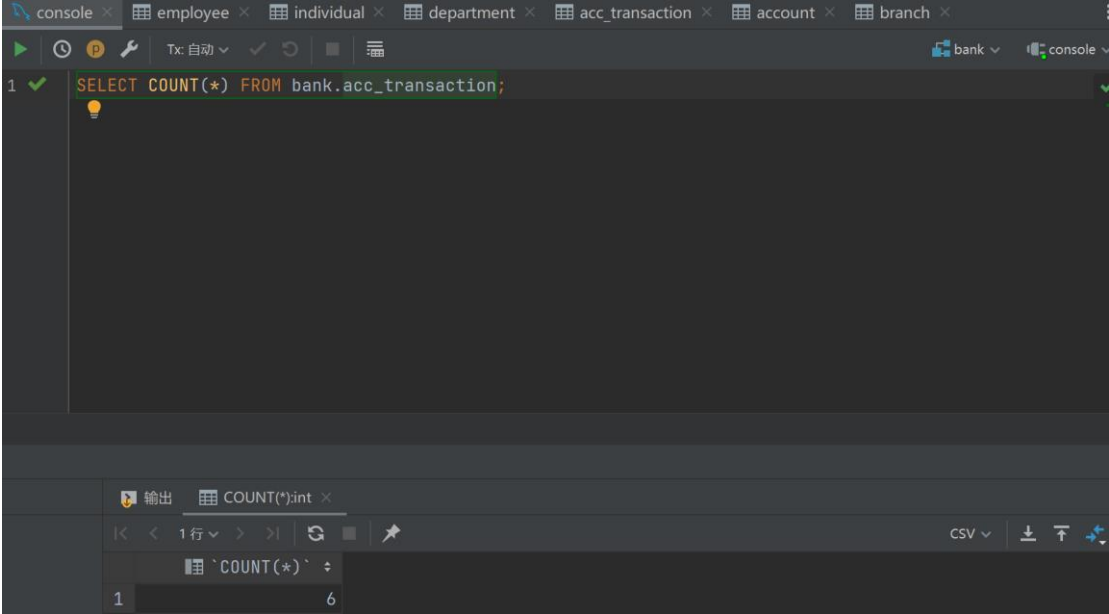


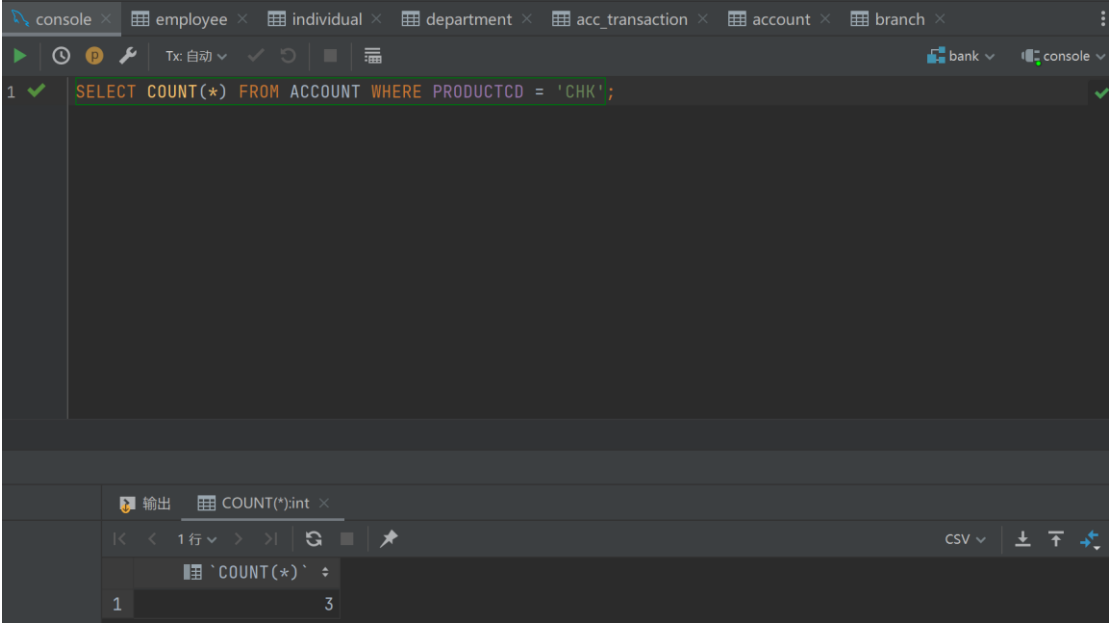
BASIC TASKS

- 

The screenshot shows a database console with a query editor and a results pane. The query is:

```
SELECT COUNT(*) FROM bank.acc_transaction;
```

The results pane shows a single row with the value 6.

