sarah.jones@example.com	+9870987098	9 Oak Street
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

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

```
126 •   SELECT AVG(balance) AS average_balance
127     FROM Accounts;
```

Result Grid	Filter Rows:		Export:	Wrap Cell Content:

	average_balance
▶	1620.000000

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

```
129 •    SELECT *
130      FROM Accounts
131      ORDER BY balance DESC
132      LIMIT 10;
```

Result Grid | Filter Rows: | Edit:

account_id	customer_id	account_type	balance
8	8	savings	4800.00
6	6	savings	3600.00
9	9	current	2350.00
3	3	savings	2000.00
7	7	current	1200.00
1	1	savings	1000.00
4	4	current	750.00
2	2	current	500.00
5	5	zero_balance	0.00
10	10	zero_balance	0.00
NULL	NULL	NULL	NULL

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

```
134 •    SELECT DATE(transaction_date) AS transaction_date, SUM(amount) AS total_deposits
135      FROM Transactions
136      WHERE transaction_type = 'deposit'
137      GROUP BY DATE(transaction_date);
138
```

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

transaction_date	total_deposits
2023-10-26	100.00
2023-10-29	75.00
2023-11-01	125.00
2023-11-04	150.00

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

```
139 •    SELECT *
140      FROM Customers
141      ORDER BY DOB ASC, DOB DESC
142      LIMIT 1, 1;
```

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

customer_id	first_name	last_name	DOB	email	phone_number	address
2	Jane	Doe	1985-02-02	jane.doe@example.com	+9876543210	2 Elm Street
NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
144 •   SELECT t.*, a.account_type
145     FROM Transactions t
146     INNER JOIN Accounts a ON t.account_id = a.account_id;
1 47
```

	transaction_id	account_id	transaction_type	amount	transaction_date	account_type
▶	1	1	deposit	100.00	2023-10-26	savings
	2	2	withdrawal	50.00	2023-10-27	current
	3	3	transfer	25.00	2023-10-28	savings
	4	4	deposit	75.00	2023-10-29	current
	5	5	withdrawal	20.00	2023-10-30	zero_balance
	6	6	transfer	15.00	2023-10-31	savings
	7	7	deposit	125.00	2023-11-01	current
	8	8	withdrawal	35.00	2023-11-02	savings
	9	9	transfer	40.00	2023-11-03	current
	10	10	deposit	150.00	2023-11-04	zero_balance

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

```
148 •   SELECT c.*, a.*
149     FROM Customers c
150     INNER JOIN Accounts a ON c.customer_id = a.customer_id;
```

	customer_id	first_name	last_name	DOB	email	phone_number	address	account_id	customer_id	account_type	balance
▶	1	John	Doe	1980-01-01	john.doe@example.com	+1234567890	1 Main Street	1	1	savings	1000.00
	2	Jane	Doe	1985-02-02	jane.doe@example.com	+9876543210	2 Elm Street	2	2	current	500.00
	3	Alice	Smith	1990-03-03	alice.smith@example.com	+0987654321	3 Oak Street	3	3	savings	2000.00
	4	Bob	Johnson	1995-04-04	bob.johnson@example.com	+1230987654	4 Maple Street	4	4	current	750.00
	5	Mary	Williams	2000-05-05	mary.williams@example.com	+5432109876	5 Pine Street	5	5	zero_balance	0.00
	6	David	Brown	2005-06-06	david.brown@example.com	+9876540987	6 Cedar Street	6	6	savings	3600.00
	7	Jessica	Miller	2010-07-07	jessica.miller@example.com	+0987098765	7 Ash Street	7	7	current	1200.00
	8	Thomas	Garcia	2015-08-08	thomas.garcia@example.com	+1239876540	8 Elm Street	8	8	savings	4800.00
	9	Sarah	Jones	2020-09-09	sarah.jones@example.com	+9870987098	9 Oak Street	9	9	current	2350.00
	10	Michael	Anderson	2023-10-10	michael.anderson@example.com	+0987654321	10 Maple Street	10	10	zero_balance	0.00

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

