

JOBSHEET

PRAKTIKUM BASIS DATA LANJUT

Jurusan Teknologi Informasi
POLITEKNIK NEGERI MALANG
2024



Week 2

SQL SERVER - SELECT, JOIN SORTING DAN FILTERING DATA

Team Teaching:

Habibie Ed Dien, S.Kom., M.T.

Irsyad Arif Mashudi, S.Kom M.Kom

Vit Zuraida, S.Kom., M.Kom.

Rokhimatul Wakhidah, S.Pd., M.T.

Annisa Taufika Firdausi, ST., MT.

Elok Nur Hamdana, S.T., M.T



Information Technology Department, Malang State
Polytechnic

Jobsheet- 1 : Introduction to Transact-SQL and Statements

SELECT, Join, Sorting, and Filtering data

Advanced Database Course

Supervisor: Advanced Database Teaching Team *September*
2024

SAFRIZAL RAHMAN_19_SIB_2G

Topics

1. Introduction to T-SQL and *Query Select*
2. Querying Multiple Tables
3. Sorting and Filtering Data

Objective

Students are expected to be able to:

1. Understanding the basic differences between Transact-SQL (T-SQL) and ANSI SQL.
2. Understanding how to create *a database* from an existing SQL file
3. Understand how to execute part or all of a SQL *script* from an existing file.
4. Understanding the concept of using '*comments*' in T-SQL.
5. Understand the concept of using the SELECT statement to analyze existing tables in *a database* .
6. Understanding how to display data in a *unique / distinct manner* .
7. Understand how to use *ALIAS* for table names and column names.
8. Understand the concept of *CASE* expressions and how to use them.
9. Students understand how to query multiple tables in a SELECT clause using JOIN.
10. Students understand how to write INNER JOIN , OUTER JOIN , SELF-JOIN and CROSS JOIN queries .
11. Students understand how to do Data Sorting , Data Filtering with predicates , Data Filtering with TOP and OFFSET-FETCH
12. Students understand how to handle missing and unknown values in real data.


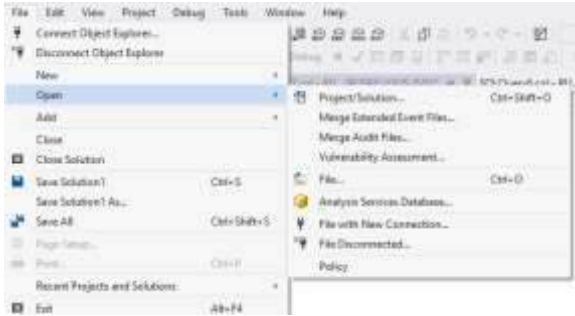
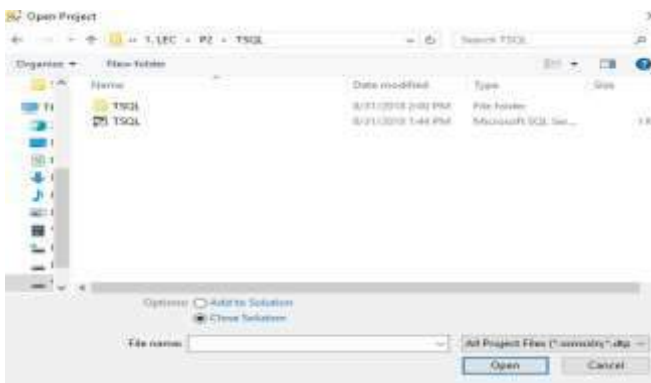
General Instructions

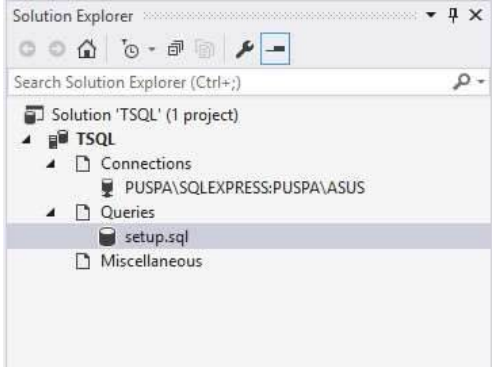
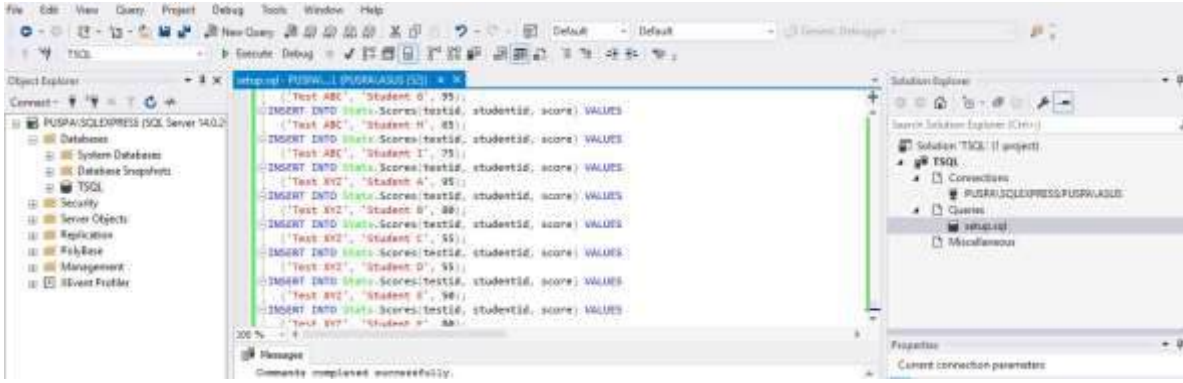
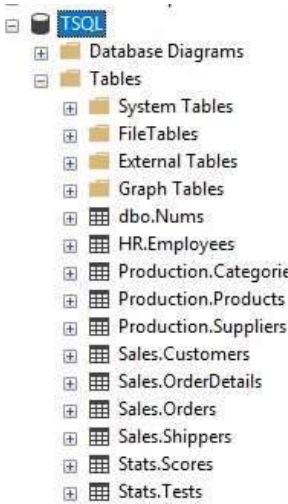
1. Follow the steps in the practical sections in the order given.
2. Answer all questions marked **[Question-X]** that are found in certain steps in each part of the practicum.
3. In each step of the practicum, there is an explanation that will help you answer the questions in instruction number 3, so read and do all the practicum parts in this jobsheet.



4. Write the answers to the questions in the instructions number 3 in a report that is done using a word processing application (Word, OpenOffice, or other similar). Export as a **PDF file** with the following name format:
- **BDL_Class_03_YourFullName .pdf**
 - Collect the PDF files as a practical report to the supervising lecturer.
 - In addition to the file name, also include your identity on the first page of the report.