1	COUNT(*)
1	6
- 

The screenshot shows a database console with a query editor and a results pane. The query is:

```
SELECT COUNT(*) FROM ACCOUNT WHERE PRODUCTCD = 'CHK';
```

The results pane shows a single row with the value 3.

1	COUNT(*)
1	3

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺ ■

```
1 ✓ SELECT TITLE, COUNT(*) AS COUNT FROM EMPLOYEE GROUP BY TITLE;
```

输出 Result 116 ×

5行 > | ↺ ■ ↗ CSV ▾

	TITLE	COUNT
1	President	1
2	Teller	8
3	Manager	3
4	Analyst	2
5	Clerk	1

3.

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺ ■

```
1 ✓ SELECT c.CUST_ID, COUNT(a.ACCOUNT_ID) AS ACCOUNT_COUNT  
2 FROM CUSTOMER c  
3 LEFT JOIN ACCOUNT a ON c.CUST_ID = a.CUST_ID  
4 GROUP BY c.CUST_ID;
```

输出 Result 117 ×

5行 > | ↺ ■ ↗ C

	CUST_ID	ACCOUNT_COUNT
1	1	2
2	2	1
3	3	0
4	4	2
5	5	1

4.

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺

```
1 ✓ SELECT SUM(AVAIL_BALANCE) FROM ACCOUNT WHERE CUST_ID = 1;
```

输出 SUM(AVAIL_BALANCE):decimal(14,2) ×

1行 > | ↺ | CSV ↓

	SUM(AVAIL_BALANCE)
1	15300.00

5.

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺

```
1 ✓ SELECT c.CUST_ID, SUM(a.AVAIL_BALANCE) AS TOTAL_AVAILABLE_BALANCE
2 FROM CUSTOMER c
3 LEFT JOIN ACCOUNT a ON c.CUST_ID = a.CUST_ID
4 GROUP BY c.CUST_ID;
```

输出 Result 119 ×

5行 > | ↺ |

	CUST_ID	TOTAL_AVAILABLE_BALANCE
1	1	15300.00
2	2	2000.00
3	3	<null>
4	4	11000.00
5	5	12000.00

6.

7.

```

1 SELECT PRODUCTCD, AVG(AVAIL_BALANCE) AS AVERAGE_AVAILABLE_BALANCE
2 FROM ACCOUNT
3 GROUP BY PRODUCTCD;

```

PRODUCTCD	AVERAGE_AVAILABLE_BALANCE
1 CHK	10066.666667
2 SAV	3366.666667

MEDIUM TASKS

8.

```

1 SELECT SUM(AVAIL_BALANCE) FROM ACCOUNT WHERE OPEN_BRANCHID = 1;

```

SUM(AVAIL_BALANCE)
1 15300.00

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺ ■

```

1 ✓ SELECT PRODUCTCD, MAX(AVAIL_BALANCE) AS HIGHEST_AVAILABLE_BALANCE
2   FROM ACCOUNT
3   GROUP BY PRODUCTCD;

```

输出 Result 122 ×

< < 2行 > > | ↺ ■ ★

	PRODUCTCD	HIGHEST_AVAILABLE_BALANCE
1	CHK	12000.00
2	SAV	5100.00

9.

console × employee × individual × department × acc_transaction × account × branch ×

Tx: 自动 ✓ ↺ ■

```

1 ✓ SELECT MIN(AVAIL_BALANCE) FROM ACCOUNT;

```

输出 MIN(AVAIL_BALANCE):decimal(14,2) ×

< < 1行 > > | ↺ ■ ★ CSV

	'MIN(AVAIL_BALANCE)'
1	2000.00

10.

console × employee × individual × department × acc_transaction × account × branch ×

1 ✓ SELECT c.CUST_ID, FLOOR(SUM(a.AVAIL_BALANCE)) AS TOTAL_AVAILABLE_BALANCE
 2 FROM CUSTOMER c
 3 LEFT JOIN ACCOUNT a ON c.CUST_ID = a.CUST_ID
 4 GROUP BY c.CUST_ID;

输出 Result 124 ×

5行

	CUST_ID	TOTAL_AVAILABLE_BALANCE
1	1	15300
2	2	2000
3	3	<null>
4	4	11000
5	5	12000

11.