```
210 •   SELECT t.*, c.first_name, c.last_name
211     FROM Transactions t
212     INNER JOIN Accounts a ON t.account_id = a.account_id
213     INNER JOIN Customers c ON a.customer_id = c.customer_id;
```

	transaction_id	account_id	transaction_type	amount	transaction_date	first_name	last_name
▶	1	1	deposit	100.00	2023-10-26	John	Doe
	2	2	withdrawal	50.00	2023-10-27	Jane	Doe
	3	3	transfer	25.00	2023-10-28	Alice	Smith
	4	4	deposit	75.00	2023-10-29	Bob	Johnson
	5	5	withdrawal	20.00	2023-10-30	Mary	Williams
	6	6	transfer	15.00	2023-10-31	David	Brown
	7	7	deposit	125.00	2023-11-01	Jessica	Miller
	8	8	withdrawal	35.00	2023-11-02	Thomas	Garcia
	9	9	transfer	40.00	2023-11-03	Sarah	Jones
	10	10	deposit	150.00	2023-11-04	Michael	Anderson

Krushnakumar Homraj Patle

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

```
157 •    SELECT c.*
158      FROM Customers c
159      INNER JOIN Accounts a ON c.customer_id = a.customer_id
160      GROUP BY c.customer_id
161      HAVING COUNT(a.account_id) > 1;
```

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

customer_id	first_name	last_name	DOB	email	phone_number	address

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

```
163 •  SELECT SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE 0 END) AS total_deposits,
164          SUM(CASE WHEN transaction_type = 'withdrawal' THEN amount ELSE 0 END) AS total_withdrawals
165    FROM Transactions;
```

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

total_deposits	total_withdrawals
450.00	105.00

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

```
167 •  SELECT account_id, DATE_FORMAT(transaction_date, '%Y-%m-%d') AS transaction_date, AVG(balance) AS average_daily_balance
168    FROM (
169        SELECT account_id, transaction_date, MAX(balance) AS balance
170        FROM Transactions
171        GROUP BY account_id, DATE(transaction_date)
172    ) AS daily_balances
173    GROUP BY account_id, transaction_date
174    HAVING DATE(transaction_date) BETWEEN '2023-10-01' AND '2023-10-31';
```

11. Calculate the total balance for each account type.

```
176 •    SELECT account_type, SUM(balance) AS total_balance
177      FROM Accounts
178      GROUP BY account_type;
```

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

account_type	total_balance
savings	11400.00
current	4800.00
zero_balance	0.00

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

```
180 •    SELECT account_id, COUNT(*) AS transaction_count
181      FROM Transactions
182      GROUP BY account_id
183      ORDER BY transaction_count DESC
184      LIMIT 1;
```

account_id	transaction_count
1	1

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

```
186 •    SELECT c.*, a.account_type, a.balance
187      FROM Customers c
188      INNER JOIN Accounts a ON c.customer_id = a.customer_id
189      WHERE a.balance > (SELECT AVG(balance) FROM Accounts)
190      ORDER BY c.last_name ASC;
191
```

customer_id	first_name	last_name	DOB	email	phone_number	address	account_type	balance
6	David	Brown	2005-06-06	david.brown@example.com	+9876540987	6 Cedar Street	savings	3600.00
8	Thomas	Garcia	2015-08-08	thomas.garcia@example.com	+1239876540	8 Elm Street	savings	4800.00
9	Sarah	Jones	2020-09-09	sarah.jones@example.com	+9870987098	9 Oak Street	current	2350.00
3	Alice	Smith	1990-03-03	alice.smith@example.com	+0987654321	3 Oak Street	savings	2000.00

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

```
192 •    SELECT *
193      FROM Transactions
194      WHERE (account_id, transaction_type, amount, transaction_date) IN (
195          SELECT account_id, transaction_type, amount, transaction_date
196          FROM Transactions
197          GROUP BY account_id, transaction_type, amount, transaction_date
198          HAVING COUNT(*) > 1
199      );
200
```

transaction_id	account_id	transaction_type	amount	transaction_date
NULL	NULL	NULL	NULL	NULL

Tasks 4: Subquery and its type:

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

