

Omkar Mankame

Few SQL projects

4 Sept 2024

I have done a few projects using SQL for data analysis. I have shared three complex and concise projects in this document. Others have been shared in the GitHub repository. The code is also mentioned in the GitHub codespace.

Project 1 – Joining 3 tables

1. Created a database and 3 tables.
2. Inserted values in the tables.

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'SCHEMAS' tree with 'myproject' selected. The central pane shows the following SQL script:

```
2 Use myproject
3
4 CREATE TABLE branch_table(
5     branch_id int
6 );
7
8 INSERT INTO branch_table (branch_id)
9 VALUES ('101'),
10        ('102'),
11        ('103'),
12        ('104');
13
14 CREATE TABLE link_branch_int_org(
15     branch_id int,
16     int_org int
17 );
18
19 INSERT INTO link_branch_int_org (branch_id, int_org)
20 VALUES ('101', '1001'),
21        ('101', '10052'),
```

The right pane shows the 'SQLAdditions' toolbar with an 'Automatic caret position' tooltip. The bottom pane shows the 'Output' window with the following results:



#	Time	Action	Message
4	15:23:06	SELECT A.full_name AS employee_name, B.full_name AS manager FROM employee_table ...	4 row(s) returned
5	15:25:43	SELECT * FROM (SELECT branch_table.branch_id, branch_table.branch_name, link_...	5 row(s) returned

3. The Tables are as below.

134

135 • `SELECT *`

136 `FROM branch_table;`



<   Filter Rows: | Edit:

	branch_id	branch_name
▶	101	Toronto
	102	Guelph
	103	Mississauga
	104	Oakville
*	NULL	NULL

134

155 • `SELECT *`

156 `FROM internal_organization_table;`



<   Filter Rows: | Export: 

	int_org_id	org_name	code_	party_id
▶	10052	org Guelph	3	101
	12	NULL	4	101
	10001	org Toronto	3	102
	7	NULL	4	103

134

141 • `SELECT *`

142 `FROM link_branch_int_org;`

<   Filter Rows: | Export:

	branch_id	int_org
▶	101	1001
	101	10052
	102	1002
	103	10052
	104	NULL

4. Joined two tables each to get desired elements.

Left join is used to get all the values from int_org and the matching values from link_branch_int_org.

```
44
45 • SELECT link_branch_int_org.branch_id,
46         link_branch_int_org.int_org AS group_code,
47         int_org.org_name AS group_name,
48         int_org.code_ AS region_code
49 FROM link_branch_int_org
50 LEFT JOIN int_org
51     ON link_branch_int_org.int_org = int_org.int_org;
52
53 ☺ /*
```

<

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

	branch_id	group_code	group_name	region_code
▶	101	1001	Toronto	3
	101	10052	null	4
	102	1002	Guelph	3
	103	10052	null	4
	104	NULL	NULL	NULL

Inner join is used to get all the values from both the tables including the ones that don't match.

```
77 • SELECT branch_table.branch_id,
78         branch_table.branch_name,
79         link_branch_int_org.int_org
80 FROM branch_table
81 INNER JOIN link_branch_int_org
82     ON link_branch_int_org.branch_id = branch_table.branch_id;
83
```

<

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

	branch_id	branch_name	int_org
▶	101	Toronto	1001
	101	Toronto	10052
	102	Guelph	1002
	103	Mississauga	10052
	104	Oakville	NULL

Result 27 x

5. Created 5 joins to get the desired result.

1st - Left Join

```
36 • SELECT link_branch_int_org.branch_id,
37         link_branch_int_org.int_org AS group_code,
38         int_org.org_name AS group_name,
39         int_org.code_ AS region_code
40 FROM link_branch_int_org
41 LEFT JOIN int_org
42     ON link_branch_int_org.int_org = int_org.int_org;
43
```

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

	branch_id	group_code	group_name	region_code
▶	101	1001	Toronto	3
	101	10052	null	4
	102	1002	Guelph	3
	103	10052	null	4
	104	NULL	NULL	NULL

