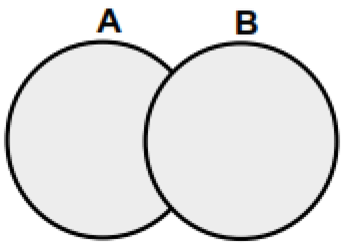
SET OPERATIONS

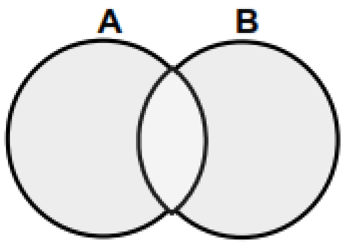
This topic is another important one, as it covers queries which involve set operators. This topic will cover all the set operators available and how they are to be used within queries. Set operators work on the result of different queries. These operators will combine the results of two/more queries into one result.

SQL Server offers four different type of set operators:

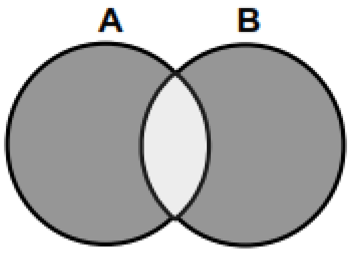
i) UNION: all distinct rows selected by either query/all elements in two sets



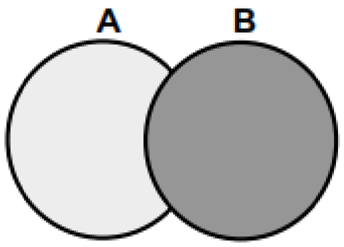
ii) UNION ALL: all rows selected by either query including duplicate rows



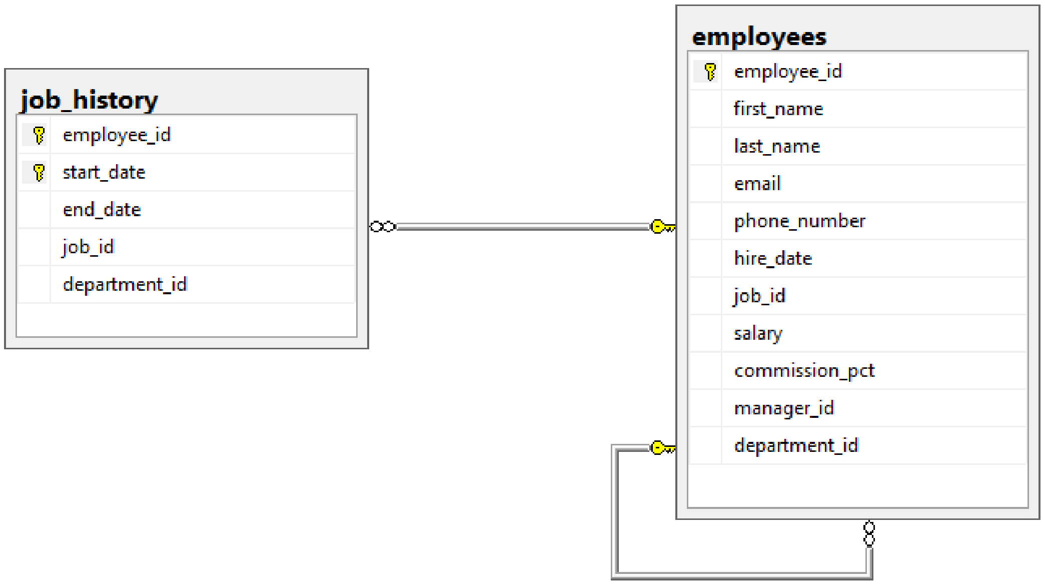
iii) INTERSECT: all distinct rows selected by both queries



iv) EXCEPT: all distinct rows that are selected by the first SELECT statement and which are not selected in the second SELECT statement.