12. a.

console × employee × individual × department × acc_transaction × account × branch ×

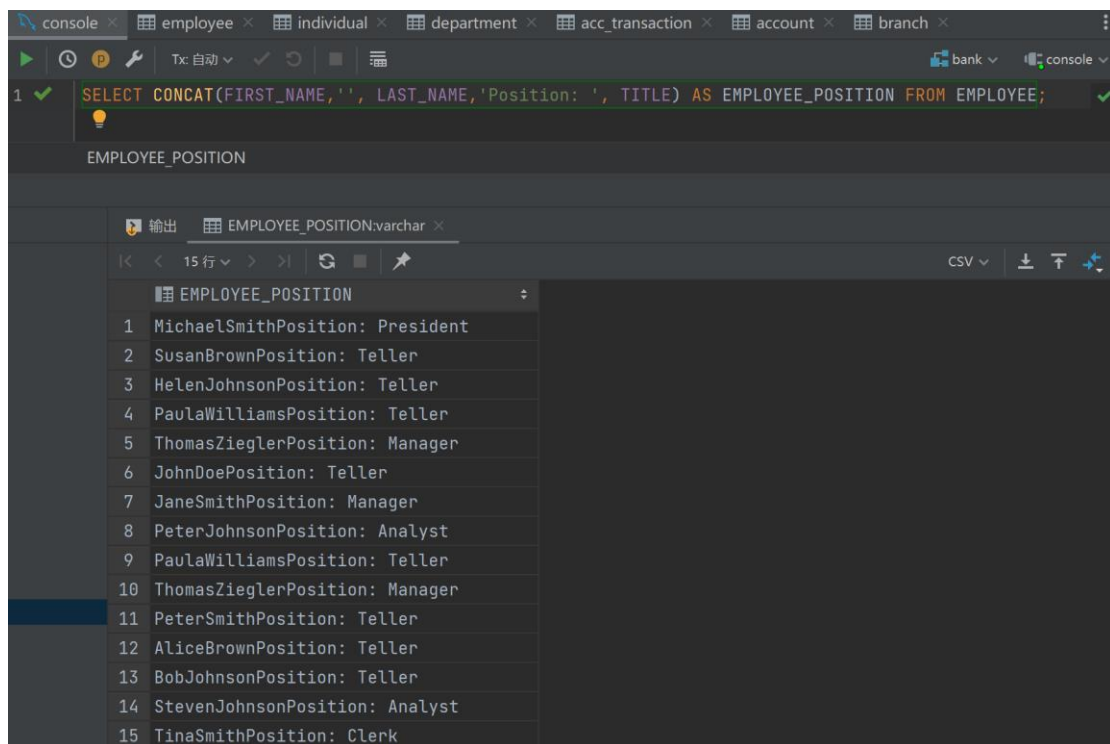
1 ✓ SELECT CONCAT(LAST_NAME, ' ', FIRST_NAME) AS EMPLOYEE_NAME FROM EMPLOYEE;

输出 EMPLOYEE_NAME:varchar ×

15行

	EMPLOYEE_NAME
1	Smith, Michael
2	Brown, Susan
3	Johnson, Helen
4	Williams, Paula
5	Ziegler, Thomas
6	Doe, John
7	Smith, Jane
8	Johnson, Peter
9	Williams, Paula
10	Ziegler, Thomas
11	Smith, Peter
12	Brown, Alice
13	Johnson, Bob
14	Johnson, Steven
15	Smith, Tina

b.