Result 3

2nd – Inner Join

3joins 3tables_join* x employee_manager Credit Card Limit

Limit to 1000 rows

```
60 WHERE branch_id = '103';
61
62 • UPDATE branch_table
63 SET branch_name = 'Oakville'
64 WHERE branch_id = '104';
65
66 • SELECT branch_table.branch_id,
67         branch_table.branch_name,
68         link_branch_int_org.int_org
69 FROM branch_table
70 INNER JOIN link_branch_int_org
71     ON link_branch_int_org.branch_id = branch_table.branch_id;
72
```

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

	branch_id	branch_name	int_org
▶	101	Toronto	1001
	101	Toronto	10052
	102	Guelph	1002
	103	Mississauga	10052
	104	Oakville	NULL

Result 4

3rd – Left Join

The screenshot shows the SQL Developer interface with a query window titled '3joins'. The query is as follows:

```
71      ON link_branch_int_org.branch_id = branch_table.branch_id;
72
73 •   SELECT link_branch_int_org.branch_id,
74          link_branch_int_org.int_org,
75          int_org.org_name,
76          int_org.code_
77 FROM link_branch_int_org
78 LEFT JOIN int_org
79      ON link_branch_int_org.int_org = int_org.int_org;
80
81 •   SELECT link_branch_int_org.branch_id,
82          link_branch_int_org.int_org,
83          int org.org name,
```

Below the query, the 'Result Grid' shows the following data:

branch_id	int_org	org_name	code_
101	1001	Toronto	3
101	10052	null	4
102	1002	Guelph	3
103	10052	null	4
104	NULL	NULL	NULL

The interface also shows a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and an 'Export' button.

4th – Left Join

The screenshot shows the SQL Developer interface with a query window titled '3joins'. The query is as follows:

```
79      ON link_branch_int_org.int_org = int_org.int_org;
80
81 •   SELECT link_branch_int_org.branch_id,
82          link_branch_int_org.int_org,
83          int_org.org_name,
84          int_org.code_
85 FROM link_branch_int_org
86 LEFT JOIN int_org
87      ON link_branch_int_org.int_org = int_org.int_org
88 WHERE int_org.code_ = '03';
89
90 •   SELECT link_branch_int_org.branch_id,
91          link branch int org.int org,
```

Below the query, the 'Result Grid' shows the following data:

branch_id	int_org	org_name	code_
101	1001	Toronto	3
102	1002	Guelph	3

The interface also shows a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and an 'Export' button.


```

89
90 • SELECT link_branch_int_org.branch_id,
91         link_branch_int_org.int_org,
92         int_org.org_name,
93         int_org.code_
94 FROM link_branch_int_org
95 LEFT JOIN int_org
96     ON link_branch_int_org.int_org = int_org.int_org
97 WHERE int_org.code_ = '04';
98
99 • SELECT *
100 FROM (SELECT link_branch_int_org.branch_id,
101         link_branch_int_org.int_org,

```

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

	branch_id	int_org	org_name	code_
▶	101	10052	null	4
	103	10052	null	4

Limit to 1000 rows

```

118 • SELECT *
119 FROM (SELECT branch_table.branch_id,
120         branch_table.branch_name,
121         link_branch_int_org.int_org
122 FROM branch_table
123 INNER JOIN link_branch_int_org
124     ON link_branch_int_org.branch_id = branch_table.branch_id) AS A
125 LEFT JOIN (SELECT link_branch_int_org.branch_id,
126         link_branch_int_org.int_org,
127         int_org.org_name,
128         int_org.code_
129 FROM link_branch_int_org
130 LEFT JOIN int_org
131     ON link_branch_int_org.int_org = int_org.int_org
132 WHERE int_org.code_ = '03') AS B
133 ON A.int_org = B.int_org;

```

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

	branch_id	branch_name	int_org	branch_id	int_org	org_name	code_
▶	101	Toronto	1001	101	1001	Toronto	3
	101	Toronto	10052	NULL	NULL	NULL	NULL
	102	Guelph	1002	102	1002	Guelph	3
	103	Mississauga	10052	NULL	NULL	NULL	NULL
	104	Oakville	NULL	NULL	NULL	NULL	NULL

6. The complex query above created a join, another join in the subquery made it possible to get the desired result as shown in the image.