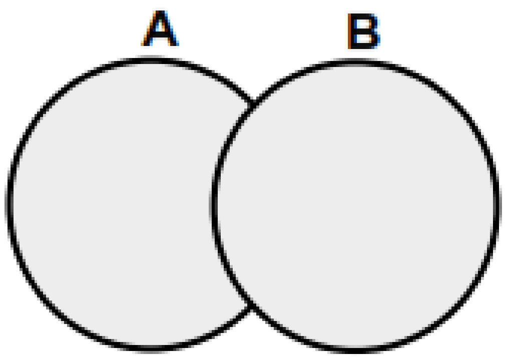
Throughout this topic we will be using the *employees* and *job_history* tables to explain the difference between the set operators that have been mentioned above. In the figure below, the tables, columns and relationships that pertain to these two tables are included.



As it can be clearly noted, the employees table stores all the information related to the employee. On the other had the job_history table is a table that keeps track of the different jobs performed by each employee while working with the company. There are certain instances where were employees have held the same job position in two different periods of time.

The UNION operator

As described in the introductory parts of this pack, the UNION operator will work on the results which are obtained from two queries and will select all the values from both results while eliminating duplicates.

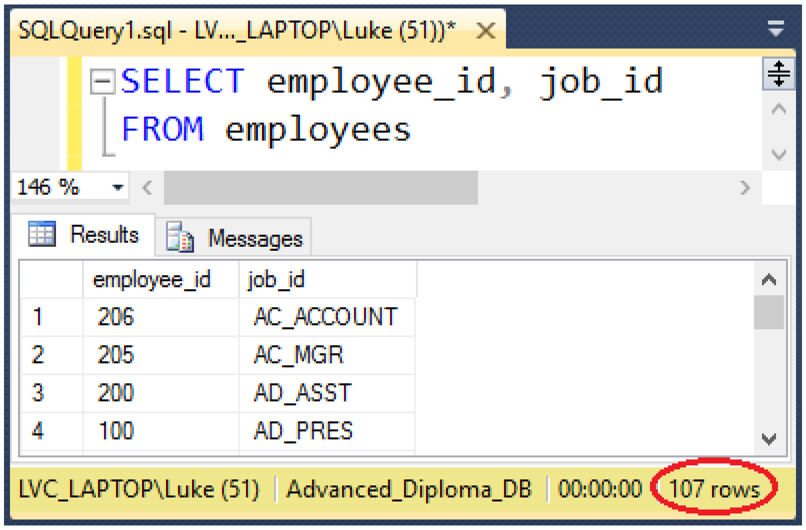


Important points regarding UNION operator:

- The SELECT statements in the queries should have the same number of columns and their data types should be identical. The only thing that can differ is the name of the columns.
- This operator considers NULL values while checking for duplicate entries

Example 1:

1) Write a query that will display the employee number and job_id within the employees table.



SQLQuery1.sql - LV..._LAPTOP\Luke (51))*

```
SELECT employee_id, job_id
FROM employees
```

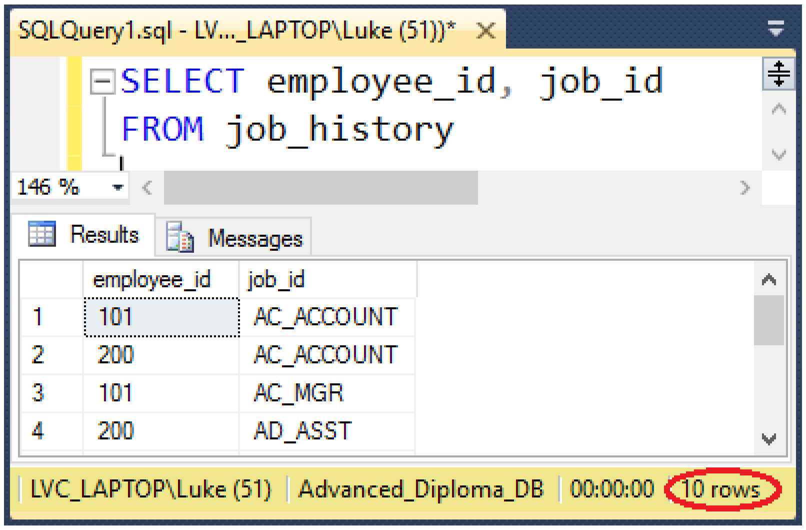
146 %

Results | Messages

	employee_id	job_id
1	206	AC_ACCOUNT
2	205	AC_MGR
3	200	AD_ASST
4	100	AD_PRES

LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 107 rows

2) Write a query that will display the employee number and job_id this time from the job_history table