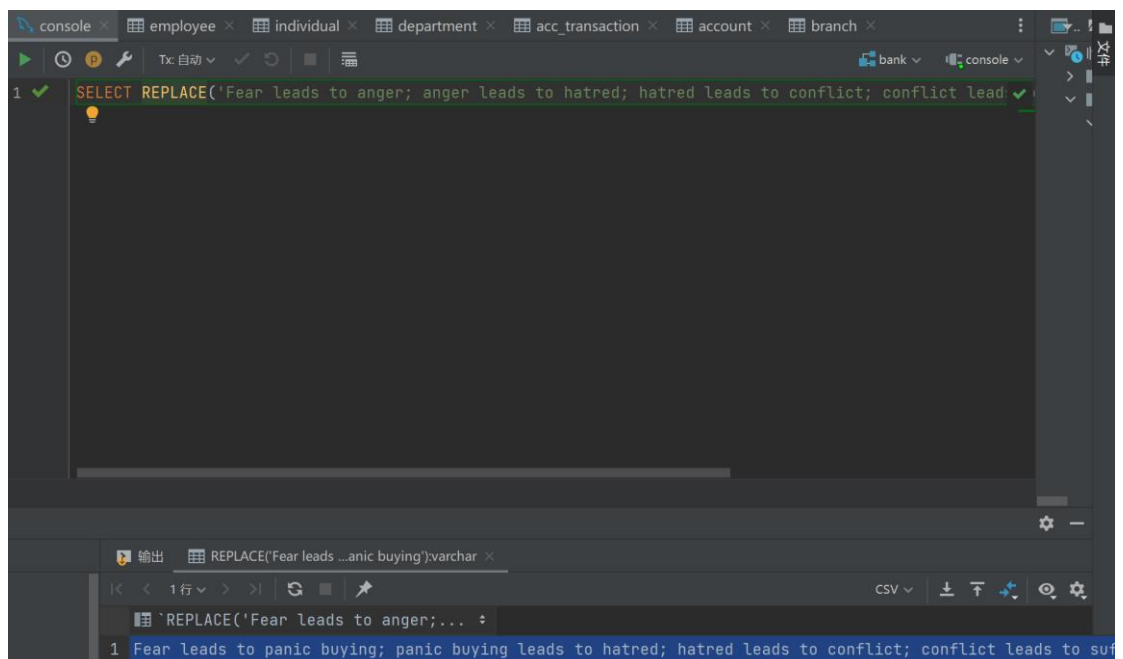
The screenshot shows a database console with a SQL query executed. The query is: `SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME, 'Position: ', TITLE) AS EMPLOYEE_POSITION FROM EMPLOYEE;`. The results are displayed in a table with 15 rows, each showing the concatenated string for an employee. The table has a single column named `EMPLOYEE_POSITION`.

EMPLOYEE_POSITION
1 MichaelSmithPosition: President
2 SusanBrownPosition: Teller
3 HelenJohnsonPosition: Teller
4 PaulaWilliamsPosition: Teller
5 ThomasZieglerPosition: Manager
6 JohnDoePosition: Teller
7 JaneSmithPosition: Manager
8 PeterJohnsonPosition: Analyst
9 PaulaWilliamsPosition: Teller
10 ThomasZieglerPosition: Manager
11 PeterSmithPosition: Teller
12 AliceBrownPosition: Teller
13 BobJohnsonPosition: Teller
14 StevenJohnsonPosition: Analyst
15 TinaSmithPosition: Clerk

ADVANCED TASKS

13.

`SELECT REPLACE('Fear leads to anger; anger leads to hatred; hatred leads to conflict; conflict leads to suffering', 'anger', 'panic buying');`



The screenshot shows a database console with a SQL query executed. The query is: `SELECT REPLACE('Fear leads to anger; anger leads to hatred; hatred leads to conflict; conflict lead`. The result is displayed in a table with 1 row, showing the string after replacing 'anger' with 'panic buying'. The table has a single column named `REPLACE('Fear leads to...panic buying')varchar`.