SQL code (Also available in GitHub Codespace) –

Use myproject;

```
/* CREATE TABLE branch_table(  
    branch_id int  
);
```

```
INSERT INTO branch_table (branch_id)  
VALUES ('101'),  
    ( '102'),  
    ('103'),  
    ('104');
```

```
CREATE TABLE link_branch_int_org(  
    branch_id int,  
    int_org int  
);
```

```
INSERT INTO link_branch_int_org (branch_id, int_org)  
VALUES ('101', '1001'),  
    ('101', '10052'),  
    ('102', '1002'),  
    ('103', '10052');
```

```
('104', null);
```

```
CREATE TABLE int_org(  
    int_org int,  
    org_name varchar(100),  
    code_int  
);
```

```
INSERT INTO int_org (int_org, org_name, code_)  
VALUES ('1001', 'Toronto', '03'),  
        ('1002', 'Guelph', '03'),  
        ('10052', 'null', '04');
```

```
UPDATE internal_organization_table  
SET org_name = 'org Guelph'  
WHERE int_org_id = '10052';
```

```
UPDATE internal_organization_table  
SET org_name = 'org Toronto'  
WHERE int_org_id = '10001';  
*/
```

```
SELECT link_branch_int_org.branch_id,  
        link_branch_int_org.int_org AS group_code,  
        int_org.org_name AS group_name,  
        int_org.code_ AS region_code  
FROM link_branch_int_org  
LEFT JOIN int_org  
    ON link_branch_int_org.int_org = int_org.int_org;
```



```
/*
```

```
ALTER TABLE branch_table  
ADD branch_name varchar(100);
```

```
ALTER TABLE branch_table  
ADD PRIMARY KEY (branch_id);
```

```
UPDATE branch_table  
SET branch_name ='Toronto'  
WHERE branch_id = '101';
```

```
UPDATE branch_table  
SET branch_name ='Guelph'  
WHERE branch_id = '102';
```

```
UPDATE branch_table  
SET branch_name ='Mississauga'  
WHERE branch_id = '103';
```

```
UPDATE branch_table  
SET branch_name ='Oakville'  
WHERE branch_id = '104';
```

```
*/
```

```
SELECT branch_table.branch_id,  
       branch_table.branch_name,  
       link_branch_int_org.int_org  
FROM branch_table
```

```
INNER JOIN link_branch_int_org
    ON link_branch_int_org.branch_id = branch_table.branch_id;
```

```
SELECT link_branch_int_org.branch_id,
       link_branch_int_org.int_org,
       int_org.org_name,
       int_org.code_
FROM link_branch_int_org
LEFT JOIN int_org
    ON link_branch_int_org.int_org = int_org.int_org;
```

```
SELECT link_branch_int_org.branch_id,
       link_branch_int_org.int_org,
       int_org.org_name,
       int_org.code_
FROM link_branch_int_org
LEFT JOIN int_org
    ON link_branch_int_org.int_org = int_org.int_org
WHERE int_org.code_ = '03';
```

```
SELECT link_branch_int_org.branch_id,
       link_branch_int_org.int_org,
       int_org.org_name,
       int_org.code_
FROM link_branch_int_org
LEFT JOIN int_org
    ON link_branch_int_org.int_org = int_org.int_org
WHERE int_org.code_ = '04';
```

```

SELECT *
FROM (SELECT link_branch_int_org.branch_id,
            link_branch_int_org.int_org,
            int_org.org_name,
            int_org.code_
FROM link_branch_int_org
LEFT JOIN int_org
        ON link_branch_int_org.int_org = int_org.int_org
WHERE int_org.code_ = '03') AS A
LEFT JOIN (SELECT link_branch_int_org.branch_id,
            link_branch_int_org.int_org,
            int_org.org_name,
            int_org.code_
FROM link_branch_int_org
LEFT JOIN int_org
        ON link_branch_int_org.int_org = int_org.int_org
WHERE int_org.code_ = '04') AS B
        ON A.int_org = B.int_org;