SQLQuery1.sql - LV..._LAPTOP\Luke (51))*

```
SELECT employee_id, job_id
FROM job_history
```

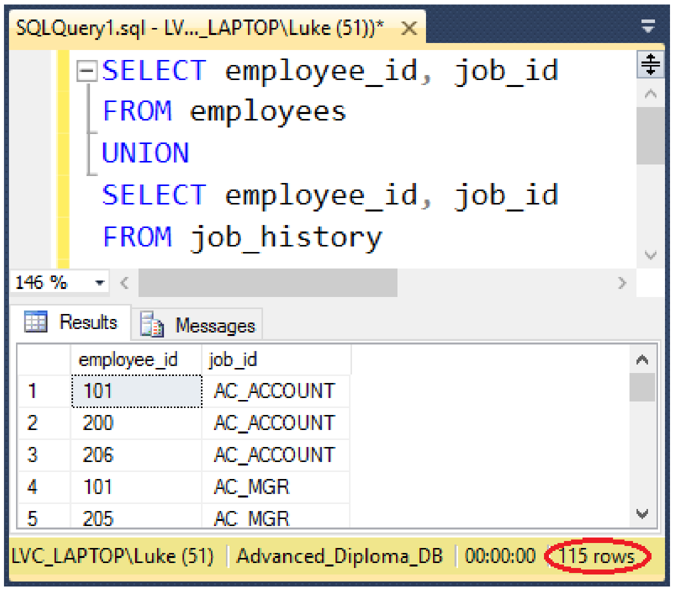
146 %

Results | Messages

	employee_id	job_id
1	101	AC_ACCOUNT
2	200	AC_ACCOUNT
3	101	AC_MGR
4	200	AD_ASST

LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 10 rows

3) Now we are to apply the UNION operator on both queries created above.



SQLQuery1.sql - LV..._LAPTOP\Luke (51))*

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history
```

146 %

Results | Messages

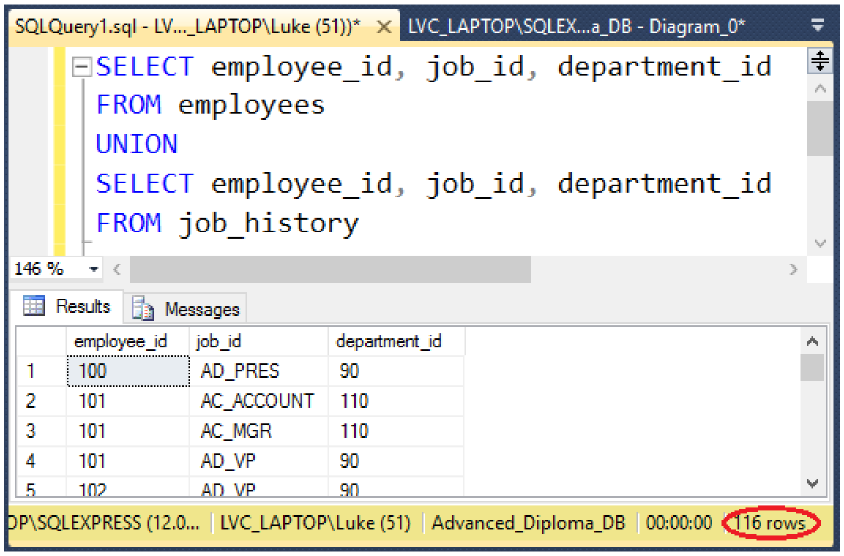
```
   employee_id  job_id
1  101          AC_ACCOUNT
2  200          AC_ACCOUNT
3  206          AC_ACCOUNT
4  101          AC_MGR
5  205          AC_MGR
```

LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 115 rows

Figure 1 - Result of example 1: Applying the UNION operator

NOTE: In the above example the query in the first part returned 107 rows of data and the query in the second part returned 10 rows of data – this is a total of 117 rows of data. Given the fact that the UNION operator include all the rows in the two queries without duplicates, it is possible to have less values while applying this operator – as a matter of fact the last query returned 115 rows.

Example 2: Write a query that will display all the unique employee_id, job_id and department_id values from the employees and job_history tables respectively using UNION



SQLQuery1.sql - LV..._LAPTOP\Luke (51))* | LVC_LAPTOP\SQLEX...a_DB - Diagram_0*

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
```