```
 3 •  SELECT c.*, a.balance
 4    FROM Customers c
 5    INNER JOIN Accounts a ON c.customer_id = a.customer_id
 6  ⊖ WHERE a.balance = (
 7        SELECT MAX(balance)
 8        FROM Accounts
 9    );
10
```

customer_id	first_name	last_name	DOB	email	phone_number	address	balance
8	Thomas	Garcia	2015-08-08	thomas.garcia@example.com	+1239876540	8 Elm Street	4800.00

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

```
11 •  SELECT AVG(a.balance) AS average_balance
12    FROM Customers c
13    INNER JOIN Accounts a ON c.customer_id = a.customer_id
14    GROUP BY c.customer_id
15    HAVING COUNT(a.account_id) > 1;
16    |
```

average_balance

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

```
17 •  SELECT a.*, t.*
18    FROM Accounts a
19    INNER JOIN Transactions t ON a.account_id = t.account_id
20  ⊖ WHERE t.amount > (
21        SELECT AVG(amount)
22        FROM Transactions
23    );
24
```

account_id	customer_id	account_type	balance	transaction_id	account_id	transaction_type	amount	transaction_date
1	1	savings	1000.00	1	1	deposit	100.00	2023-10-26
4	4	current	750.00	4	4	deposit	75.00	2023-10-29
7	7	current	1200.00	7	7	deposit	125.00	2023-11-01
10	10	zero_balance	0.00	10	10	deposit	150.00	2023-11-04

4. Identify customers who have no recorded transactions.

```
27 •  SELECT *
28    FROM Customers c
29    WHERE NOT EXISTS (
30        SELECT *
31        FROM Transactions t
32        WHERE c.customer_id = t.account_id
33    );
```

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

customer_id	first_name	last_name	DOB	email	phone_number	address
NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
35 •  SELECT SUM(balance) AS total_balance
36    FROM Accounts a
37    WHERE NOT EXISTS (
38        SELECT *
39        FROM Transactions t
40        WHERE a.account_id = t.account_id
41    );
42
```

Result Grid	Filter Rows:		Export:	Wrap Cell Content:

total_balance
NULL

6. Retrieve transactions for accounts with the lowest balance.

```
43 •  SELECT t.*
44    FROM Transactions t
45    INNER JOIN Accounts a ON t.account_id = a.account_id
46    WHERE a.balance = (
47        SELECT MIN(balance)
48        FROM Accounts
49    );
50
51
```

transaction_id	account_id	transaction_type	amount	transaction_date
5	5	withdrawal	20.00	2023-10-30
10	10	deposit	150.00	2023-11-04

7. Identify customers who have accounts of multiple types.

```
51 •  SELECT c.*
52    FROM Customers c
53    INNER JOIN Accounts a ON c.customer_id = a.customer_id
54    GROUP BY c.customer_id
55    HAVING COUNT(DISTINCT a.account_type) > 1;
56
57
```

customer_id	first_name	last_name	DOB	email	phone_number	address

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

```
57 •  SELECT account_type, COUNT(*) AS total_accounts,
58        (COUNT(*) / (SELECT COUNT(*) FROM Accounts)) * 100 AS percentage
59    FROM Accounts
60    GROUP BY account_type;
61
62
```

account_type	total_accounts	percentage
savings	4	40.0000
current	4	40.0000
zero_balance	2	20.0000

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

```
62 •  SELECT *
63     FROM Transactions
64   ⊖ WHERE account_id IN (
65         SELECT account_id
66         FROM Accounts
67         WHERE customer_id = 4
68     );
```

transaction_id	account_id	transaction_type	amount	transaction_date
4	4	deposit	75.00	2023-10-29
NULL	NULL	NULL	NULL	NULL

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

```
70 •  SELECT account_type, SUM(balance) AS total_balance
71     FROM Accounts a
72   ⊖ WHERE a.account_type IN (
73         SELECT DISTINCT account_type
74         FROM Accounts
75     )
76     GROUP BY account_type;
77
```

account_type	total_balance
savings	11400.00
current	4800.00
zero_balance	0.00