```

```

SELECT *
FROM (SELECT branch_table.branch_id,
            branch_table.branch_name,
            link_branch_int_org.int_org
FROM branch_table
INNER JOIN link_branch_int_org
        ON link_branch_int_org.branch_id = branch_table.branch_id) AS A
LEFT JOIN (SELECT link_branch_int_org.branch_id,
            link_branch_int_org.int_org,
            int_org.org_name,

```

```
int_org.code_  
FROM link_branch_int_org  
LEFT JOIN int_org  
    ON link_branch_int_org.int_org = int_org.int_org  
WHERE int_org.code_ = '03') AS B  
    ON A.int_org = B.int_org;
```

```
SELECT *  
FROM branch_table;
```

```
SELECT *  
FROM int_org;
```

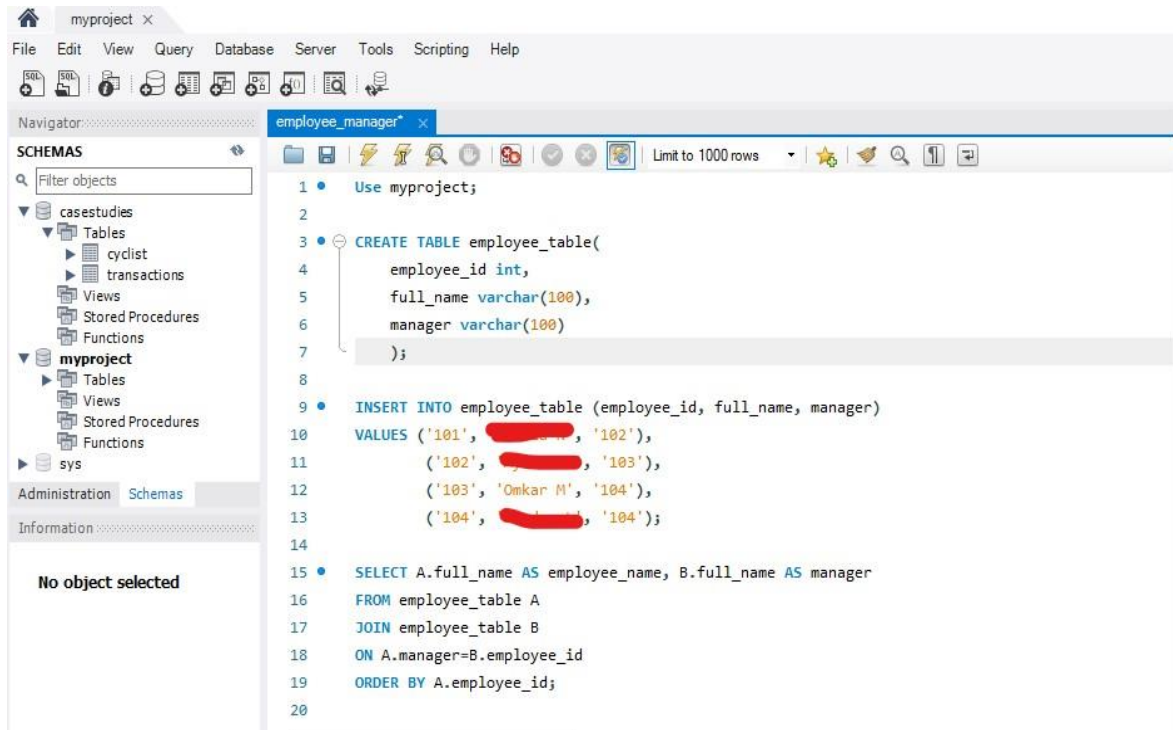
```
SELECT *  
FROM link_branch_int_org;
```

```
SELECT *  
FROM internal_organization_table;
```