146 %

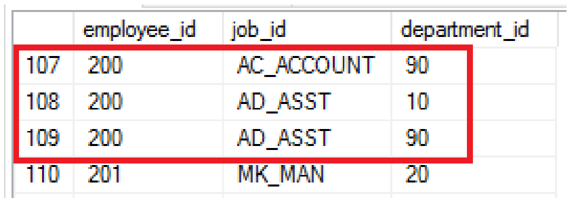
Results | Messages

```
   employee_id  job_id      department_id
1  100          AD_PRES     90
2  101          AC_ACCOUNT  110
3  101          AC_MGR      110
4  101          AD_VP       90
5  102          AD_VP       90
```

DP\SQLEXPRESS (12.0... | LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 116 rows

Figure 2 – Result of example 2: Applying UNION on employee_id, job_id and department_id in employees and job_history

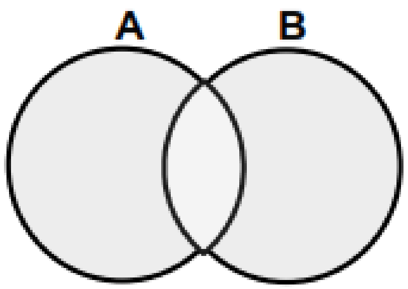
NOTE: The result of the latest query (in example 2) has 1 more row that the result in example 1. This means that there is a person that had the same job_id but worked in a different department. The screenshot below explains this increase – employee 200worked as an AD_ASST both in department 10 and department 90



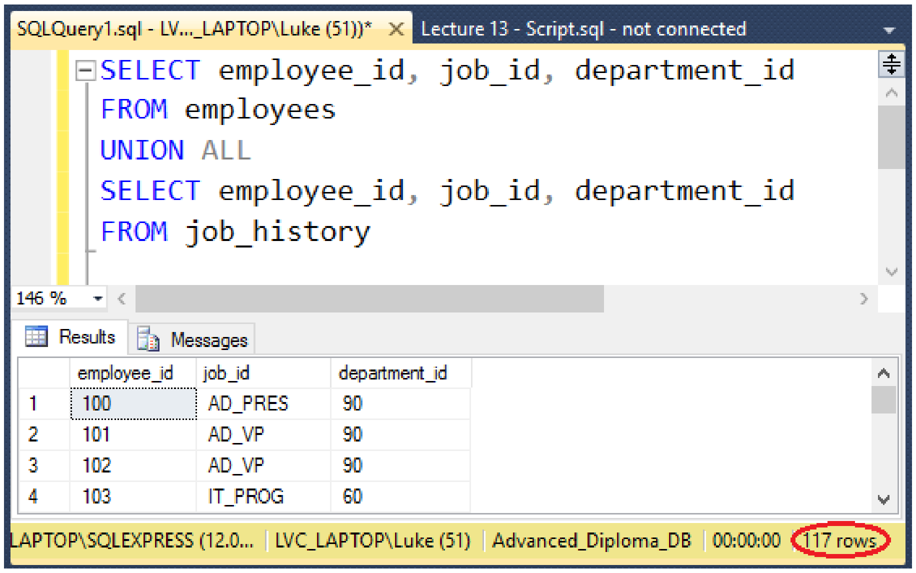
	employee_id	job_id	department_id
107	200	AC_ACCOUNT	90
108	200	AD_ASST	10
109	200	AD_ASST	90
110	201	MK_MAN	20

The UNION ALL operator

The UNION ALL operator is very similar to the previous operator with the only difference being that all the rows form both queries are returned including duplicate values.