Practical Preparation: Creating a Database from Existing SQL

Step	Information
1	Create a TSQL database 
2	On the File menu , click Open and click Project/Solution . 
3	In the Open Project file.  window , open the given project

4	<p>Next, the Solution Explorer window will display the following display. Then please open the “Setup” file. This file contains the sql this practicum.</p> 
5	<p>After the setup file is opened, a display like the image below will appear. Then click <i>Execute</i> and please wait until the process is complete.</p>
	
6	<p>After the process is successful, several tables will be formed, as shown in the image below.</p> 



7

For example, to check records in the Sales.Customers table, please execute the command below:

```
USE [TSQL]
GO

SELECT [custid]
      ,[companyname]
      ,[contactname]
      ,[contacttitle]
      ,[address]
      ,[city]
      ,[region]
      ,[postalcode]
      ,[country]
      ,[phone]
      ,[fax]
FROM [Sales].[Customers]
GO
```

8

The results of the SQL command above are as follows

Results		Messages					
	custid	companyname	contactname	contacttitle	address	city	region
1	1	Customer NRZBB	Allen, Michael	Sales Representative	Obere Str. 0123	Berlin	NULL
2	2	Customer MLTDN	Hassall, Mark	Owner	Avda. de la Constitución 5678	México D.F.	NULL
3	3	Customer KBUDE	Peoples, John	Owner	Mataderos 7890	México D.F.	NULL
4	4	Customer HFBZG	Amdt, Torsten	Sales Representative	7890 Hanover Sq.	London	NULL
5	5	Customer HGVLZ	Higginbotham, Tom	Order Administrator	Berguvsvägen 5678	Luleå	NULL



Practical –

Part 1: Executing part or all of a SQL script

Step	Information
1	<p>Please type the following <i>query in your query panel</i> then click <i>execute</i> . Note the results displayed.</p> <pre>SELECT * FROM Sales.Customers;</pre>
2	<p>Please add the</p> <pre>SELECT * FROM Sales.Customers; SELECT custid, companyname, contactname, contacttitle, address, city, region, postalcode, country, phone, fax FROM Sales.Customers;</pre> <p>following <i>query to your query panel</i> then click <i>execute</i> . Note the results</p>

Practical –

Microsoft SQL Server Enterprise Edition

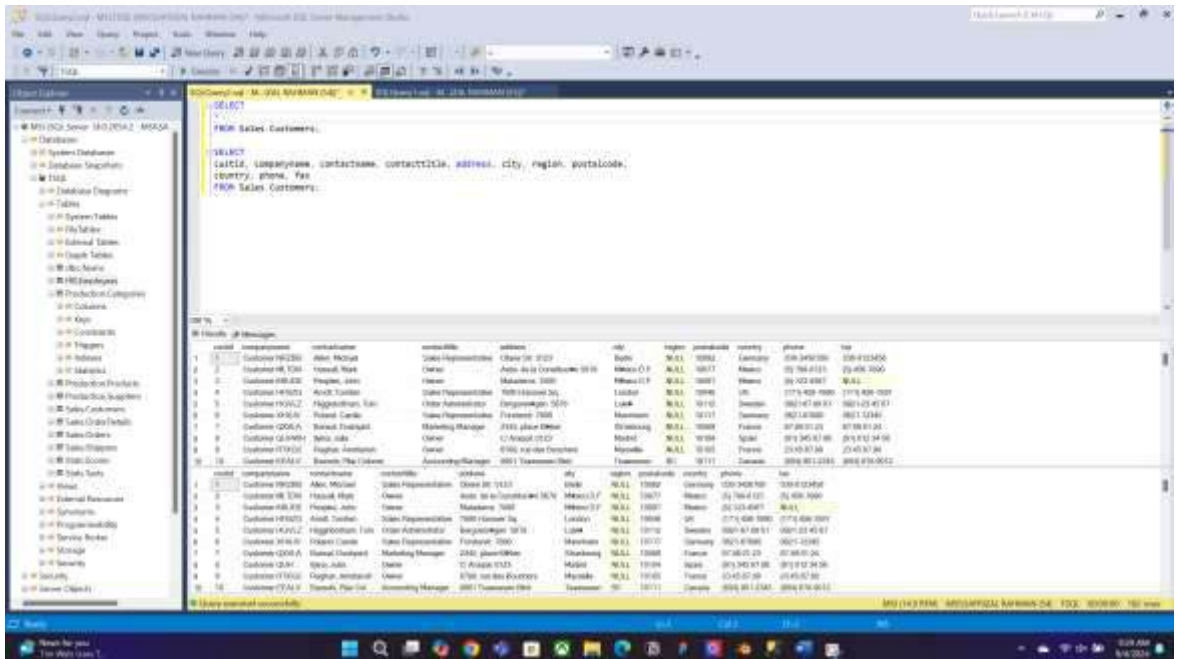
SELECT

FROM Sales.Customers;

Results

id	idempno	contactname	contacttitle	address	city	region	postalcode	country	phone	fax
1	Customer 000001	Adrian, Michael	Sales Representative	Strada 55, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
2	Customer 000002	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
3	Customer 000003	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
4	Customer 000004	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
5	Customer 000005	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
6	Customer 000006	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
7	Customer 000007	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
8	Customer 000008	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
9	Customer 000009	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
10	Customer 000010	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
11	Customer 000011	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
12	Customer 000012	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
13	Customer 000013	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
14	Customer 000014	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
15	Customer 000015	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
16	Customer 000016	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
17	Customer 000017	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
18	Customer 000018	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
19	Customer 000019	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
20	Customer 000020	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001
21	Customer 000021	Alfred, Maria	Sales Representative	Strada 10, 01011	Bucuresti	RO	06000	Romania	(021) 4301000	(021) 4301001

Practical –



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The 'Messages' pane displays the results of a query executed on the 'Sales.Customers' table. The query is as follows:

```
SELECT
custid, companyname, contactname, contacttitle, address, city, region, postalcode,
country, phone, fax
FROM Sales.Customers;
```

The results are displayed in a table with the following columns: custid, companyname, contactname, contacttitle, address, city, region, postalcode, country, phone, fax. The table contains 10 rows of data, including customers from the USA, Canada, and Mexico.

displayed

Make a selection on one of the existing queries then click execute . Note the results displayed. **What is the difference with the results in the second step above? (Question 1)**