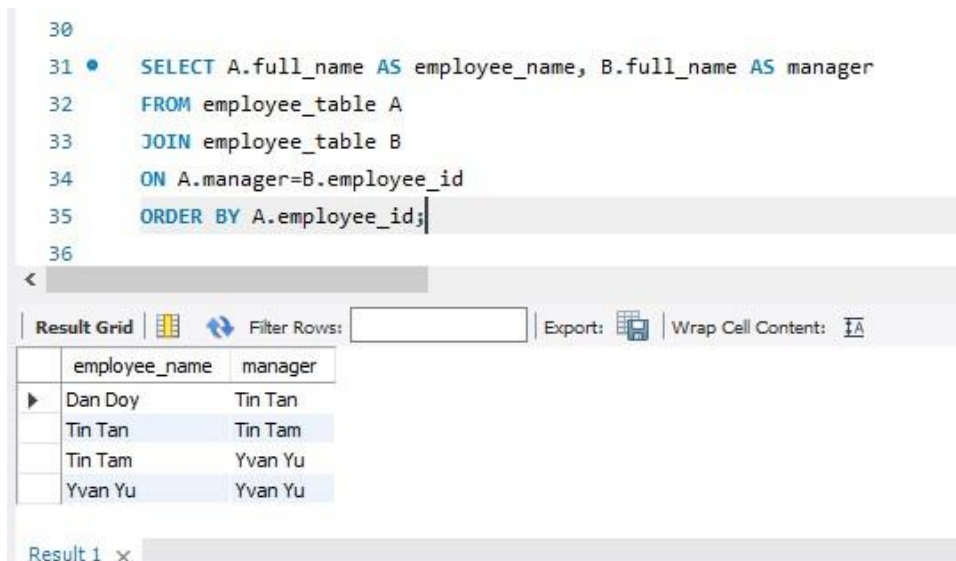
Project 2 - Employee_Manager Table – Self Join

1. Created database.
2. Created a table with employees, id and manager. Inserted values.

The values have been updated in the result table. The below image is for reference and understanding only.



3. Used Self Join to produce result as seen below.



SQL code (Also available in GitHub Codespace) –

Use myproject;

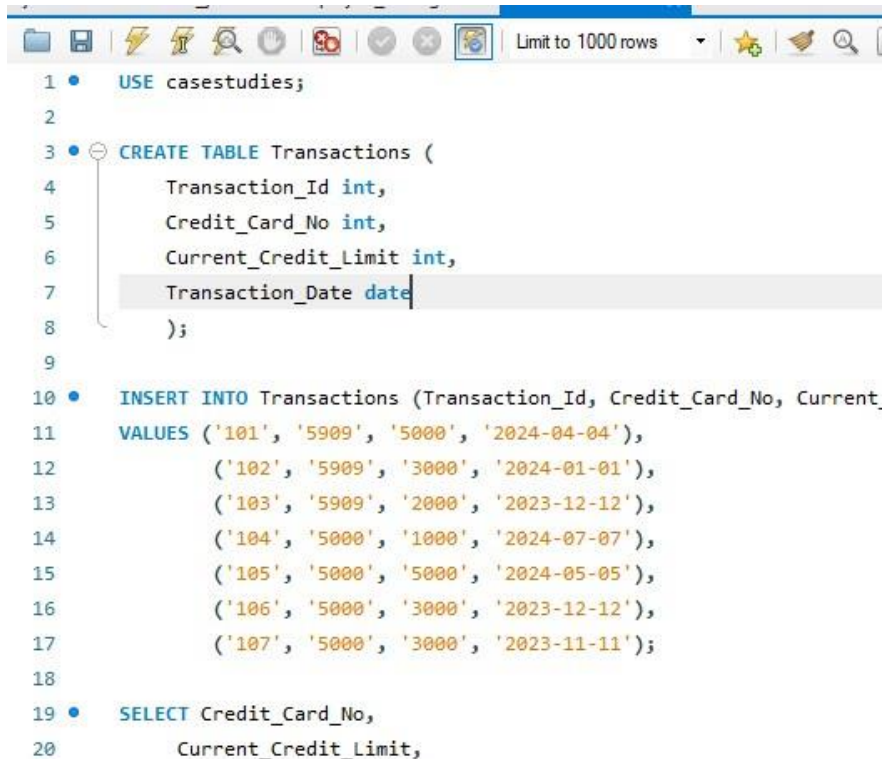
```
CREATE TABLE employee_table(  
    employee_id int,  
    full_name varchar(100),  
    manager varchar(100)  
);
```

```
INSERT INTO employee_table (employee_id, full_name, manager)  
VALUES ('101', 'Dan Doy ', '102'),  
    ('102', 'Tin Tan', '103'),  
    ('103', ' Tin Tam ', '104'),  
    ('104', 'Yvan Yu ', '104');
```

```
SELECT A.full_name AS employee_name, B.full_name AS manager  
FROM employee_table A  
JOIN employee_table B  
ON A.manager=B.employee_id  
ORDER BY A.employee_id;
```