REPLACE('Fear leads to...panic buying')varchar
1 Fear leads to panic buying; panic buying leads to hatred; hatred leads to conflict; conflict leads to suffering

14.

```
console × employee × individual × department × acc_transaction × account ×
Tx: 自动 ✓ ↺
1 UPDATE CUSTOMER
2 SET FED_ID = CASE
3     WHEN FED_ID LIKE '%-%' THEN REPLACE(FED_ID, '-', '')
4     WHEN FED_ID LIKE '%-%' THEN CONCAT(LEFT(FED_ID, 2), RIGHT(FED_ID, 6))
5     ELSE FED_ID
6 END;
```

customer × employee × individual × department × acc_transaction × account × branch ×

5行 > | Tx: 自动 DDL 搜索 CSV 下载 打印 全屏

WHERE ORDER BY

	CUST_ID	ADDRESS	CITY	CUST_TYPE_CD	FED_ID	POSTAL_CODE	STATE
1	1	123 Main St	Woburn	I	123456789	01801	MA
2	2	456 Elm St	Concord	I	987654321	03301	NH
3	3	789 Oak St	Portsmouth	B	111223333	03801	NH
4	4	888 Pine St	Boston	I	444556666	02101	MA
5	5	999 Cedar St	Providence	I	777889999	02901	RI

15.

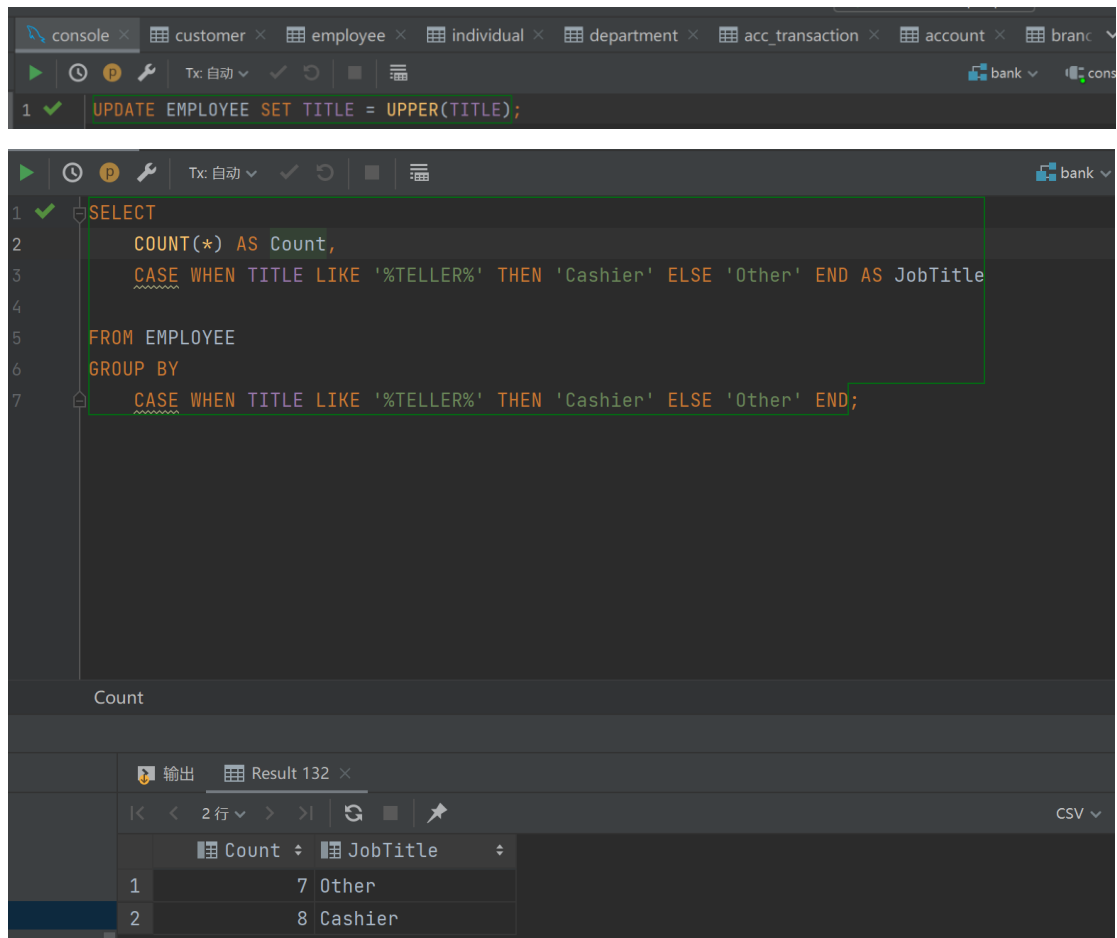
```
console × customer × employee × individual × department × acc_transaction × account ×
Tx: 自动 ✓ ↺
1 SELECT YEAR(TXN_DATE) AS YEAR, COUNT(*) AS COUNT
2 FROM bank.acc_transaction
3 GROUP BY YEAR(TXN_DATE);
```

输出 Result 129 ×

1行 > | CSV

	YEAR	COUNT
1	2023	6

16.