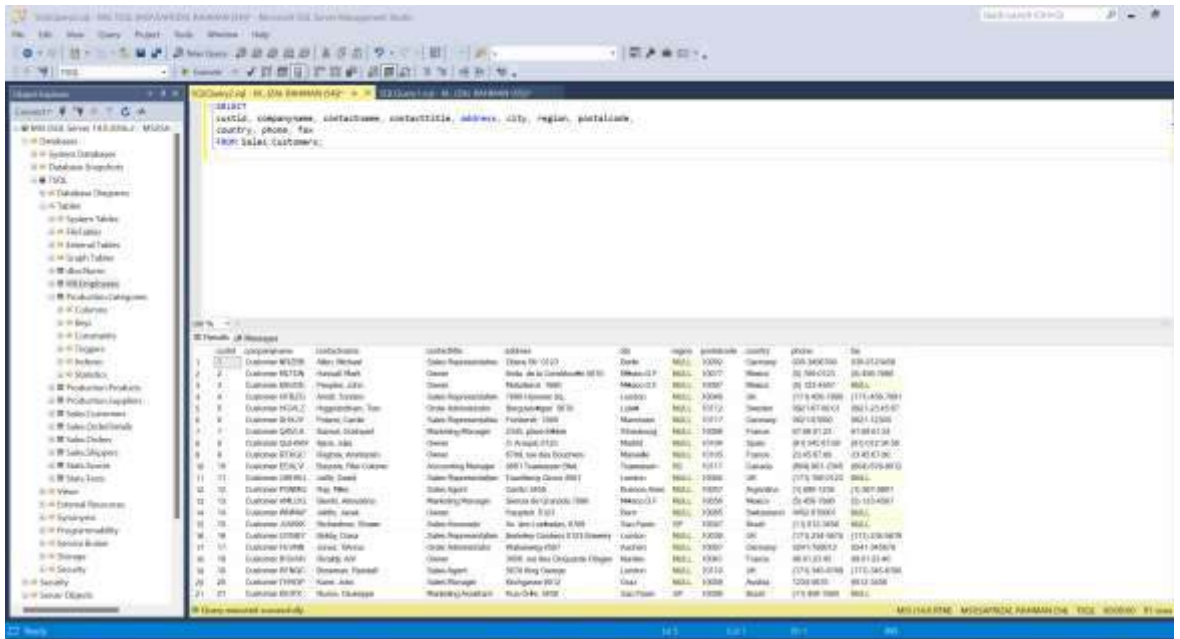
SELECT
custid, companyname, contactname, contacttitle, address, city, region, postalcode,
country, phone, fax
FROM Sales.Customers;

3

```
SELECT
FROM Sales.Customers;

SELECT
custid, companyname, contactname, contacttitle, address, city, region, postalcode,
country, phone, fax
FROM Sales.Customers;
```


Practical –



custid	companyname	contactname	contacttitle	address	city	region	postalcode	country	phone	fax
1	Customer MW10B	Alex Michael	Sales Representative	Elm St 1010	Orlando	FL	32809	USA	(407) 555-1234	(407) 555-5678
2	Customer MW10A	Harold Blank	Owner	Strada de la Libertate 87 B	Milwaukee	WI	53217	USA	(414) 555-1234	(414) 555-5678
3	Customer MW10C	Phyllis Allen	Owner	Marktstrasse 7890	Munich	BY	80335	Germany	(49) 89 123 4567	(49) 89 123 4567
4	Customer MW10D	Arvid Torsson	Sales Representative	1000 E Street SE	London	ON	N6A 3K9	Canada	(416) 456-7890	(416) 456-7890
5	Customer MW10E	Heggenstam, Tom	Order Administrator	Bergstrasse 66 10	Leipzig	SA	04109	Germany	(49) 341 456 789	(49) 341 456 789
6	Customer MW10F	Frederic Coute	Sales Representative	Friedrichstrasse 67	Munich	BY	80333	Germany	(49) 89 123 456	(49) 89 123 456
7	Customer MW10G	Samuel Stuber	Marketing Manager	2500, place Wilfrid	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
8	Customer MW10H	Heidi Zula	Owner	11, rue de la Paix	Montreal	QC	H3Y 1P9	Canada	(514) 354-5678	(514) 354-5678
9	Customer MW10I	Hughes, Antonette	Owner	6789, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
10	Customer MW10J	Heidi Zula	Accounting Manager	6981, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
11	Customer MW10K	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
12	Customer MW10L	Heidi Zula	Sales Agent	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
13	Customer MW10M	Heidi Zula	Marketing Manager	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
14	Customer MW10N	Heidi Zula	Owner	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
15	Customer MW10O	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
16	Customer MW10P	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
17	Customer MW10Q	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
18	Customer MW10R	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
19	Customer MW10S	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
20	Customer MW10T	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
21	Customer MW10U	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
22	Customer MW10V	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
23	Customer MW10W	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
24	Customer MW10X	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
25	Customer MW10Y	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
26	Customer MW10Z	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678
27	Customer MW10A	Heidi Zula	Sales Representative	1000, rue des Bonheurs	Montreal	QC	H3T 1B7	Canada	(514) 354-5678	(514) 354-5678

The two SQL queries you've provided are similar but differ in the level of detail they return:

1. First Query:

sql

SELECT

FROM Sales.Customers;

- This query selects all columns from the Sales.Customers table. The asterisk (*) is a wildcard that tells the database to return every column for each row in the table.

2. Second Query:

sql

SELECT

custid, companyname, contactname, contacttitle, address, city, region, postalcode, country, phone, fax

FROM Sales.Customers;

- This query selects specific columns from the Sales.Customers table. It explicitly lists the columns custid, companyname, contactname, contacttitle, address, city, region, postalcode, country, phone, fax.



Practical –

	<p>postalcode, country, phone, and fax, meaning only these columns will be returned in the result set.</p> <p>Difference in Results:</p> <ul style="list-style-type: none">- The first query (SELECT) will return all columns in the Sales.Customers table, regardless of how many columns the table contains.- The second query will return only the specified columns, which might be fewer than the total number of columns in the table. <p>What to Observe in Execution:</p> <p>When you execute these queries:</p> <ul style="list-style-type: none">- For the first query, you'll see a result set that includes every column available in the Sales.Customers table.- For the second query, you'll see a more focused result set, showing only the columns explicitly listed in the query. <p>This difference is important when you want to limit the data returned, especially if you're only interested in certain attributes of the customers and not the entire dataset.</p>
4	<p>In the query panel please type</p> <pre>SELECT * FROM</pre>
5	<p>then on the Object Explorer tab – Tables please find the Sales.Customers table. Click the table and drag it to the query pane I . The result is as shown below, after that add a semicolon after the name of the table in question and click execute.</p> <pre>SELECT * FROM [Sales].[Customers];</pre>



Practical –

SQLQuery2.sql - M...ZAI RAHMAN (54) * * * SQLQuery1.sql - M...ZAI RAHMAN (55) *

SELECT *
FROM
[Sales].[Customers];