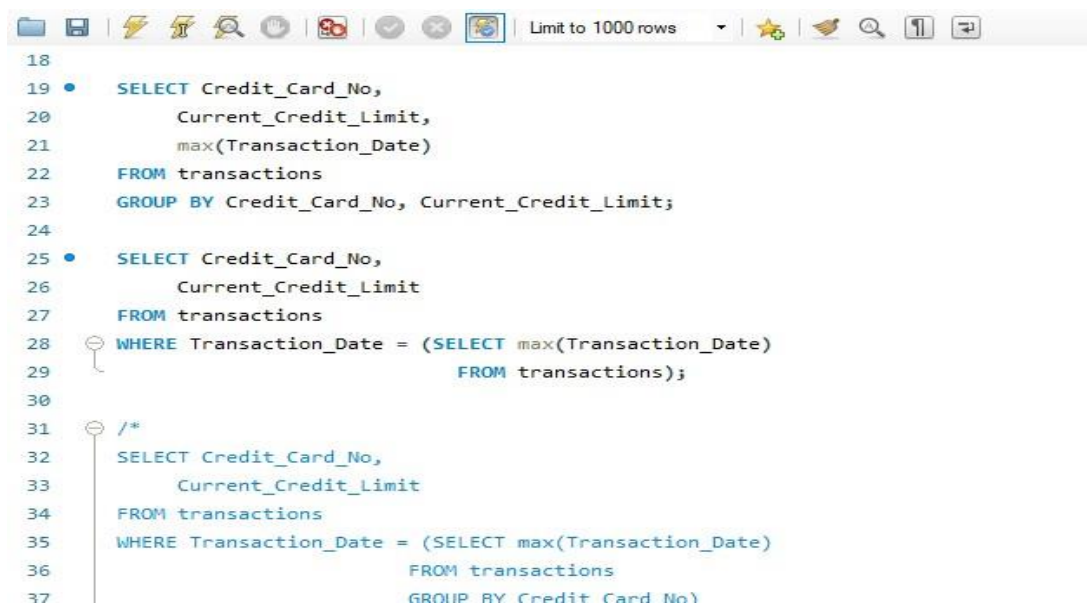
Project 3 – Credit Limit of Credit Card

1. Created database and Table with transaction id, credit card number, credit card limit and transaction date.
2. Inserted values in the table.



```
1 • USE casestudies;
2
3 • CREATE TABLE Transactions (
4     Transaction_Id int,
5     Credit_Card_No int,
6     Current_Credit_Limit int,
7     Transaction_Date date
8 );
9
10 • INSERT INTO Transactions (Transaction_Id, Credit_Card_No, Current_
11 VALUES ('101', '5909', '5000', '2024-04-04'),
12         ('102', '5909', '3000', '2024-01-01'),
13         ('103', '5909', '2000', '2023-12-12'),
14         ('104', '5000', '1000', '2024-07-07'),
15         ('105', '5000', '5000', '2024-05-05'),
16         ('106', '5000', '3000', '2023-12-12'),
17         ('107', '5000', '3000', '2023-11-11');
18
19 • SELECT Credit_Card_No,
20         Current_Credit_Limit,
```

3. Understood data and framed results using joins.



```
18
19 • SELECT Credit_Card_No,
20         Current_Credit_Limit,
21         max(Transaction_Date)
22 FROM transactions
23 GROUP BY Credit_Card_No, Current_Credit_Limit;
24
25 • SELECT Credit_Card_No,
26         Current_Credit_Limit
27 FROM transactions
28 WHERE Transaction_Date = (SELECT max(Transaction_Date)
29                           FROM transactions);
30
31 /*
32 SELECT Credit_Card_No,
33         Current_Credit_Limit
34 FROM transactions
35 WHERE Transaction_Date = (SELECT max(Transaction_Date)
36                           FROM transactions
37                           GROUP BY Credit Card No)
```