Example 3: Write a query that will display all the unique employee_id, job_id and department_id values from the employees and job_history tables respectively using UNION ALL


SQLQuery1.sql - LV..._LAPTOP\Luke (51))*
Lecture 13 - Script.sql - not connected
146 %
Results
Messages

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
```

	employee_id	job_id	department_id
1	100	AD_PRES	90
2	101	AD_VP	90
3	102	AD_VP	90
4	103	IT_PROG	60

LAPTOP\SQLEXPRESS (12.0... | LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 117 rows

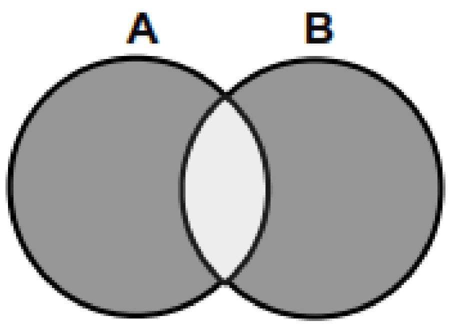
Figure 3 - Result of example 3 - Applying UNION ALL on employee_id, job_id and department_id in employees and job_history

NOTE:

- The difference between the query that was included in example 2 and example 3 is mainly the use of a different set operator – UNION and UNION ALL respectively.
- The use of a different operator results in another difference as well – this difference refers to the number of rows returned. The query in example 2 returns 116 rows as there is one row which is duplicate and this is removed by the UNION operator. On the other hand the query in example 3 returns 117 rows which is the total number of rows from both queries (including duplicate values).
-

The INTERSECT operator

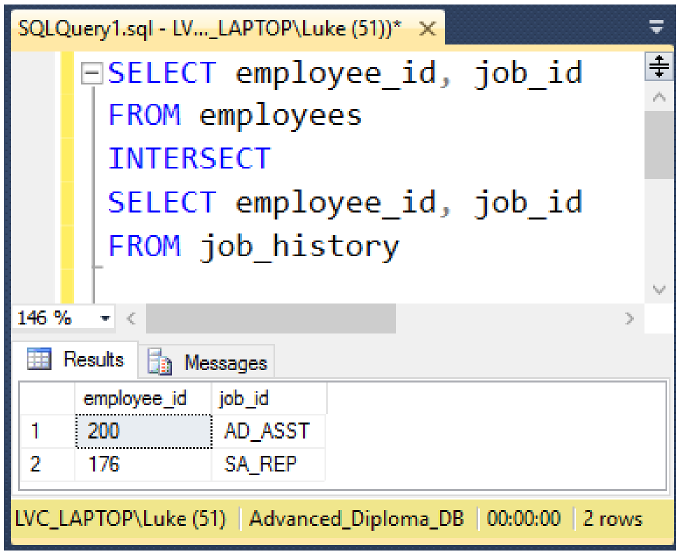
The INTERSECT operator will compare the output of two queries and will return only the common rows.



Important points regarding INTERSECT operator:

- The SELECT statements in the queries should have the same number of columns and their data types should be identical. The only thing that can differ is the name of the columns.
- This operator considers NULL values while checking for duplicate entries
- The order in which the statements are placed does not affect the final result (it remains the same)

Example 4: Write a query that displays the employee number and job number of all the employees whose employee number and job number in the employees table is the same to that found in any entry in the job_history table



SQLQuery1.sql - LV..._LAPTOP\Luke (51))*

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history
```