100 %

Results Messages

	custid	companyname	contactname	contacttitle	address	city	region	postalcode	country	phone	fax
1	1	Customer NR286	Allen, Michael	Sales Representative	Oberstr. 1123	Berlin	NULL	10082	Germany	(30) 3456789	(30) 0123456
2	2	Customer M1234	Hassell, Mark	Owner	Avenida de la Constitución 5678	Mexico D.F.	NULL	10077	Mexico	(5) 286-0123	(5) 456-7890
3	3	Customer K81012	Papinen, Johni	Owner	Matadero 7890	Mexico D.F.	NULL	10092	Mexico	(5) 123-4567	NULL
4	4	Customer HFB20	Auerdt, Tordien	Sales Representative	7900 Hanover Sq.	London	NULL	10046	UK	(171) 456-7890	(171) 456-7891
5	5	Customer HFBVLZ	Hjgenbolham, Tora	Order Administrator	Berguvavägen 3678	Luleå	NULL	10112	Sweden	0821-6789 01	0821-23 45 67
6	6	Customer XH4GJV	Polinski, Carole	Sales Representative	Fondriest 7090	Marshall	NULL	10117	Germany	0621-67890	0621-12345
7	7	Customer QXVLA	Barsel, Dushyant	Marketing Manager	2345, place Nôber	Strasbourg	NULL	10098	France	87 89 01 23	87 89 01 24
8	8	Customer CLPWHH	Raisa, Julia	Owner	C/ Anadol 0123	Madrid	NULL	10104	Spain	(91) 345 67 89	(91) 012 34 56

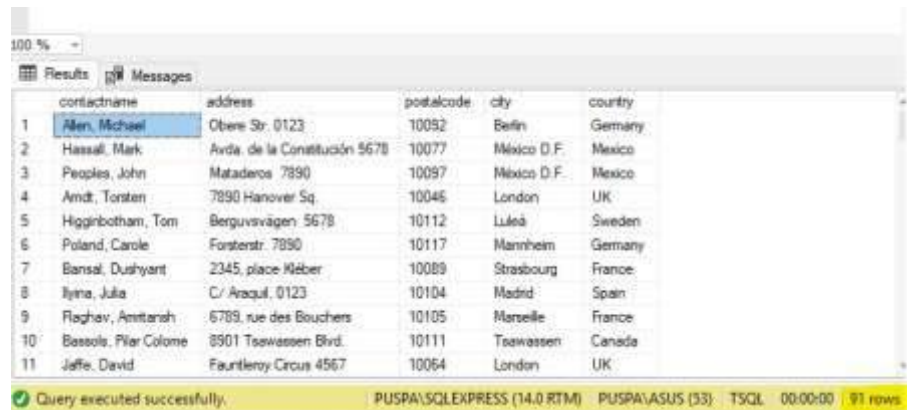
Part 2: Using the SELECT statement for specific columns

Step	Information
1	In the query panel, please type the script below <pre> SELECT contactname, address, postalcode, city, country FROM Sales.Customers; </pre>
2	<i>Highlights query above and click execute</i>

Practical –

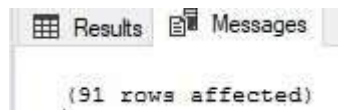
3

Please observe the results. How many *rows* are produced? To find out, you can do it on the results tab as shown in the image below



	contactname	address	postalcode	city	country
1	Allen, Michael	Obere Str. 0123	10092	Berlin	Germany
2	Hassall, Mark	Avenida de la Constitución 5678	10077	México D.F.	Mexico
3	Peoples, John	Mataderos 7890	10097	México D.F.	Mexico
4	Amdt, Torsten	7890 Hanover Sq.	10046	London	UK
5	Hogginbotham, Tom	Berguvsvägen 5678	10112	Luleå	Sweden
6	Poland, Carole	Forsterstr. 7890	10117	Mannheim	Germany
7	Bansal, Dushyant	2345, place Kléber	10089	Strasbourg	France
8	Ilyina, Julia	C/ Araquil, 0123	10104	Madrid	Spain
9	Flaghavi, Amritansh	6789, rue des Bouchers	10105	Marseille	France
10	Bossola, Pilar Colome	8901 Tsewassen Blvd.	10111	Tsewassen	Canada
11	Jaffe, David	Fauntleroy Circus 4567	10064	London	UK

Or you can also go to the messages tab as shown in the image below.



Part 3: Using the SELECT statement to display data *uniquely* / *DISTINCT*

Step	Information
1	<p>In <i>the query</i> panel, please type <i>the script</i> below</p> <pre>SELECT country FROM Sales.Customers;</pre>
2	<p><i>Highlights query</i> above and click <i>execute</i></p>
3	<p>Please observe the results. <i>Is there any duplicate data? If YES, why? Capture the results of executing the SQL script above</i></p>



Practical –

The screenshot shows a SQL query result set with the following data:

	country
45	USA
46	Venezuela
47	Venezuela
48	USA
49	Italy
50	Belgium
51	Canada
52	Germany
53	UK
54	Argentina
55	USA
56	Germany
57	France
58	Mexico
59	Austria
60	Portugal
61	Brazil
62	Brazil
63	Germany
64	Argentina
65	USA
66	Italy
67	Brazil
68	Switzerland
69	Spain
70	Norway
71	USA
72	UK
73	Denmark
74	France
75	USA
76	Belgium
77	USA
78	USA
79	Germany
80	Mexico
81	Brazil
82	USA

Handwritten annotations in red include a box labeled "Duplicate" with arrows pointing to the following rows (country):

- Row 55: USA
- Row 62: Brazil
- Row 65: USA
- Row 71: USA
- Row 77: USA
- Row 78: USA
- Row 82: USA

At the bottom of the screenshot, a status bar indicates: "Query executed successfully."

The query you executed:

sql

```
SELECT country  
FROM Sales.Customers;
```

returns a list of all the countries associated with the customers in the Sales.Customers table. The result you've provided shows the country for each customer record in the table.