The screenshot shows a SQL IDE interface. The top panel displays two SQL statements. The first is an UPDATE statement: `UPDATE EMPLOYEE SET TITLE = UPPER(TITLE);`. The second is a SELECT statement: `SELECT COUNT(*) AS Count, CASE WHEN TITLE LIKE '%TELLER%' THEN 'Cashier' ELSE 'Other' END AS JobTitle FROM EMPLOYEE GROUP BY CASE WHEN TITLE LIKE '%TELLER%' THEN 'Cashier' ELSE 'Other' END;`. The bottom panel shows the results of the SELECT statement in a table with two columns: Count and JobTitle. The results are: 1 Other and 8 Cashier.

```

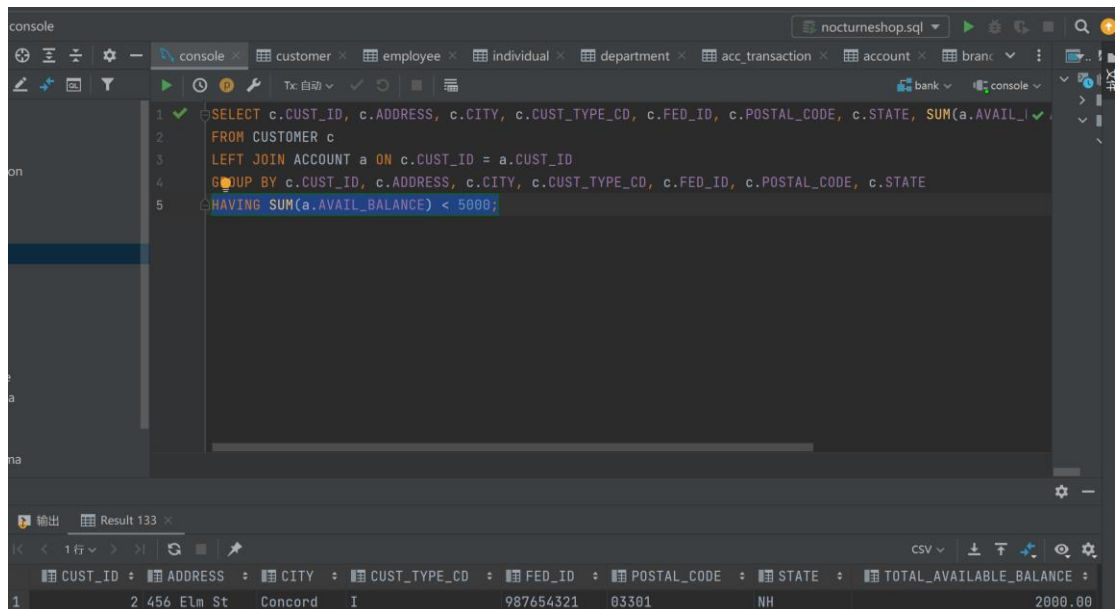
1 UPDATE EMPLOYEE SET TITLE = UPPER(TITLE);

2 SELECT
3     COUNT(*) AS Count,
4     CASE WHEN TITLE LIKE '%TELLER%' THEN 'Cashier' ELSE 'Other' END AS JobTitle
5 FROM EMPLOYEE
6 GROUP BY
7     CASE WHEN TITLE LIKE '%TELLER%' THEN 'Cashier' ELSE 'Other' END;

```

Count	JobTitle
1	Other
8	Cashier

17.