146 %

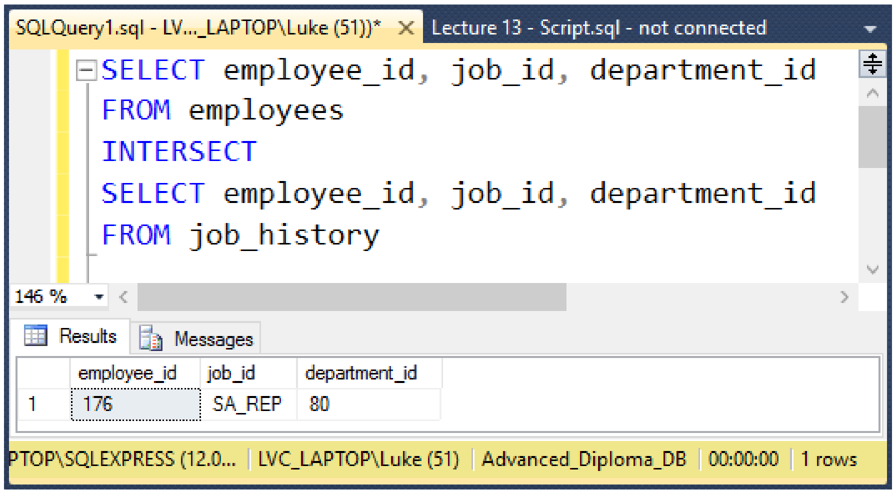
Results | Messages

	employee_id	job_id
1	200	AD_ASST
2	176	SA_REP

LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 2 rows

Figure 4 - Result of example 4 - The rows which have a common employee_id and job_id in both queries

Example 5: Write a query that will return the common values for queries which involve the employee_id, job_id and department_id in the employees and the job_history tables.



SQLQuery1.sql - LV..._LAPTOP\Luke (51))* | Lecture 13 - Script.sql - not connected

```
SELECT employee_id, job_id, department_id
FROM employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM job_history
```

146 %

Results | Messages

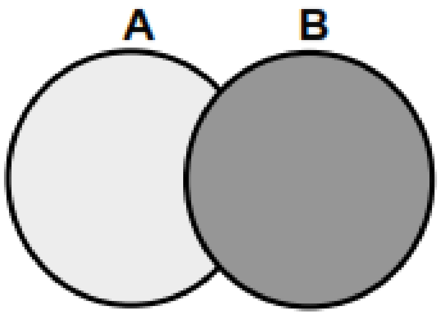
	employee_id	job_id	department_id
1	176	SA_REP	80

PTOP\SQLEXPRESS (12.0... | LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 1 rows

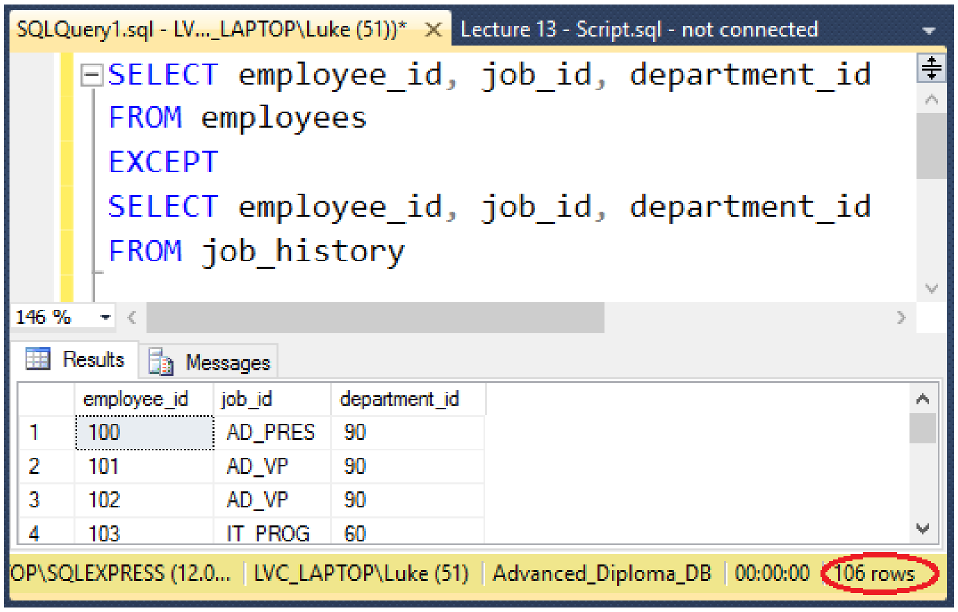
Figure 5 - Result of example 5 – common row in two statements

The EXCEPT operator

The last set operator that will be covered in this topic is the EXCEPT. This operator is returns all the rows that are only present in the first query and not in the second query. Any rows that form part of the second query will not be included in the result. It is therefore understood that the order of the statements is very important in this operator.