What the Query Does:



Practical –

	<p>- SELECT country: This part specifies that you want to retrieve the country column.</p> <p>- FROM Sales.Customers;: This indicates the table from which you want to retrieve the data.</p> <p>Understanding the Output:</p> <p>- Each row in the result corresponds to a country value from a customer record.</p> <p>- The same country might appear multiple times because multiple customers can be from the same country.</p> <p>Additional Considerations:</p> <p>If you want to see a list of unique countries (i.e., each country appearing only once), you can modify the query using DISTINCT:</p> <pre>sql SELECT DISTINCT country FROM Sales.Customers;</pre> <p>This will return each country only once, regardless of how many customers are from that country.</p> <p>If you want to count how many customers are from each country, you could use:</p> <pre>sql SELECT country, COUNT() as customer_count FROM Sales.Customers GROUP BY country;</pre> <p>This query will return each country along with the number of customers from that country.</p>
4	<p>In <i>the query</i> pane, please type <i>the script</i> below.</p> <pre>SELECT DISTINCT country FROM Sales.Customers;</pre> <p>Please click <i>execute</i> and observe the results.</p>

Practical –

*Is there any duplicate data? Explain the difference in results in step 4 and step 3!? What are the benefits of the **DISTINCT** command? Capture the results of executing the SQL script above (Question 3)*

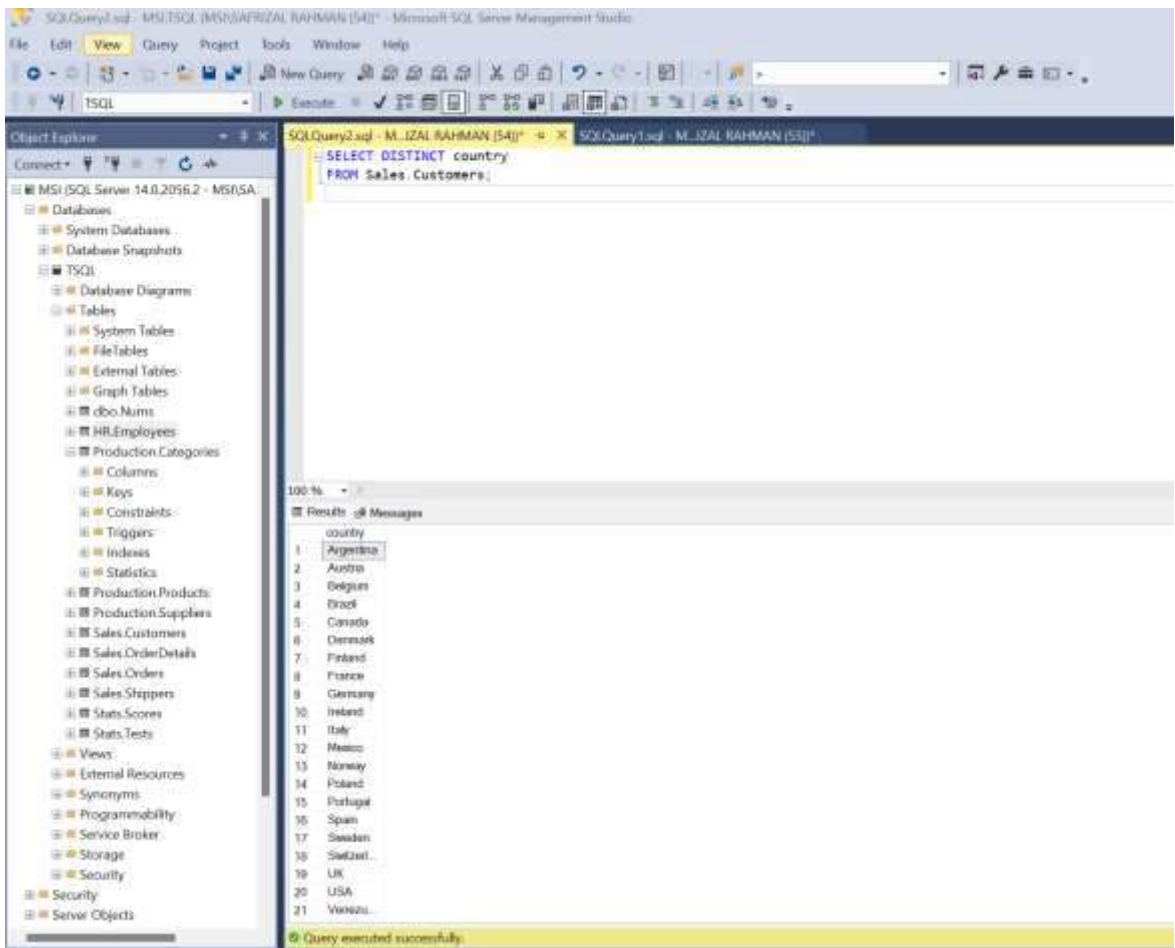
If we use (DISTINCT) returns the unique values of a specified column in the query results.

Each row in the results corresponds to a country value from the customer data.

The same country may appear multiple times because some customers may be from the same country.

Therefore it can be concluded that distinct reduces duplication

5



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'Sales' database expanded, showing tables like 'Sales.Customers'. The right pane shows the 'Query Editor' with the following SQL script:

```
SELECT DISTINCT country
FROM Sales.Customers;
```

The bottom pane shows the 'Results' tab with the following output:

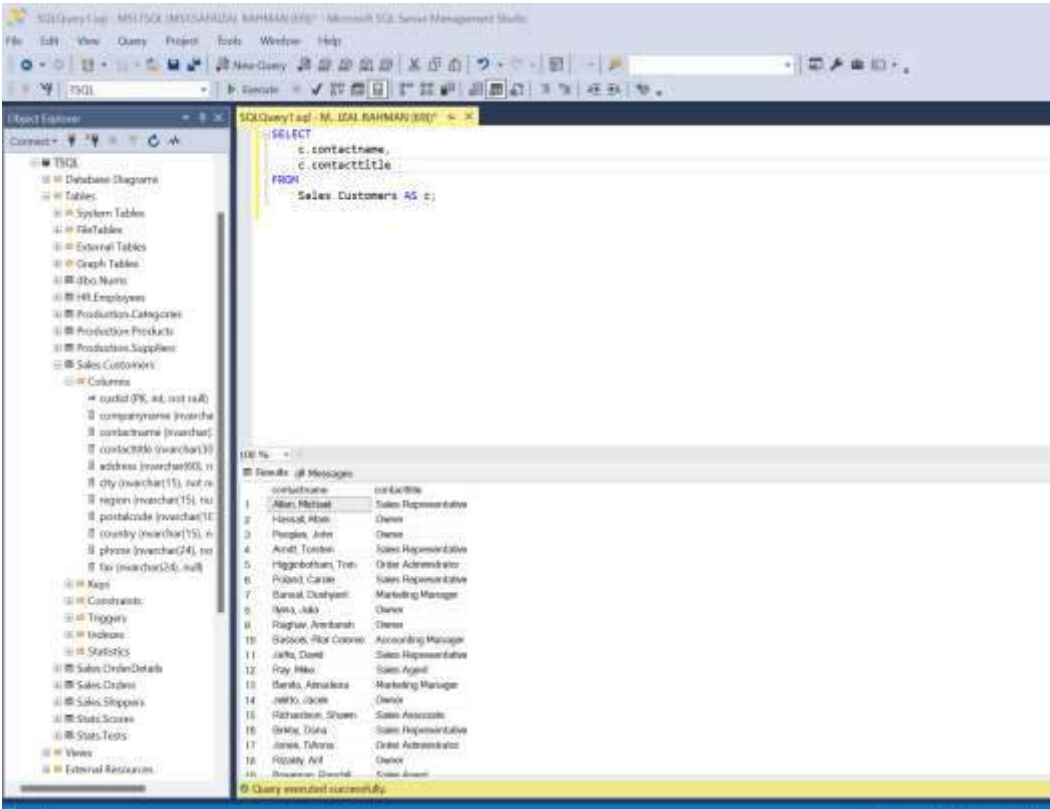
country
1 Argentina
2 Austria
3 Belgium
4 Brazil
5 Canada
6 Denmark
7 Finland
8 France
9 Germany
10 Ireland
11 Italy
12 Mexico
13 Norway
14 Poland
15 Portugal
16 Spain
17 Sweden
18 Switzerland
19 UK
20 USA
21 Venezuela