The screenshot shows a SQL IDE interface. The top panel displays a complex SELECT statement: `SELECT c.CUST_ID, c.ADDRESS, c.CITY, c.CUST_TYPE_CD, c.FED_ID, c.POSTAL_CODE, c.STATE, SUM(a.AVAIL_BALANCE) AS TOTAL_AVAILABLE_BALANCE FROM CUSTOMER c LEFT JOIN ACCOUNT a ON c.CUST_ID = a.CUST_ID GROUP BY c.CUST_ID, c.ADDRESS, c.CITY, c.CUST_TYPE_CD, c.FED_ID, c.POSTAL_CODE, c.STATE HAVING SUM(a.AVAIL_BALANCE) < 5000;`. The bottom panel shows the results of the SELECT statement in a table with columns: CUST_ID, ADDRESS, CITY, CUST_TYPE_CD, FED_ID, POSTAL_CODE, STATE, and TOTAL_AVAILABLE_BALANCE. The results are: 1 2 456 Elm St Concord I 987654321 03301 NH 2000.00.

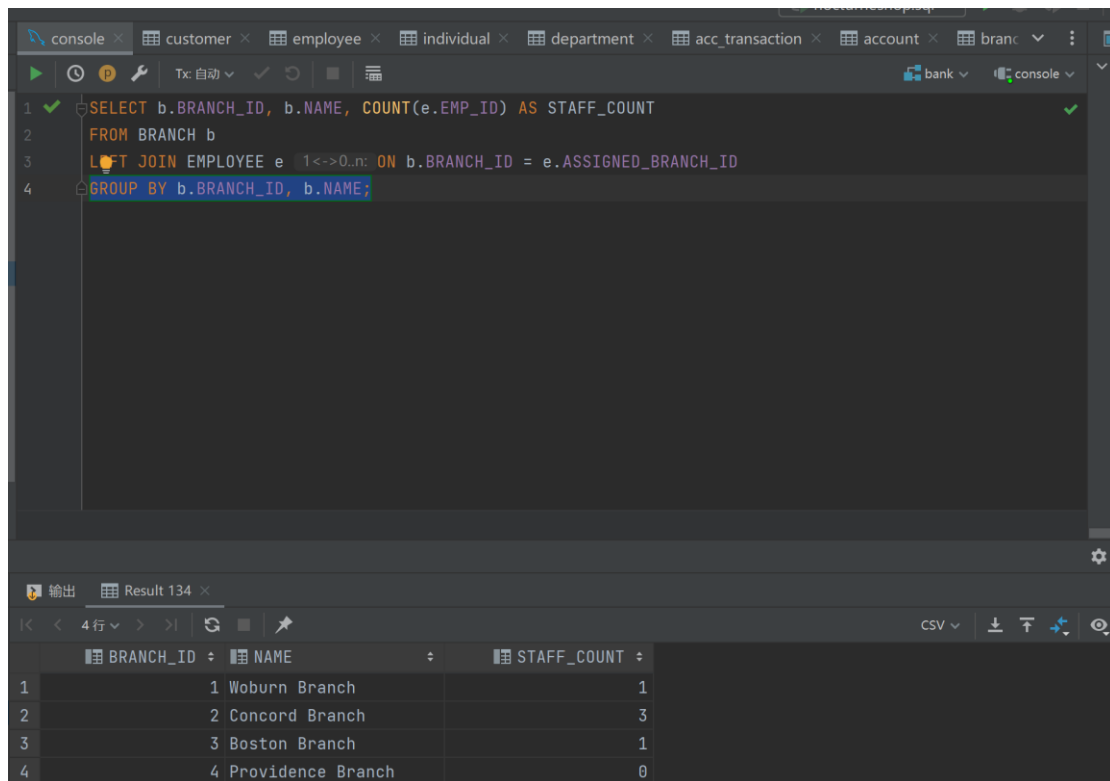
```

1 SELECT c.CUST_ID, c.ADDRESS, c.CITY, c.CUST_TYPE_CD, c.FED_ID, c.POSTAL_CODE, c.STATE, SUM(a.AVAIL_BALANCE) AS TOTAL_AVAILABLE_BALANCE
2 FROM CUSTOMER c
3 LEFT JOIN ACCOUNT a ON c.CUST_ID = a.CUST_ID
4 GROUP BY c.CUST_ID, c.ADDRESS, c.CITY, c.CUST_TYPE_CD, c.FED_ID, c.POSTAL_CODE, c.STATE
5 HAVING SUM(a.AVAIL_BALANCE) < 5000;

```

CUST_ID	ADDRESS	CITY	CUST_TYPE_CD	FED_ID	POSTAL_CODE	STATE	TOTAL_AVAILABLE_BALANCE
1	2 456 Elm St	Concord	I	987654321	03301	NH	2000.00

18.