4. Final result was attained using joins, group by and alias.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'SCHEMAS' tree with 'casestudies' expanded, showing 'Tables' (cyclist, transactions) and 'Views'. The main query window is titled 'Credit Card Limit' and contains the following SQL code:

```
64 FROM transactions
65 GROUP BY Credit_Card_No)
66 ON Credit_Card_No.home = groupedtt.home
67 AND tt.datetime = groupedtt.MaxDateTime;
68 */
69
70 • SELECT Credit_Card_No,
71         Current_Credit_Limit
72 FROM transactions AS a
73 WHERE Transaction_Date = (SELECT max(Transaction_Date)
74                           FROM transactions as b
75                           WHERE a.Credit_Card_No = b.Credit_Card_No);
76
```

The bottom pane shows the 'Result Grid' with the following data:

Credit_Card_No	Current_Credit_Limit
5909	5000
5000	1000

Annotations on the image include 'Automatic o toolbar to m caret positi' on the right and 'No object selected' at the bottom left.

SQL code (Also available in GitHub Codespace) –

USE casestudies;

CREATE TABLE Transactions (

Transaction_Id int,

Credit_Card_No int,

Current_Credit_Limit int,

Transaction_Date date

);

INSERT INTO Transactions (Transaction_Id, Credit_Card_No, Current_Credit_Limit, Transaction_Date)

```
VALUES ('101', '5909', '5000', '2024-04-04'),  
        ('102', '5909', '3000', '2024-01-01'),  
        ('103', '5909', '2000', '2023-12-12'),  
        ('104', '5000', '1000', '2024-07-07'),  
        ('105', '5000', '5000', '2024-05-05'),  
        ('106', '5000', '3000', '2023-12-12'),  
        ('107', '5000', '3000', '2023-11-11');
```

```
SELECT Credit_Card_No,  
        Current_Credit_Limit,  
        max(Transaction_Date)  
FROM transactions  
GROUP BY Credit_Card_No, Current_Credit_Limit;
```

```
SELECT Credit_Card_No,  
        Current_Credit_Limit  
FROM transactions  
WHERE Transaction_Date = (SELECT max(Transaction_Date)  
                           FROM transactions);
```

```
/*  
SELECT Credit_Card_No,  
        Current_Credit_Limit  
FROM transactions  
WHERE Transaction_Date = (SELECT max(Transaction_Date)  
                           FROM transactions  
                           GROUP BY Credit_Card_No)  
        GROUP BY Transaction_Date  
ORDER BY Transaction_Date;
```

```

SELECT Credit_Card_No,
        Current_Credit_Limit
FROM transactions AS s1
JOIN (SELECT
        Credit_Card_No,
        Current_Credit_Limit,
        MAX(`Transaction_Date`)
FROM transactions AS dt
GROUP BY Credit_Card_No) AS s2;

ON s1.Credit_Card_No = s2.Credit_Card_No
ORDER BY `datetime`;

```

```

SELECT Credit_Card_No,
        Current_Credit_Limit
FROM transactions
Transaction_Date;

```

```

SELECT Credit_Card_No,
        Current_Credit_Limit
FROM transactions
INNER JOIN
    (SELECT Credit_Card_No, MAX(Transaction_Date)
    FROM transactions
    GROUP BY Credit_Card_No)
ON Credit_Card_No.home = groupedtt.home
AND tt.datetime = groupedtt.MaxDateTime;

*/

```

```
SELECT Credit_Card_No,  
       Current_Credit_Limit  
FROM transactions AS a  
WHERE Transaction_Date = (SELECT max(Transaction_Date)  
                          FROM transactions as b  
                          WHERE a.Credit_Card_No = b.Credit_Card_No);
```

```
SELECT max(Transaction_Date)  
      FROM transactions;
```