A status bar at the bottom indicates 'Query executed successfully.'

Practical –

--	--

Part 4: Using *AS* for table names and column names

Step	Information																																								
1	<p>In the query panel, please type the script below</p> <pre>SELECT c.contactname, c.contacttitle FROM Sales.Customers AS c;</pre>																																								
2	<p>Highlights query above and click <i>execute</i>. Observe the results</p>  <table border="1"> <thead> <tr> <th>contactname</th> <th>contacttitle</th> </tr> </thead> <tbody> <tr><td>1. Allen, Michael</td><td>Sales Representative</td></tr> <tr><td>2. Antonio, Maria</td><td>Owner</td></tr> <tr><td>3. Berglund, John</td><td>Owner</td></tr> <tr><td>4. Anelli, Tomoko</td><td>Sales Representative</td></tr> <tr><td>5. Higginbotham, Tom</td><td>Order Administrator</td></tr> <tr><td>6. Poonai, Chand</td><td>Sales Representative</td></tr> <tr><td>7. Bonzel, Danyell</td><td>Marketing Manager</td></tr> <tr><td>8. Ayala, Julia</td><td>Owner</td></tr> <tr><td>9. Rajnar, Anandhar</td><td>Owner</td></tr> <tr><td>10. Gaisvick, Peter</td><td>Accounting Manager</td></tr> <tr><td>11. Jaffe, David</td><td>Sales Representative</td></tr> <tr><td>12. Ray, Mike</td><td>Sales Agent</td></tr> <tr><td>13. Bando, Atsuko</td><td>Marketing Manager</td></tr> <tr><td>14. Jett, John</td><td>Owner</td></tr> <tr><td>15. Richardson, Shawn</td><td>Sales Associate</td></tr> <tr><td>16. Brink, David</td><td>Sales Representative</td></tr> <tr><td>17. Jones, Thomas</td><td>Order Administrator</td></tr> <tr><td>18. Popov, Ivan</td><td>Owner</td></tr> <tr><td>19. Brown, David</td><td>Owner</td></tr> </tbody> </table>	contactname	contacttitle	1. Allen, Michael	Sales Representative	2. Antonio, Maria	Owner	3. Berglund, John	Owner	4. Anelli, Tomoko	Sales Representative	5. Higginbotham, Tom	Order Administrator	6. Poonai, Chand	Sales Representative	7. Bonzel, Danyell	Marketing Manager	8. Ayala, Julia	Owner	9. Rajnar, Anandhar	Owner	10. Gaisvick, Peter	Accounting Manager	11. Jaffe, David	Sales Representative	12. Ray, Mike	Sales Agent	13. Bando, Atsuko	Marketing Manager	14. Jett, John	Owner	15. Richardson, Shawn	Sales Associate	16. Brink, David	Sales Representative	17. Jones, Thomas	Order Administrator	18. Popov, Ivan	Owner	19. Brown, David	Owner
contactname	contacttitle																																								
1. Allen, Michael	Sales Representative																																								
2. Antonio, Maria	Owner																																								
3. Berglund, John	Owner																																								
4. Anelli, Tomoko	Sales Representative																																								
5. Higginbotham, Tom	Order Administrator																																								
6. Poonai, Chand	Sales Representative																																								
7. Bonzel, Danyell	Marketing Manager																																								
8. Ayala, Julia	Owner																																								
9. Rajnar, Anandhar	Owner																																								
10. Gaisvick, Peter	Accounting Manager																																								
11. Jaffe, David	Sales Representative																																								
12. Ray, Mike	Sales Agent																																								
13. Bando, Atsuko	Marketing Manager																																								
14. Jett, John	Owner																																								
15. Richardson, Shawn	Sales Associate																																								
16. Brink, David	Sales Representative																																								
17. Jones, Thomas	Order Administrator																																								
18. Popov, Ivan	Owner																																								
19. Brown, David	Owner																																								