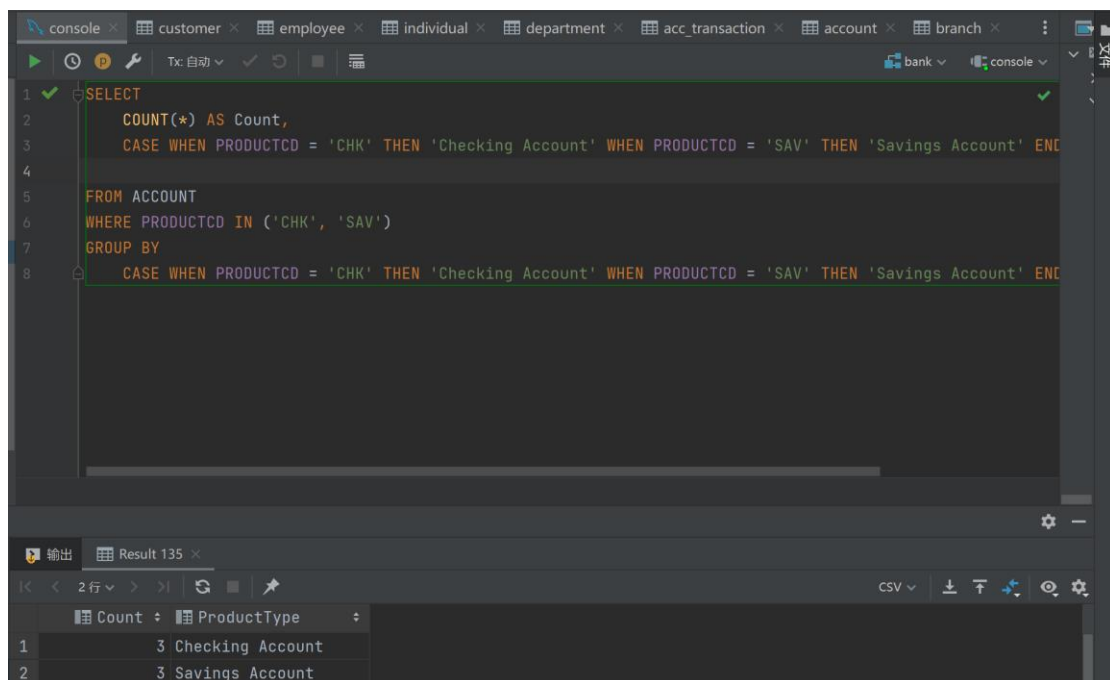


The screenshot shows a database console interface with a SQL query and its results. The query is a LEFT JOIN between the BRANCH and EMPLOYEE tables, counting the number of employees assigned to each branch. The results are displayed in a table with 4 rows.

```
1 SELECT b.BRANCH_ID, b.NAME, COUNT(e.EMP_ID) AS STAFF_COUNT
2 FROM BRANCH b
3 LEFT JOIN EMPLOYEE e 1<->0..n: ON b.BRANCH_ID = e.ASSIGNED_BRANCH_ID
4 GROUP BY b.BRANCH_ID, b.NAME;
```

BRANCH_ID	NAME	STAFF_COUNT
1	Woburn Branch	1
2	Concord Branch	3
3	Boston Branch	1
4	Providence Branch	0

19.



The screenshot shows a database console interface with a SQL query and its results. The query is a SELECT statement that counts the number of accounts for each product type, categorized by a CASE WHEN statement. The results are displayed in a table with 2 rows.

```
1 SELECT
2   COUNT(*) AS Count,
3   CASE WHEN PRODUCTCD = 'CHK' THEN 'Checking Account' WHEN PRODUCTCD = 'SAV' THEN 'Savings Account' END
4
5 FROM ACCOUNT
6 WHERE PRODUCTCD IN ('CHK', 'SAV')
7 GROUP BY
8   CASE WHEN PRODUCTCD = 'CHK' THEN 'Checking Account' WHEN PRODUCTCD = 'SAV' THEN 'Savings Account' END
```

Count	ProductType
3	Checking Account
3	Savings Account