Example 5: Write a query that will include all the rows whose employee_id, job_id and department_id is found in the employees table but not in the job_history table. Use a set operator in the process.


SQLQuery1.sql - LV..._LAPTOP\Luke (51))*
Lecture 13 - Script.sql - not connected
146 %
Results
Messages

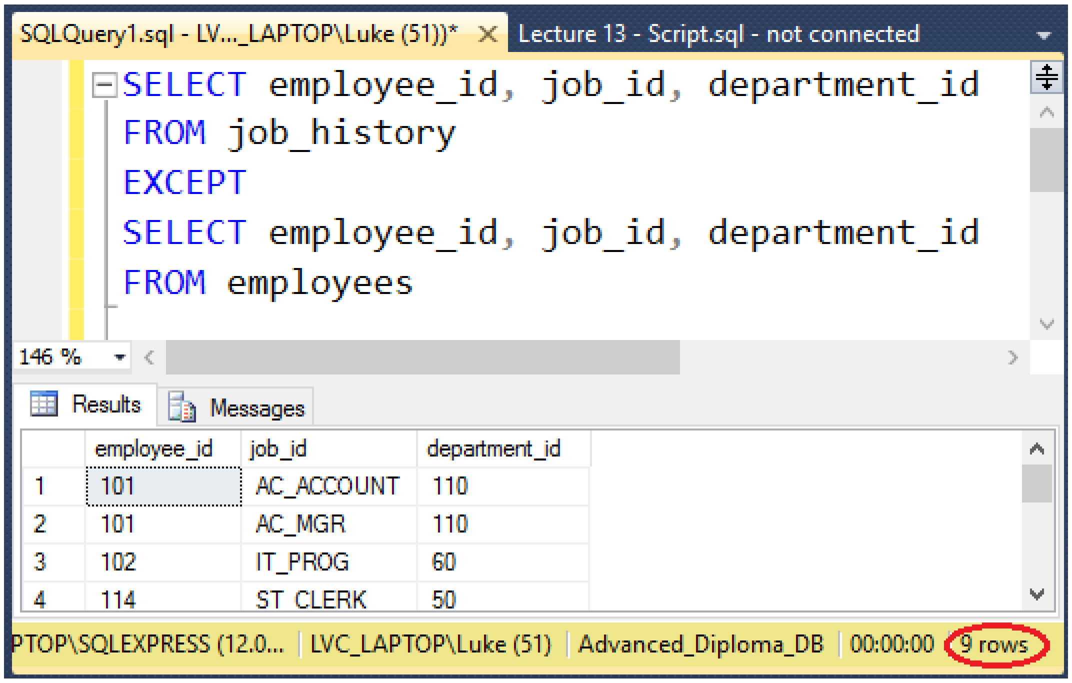
```
SELECT employee_id, job_id, department_id
FROM employees
EXCEPT
SELECT employee_id, job_id, department_id
FROM job_history
```

	employee_id	job_id	department_id
1	100	AD_PRES	90
2	101	AD_VP	90
3	102	AD_VP	90
4	103	IT_PROG	60

```
OP\SQLEXPRESS (12.0... | LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 106 rows
```

Figure 6 - Result of example 6 - The except operator

<u>Example 5</u>: Write a query that will include all the rows whose employee_id, job_id and department_id is found in the job_history table but not in the employees table. Use a set operator in the process.


SQLQuery1.sql - LV..._LAPTOP\Luke (51))*
Lecture 13 - Script.sql - not connected
146 %
Results
Messages

```
SELECT employee_id, job_id, department_id
FROM job_history
EXCEPT
SELECT employee_id, job_id, department_id
FROM employees
```

	employee_id	job_id	department_id
1	101	AC_ACCOUNT	110
2	101	AC_MGR	110
3	102	IT_PROG	60
4	114	ST_CLERK	50

PTOP\SQLEXPRESS (12.0... | LVC_LAPTOP\Luke (51) | Advanced_Diploma_DB | 00:00:00 | 9 rows

Figure 7 – Result of example 7 – The except operator once more

NOTE: The difference between the result in examples 6 and 7 is that the position of the statement has been changed and hence a different result is returned.