Practical –

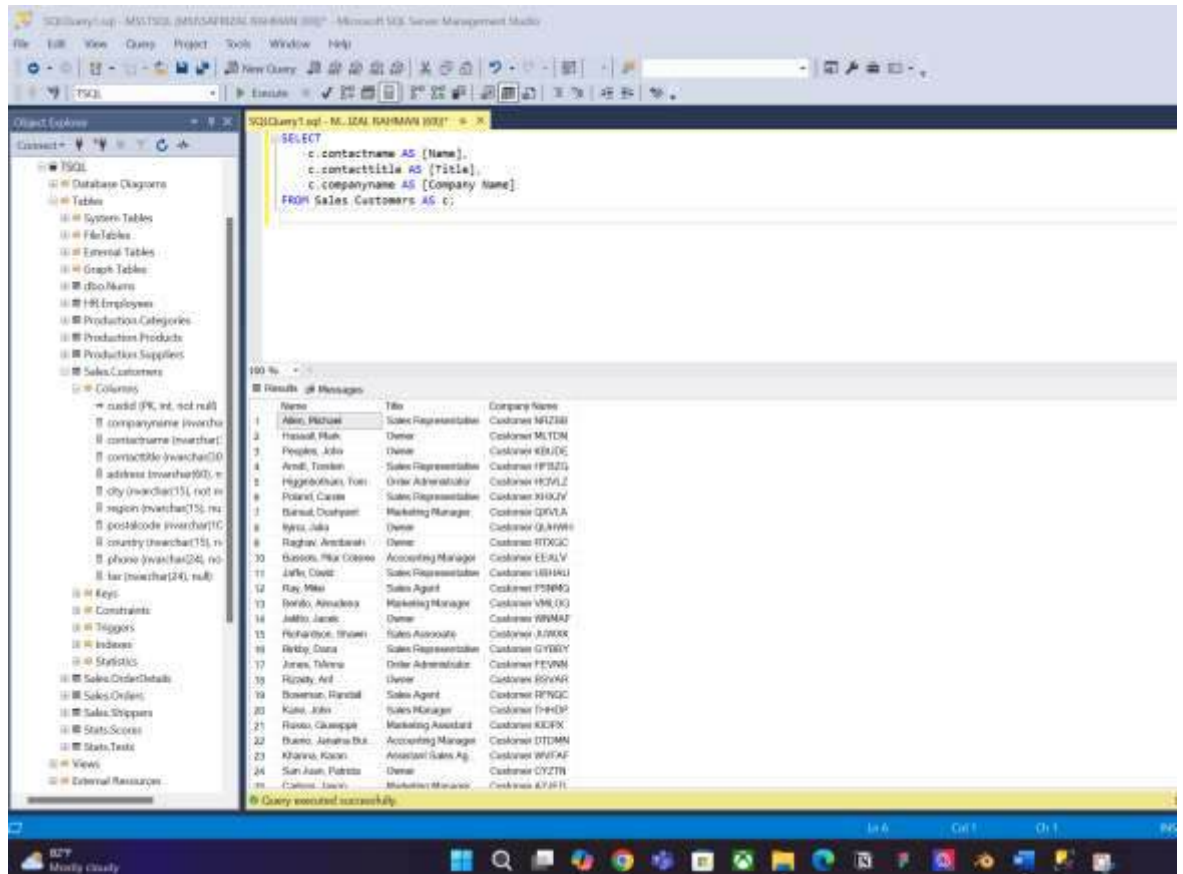
3

In the query panel, please type the script below.

```
SELECT
    c.contactname AS Name, c.contacttitle AS Title, c.companyname AS [Company Name]
FROM Sales.Customers AS c;
```

4

Highlights query above and click execute . Observe the results.



5

What is the difference between the execution results of the query stage 1 and stage 3 above? What are the benefits of the AS command? Please explain! Capture the results of the SQL script execution above (Question 4)

Differences in Execution Results

1. Query Stage 1:

sql



Practical –

```
SELECT  
  c.contactname AS [Name],  
  c.contacttitle AS [Title],  
  c.companyname AS [Company Name]  
FROM Sales.Customers AS c;
```

- Columns Selected: contactname, contacttitle, and companyname.
- Aliases Used: The columns are renamed to Name, Title, and Company Name respectively.
- Table Referenced: Sales.Customers.

Result Example:

Name	Title	Company Name
John Doe	Manager	ABC Corp
Jane Smith	CEO	XYZ Inc

2. Query Stage 3:

```
sql  
SELECT c.contactname, c.companyname  
FROM Customers AS c;
```

- Columns Selected: contactname and companyname.
- Aliases Used: None.
- Table Referenced: Customers.

Result Example:

contactname	companyname
John Doe	ABC Corp
Jane Smith	XYZ Inc

Benefits of the AS Command

The AS command in SQL is used to create aliases for columns or tables. Here are some benefits:



Practical –

1. Readability: Aliases make the output more readable and understandable, especially when column names are long or not user-friendly.
2. Clarity: They help clarify the purpose of the columns in the result set, making it easier for others to understand the data.
3. Convenience: Aliases can simplify complex queries by providing shorter, more meaningful names.

Capturing the Results

Since I can't execute SQL queries directly, I recommend running the provided SQL scripts in your SQL environment to capture the actual results. Here are the scripts again for your reference:

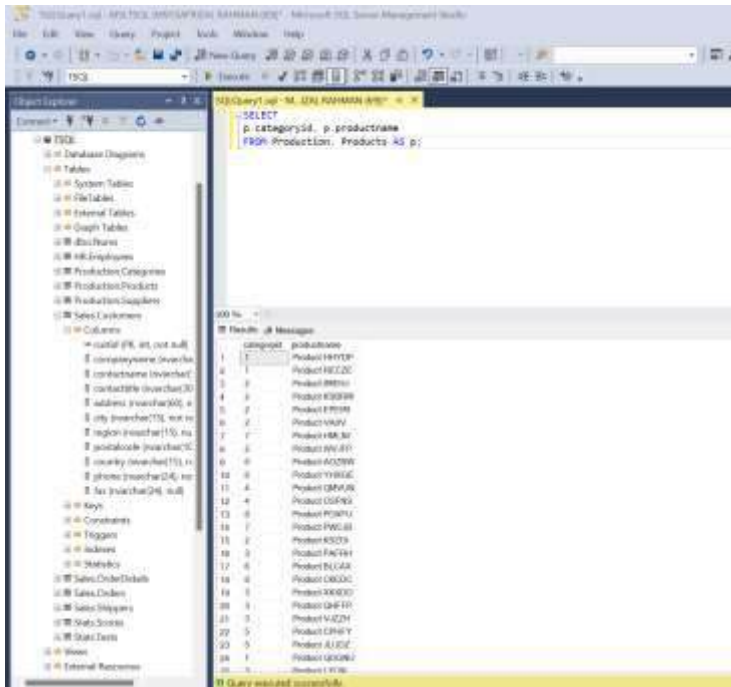
1. Query Stage 1:

```
sql
SELECT
    c.contactname AS [Name],
    c.contacttitle AS [Title],
    c.companyname AS [Company Name]
FROM Sales.Customers AS c;
```

2. Query Stage 3:

```
sql
SELECT c.contactname, c.companyname
FROM Customers AS c;
```

Practicum – Part 5: Use of CASE

Step	Information
1	<p>In the query panel, please type the script below</p> <pre>SELECT p.categoryid, p.productname FROM Production.Products AS p;</pre>
2	<p>Highlights query above and click execute. Observe the results</p> 
3	<p>In the query panel, please type the script below.</p> <pre>SELECT p.categoryid, p.productname, CASE WHEN p.categoryid = 1 THEN 'Beverages' WHEN p.categoryid = 2 THEN 'Condiments' WHEN p.categoryid = 3 THEN 'Confections' WHEN p.categoryid = 4 THEN 'Dairy Products' WHEN p.categoryid = 5 THEN 'Grains/Cereals' WHEN p.categoryid = 6 THEN 'Meat/Poultry' WHEN p.categoryid = 7 THEN 'Produce' WHEN p.categoryid = 8 THEN 'Seafood' ELSE 'Other' END AS categoryname FROM Production.Products AS p;</pre>



4

Highlights query above and click execute . Observe the results.

```
SQLQuery1.sql - M...ZAL RAHMAN (69%) - X
--SELECT
p.categoryid, p.productname,
CASE
WHEN p.categoryid = 1 THEN 'Beverages'
WHEN p.categoryid = 2 THEN 'Condiments'
WHEN p.categoryid = 3 THEN 'Confections'
WHEN p.categoryid = 4 THEN 'Dairy Products'
WHEN p.categoryid = 5 THEN 'Grains/Cereals'
WHEN p.categoryid = 6 THEN 'Meat/Poultry'
WHEN p.categoryid = 7 THEN 'Produce'
WHEN p.categoryid = 8 THEN 'Seafood'
ELSE 'Other'
END AS categoryname
FROM Production.Products AS p;
```

categoryid	productname	categoryname
1	Product HPYDZP	Beverages
2	Product RECZE	Beverages
3	Product IMEHU	Condiments
4	Product KSBRM	Condiments
5	Product EPEIM	Condiments
6	Product VAVV	Condiments
7	Product HMLNI	Produce
8	Product WVJFP	Condiments
9	Product AQZDW	Meat/Poultry
10	Product YHAGE	Seafood
11	Product QMVUN	Dairy Products
12	Product OSPNS	Dairy Products
13	Product PXXFU	Seafood
14	Product PWCJL	Produce
15	Product KSZDI	Condiments
16	Product PAFRH	Confections
17	Product BLCAK	Meat/Poultry
18	Product QNEDC	Seafood

Query executed successfully.

5

What is the difference between the execution results of the query stage 1 and stage 3 above? What are the benefits of the CASE command? Please explain! Capture the results of the SQL script execution above

Differences in Execution Results

1. Query Stage 1:
2. SELECT
3. p.categoryid, p.productname
4. FROM Production.Products AS p;
 - o Columns Selected: categoryid and productname.
 - o Aliases Used: None.
 - o Table Referenced: Production.Products.

Result Example:



categoryid	productname
1	Chai
2	Aniseed Syrup

5. Query Stage 3:

6. SELECT

7. p.categoryid, p.productname,

8. CASE

9. WHEN p.categoryid = 1 THEN 'Beverages'

10. WHEN p.categoryid = 2 THEN 'Condiments'

11. WHEN p.categoryid = 3 THEN 'Confections'

12. WHEN p.categoryid = 4 THEN 'Dairy Products'

13. WHEN p.categoryid = 5 THEN 'Grains/Cereals'

14. WHEN p.categoryid = 6 THEN 'Meat/Poultry'

15. WHEN p.categoryid = 7 THEN 'Produce'

16. WHEN p.categoryid = 8 THEN 'Seafood'

17. ELSE 'Other'

18. END AS categoryname

19. FROM Production.Products AS p;

- o **Columns Selected:** categoryid, productname, and a new column categoryname generated by the CASE statement.

- o **Aliases Used:** categoryname for the result of the CASE statement.

- o **Table Referenced:** Production.Products.

Result Example:

categoryid	productname	categoryname
1	Chai	Beverages
2	Aniseed Syrup	Condiments

Benefits of the CASE Command

The CASE command in SQL is used to implement conditional logic within queries. Here are some benefits:



	<ol style="list-style-type: none">1. Conditional Logic: It allows you to apply different conditions and return specific values based on those conditions, similar to an IF-THEN-ELSE statement in programming.2. Data Transformation: You can transform data dynamically within your query, making it more readable and meaningful.3. Simplifies Complex Queries: It helps in simplifying complex queries by avoiding multiple SELECT statements or JOIN operations.4. Flexibility: It provides flexibility to handle various scenarios directly within the SQL query. <p>Capturing the Results</p> <p>Since I can't execute SQL queries directly, I recommend running the provided SQL scripts in your SQL environment to capture the actual results. Here are the scripts again for your reference:</p> <ol style="list-style-type: none">1. Query Stage 1:2. SELECT3. p.categoryid, p.productname4. FROM Production.Products AS p;5. Query Stage 3:6. SELECT7. p.categoryid, p.productname,8. CASE9. WHEN p.categoryid = 1 THEN 'Beverages'10. WHEN p.categoryid = 2 THEN 'Condiments'11. WHEN p.categoryid = 3 THEN 'Confections'12. WHEN p.categoryid = 4 THEN 'Dairy Products'13. WHEN p.categoryid = 5 THEN 'Grains/Cereals'14. WHEN p.categoryid = 6 THEN 'Meat/Poultry'15. WHEN p.categoryid = 7 THEN 'Produce'16. WHEN p.categoryid = 8 THEN 'Seafood'17. ELSE 'Other'18. END AS categoryname19. FROM Production.Products AS p;
6	In the query panel, please type the script below.



```
SELECT
  p.categoryid, p.productname,
  CASE
    WHEN p.categoryid = 1 THEN 'Beverages'
    WHEN p.categoryid = 2 THEN 'Condiments'
    WHEN p.categoryid = 3 THEN 'Confections'
    WHEN p.categoryid = 4 THEN 'Dairy Products'
    WHEN p.categoryid = 5 THEN 'Grains/Cereals'
    WHEN p.categoryid = 6 THEN 'Meat/Poultry'
    WHEN p.categoryid = 7 THEN 'Produce'
    WHEN p.categoryid = 8 THEN 'Seafood'
    ELSE 'Other'
  END AS categoryname,
  CASE
    WHEN p.categoryid IN (1, 7, 8) THEN 'Campaign Products'
    ELSE 'Non-Campaign Products'
  END AS iscampaign
FROM Production.Products AS p;
```

Please capture the results, what data is obtained from the query command above? Explain (Question 6)

categoryid	productname	categoryname	iscampaign
1	Product 1000001	Beverages	1
2	Product 1000002	Condiments	1
3	Product 1000003	Confections	1
4	Product 1000004	Dairy Products	1
5	Product 1000005	Grains/Cereals	1
6	Product 1000006	Meat/Poultry	1
7	Product 1000007	Produce	1
8	Product 1000008	Seafood	1

Question 6: Capturing the Results and Explanation

The provided query categorizes products by their categoryid and indicates whether they are part of a "Campaign Product" or "Non-Campaign Product." The results obtained from the query include:

Product Category: Identified by the categoryid and translated into readable category names such as 'Beverages', 'Condiments', 'Seafood', etc.

Product Name: The name of the product.

Category Name: This is an alias column created using the CASE statement to map the categoryid to the actual category name.

Is Campaign: Another alias column that determines if a product is a "Campaign Product" based on its categoryid.



Example Result Data:

From the provided data, some rows might look like this:

mathematica

Copy code

1 Product HHYDP Beverages Campaign Products

2 Product IMEHJ Condiments Non-Campaign Products

8 Product YHXGE Seafood Campaign Products

Based on question number 6, please display data that is in the 'seafood' category only and use the *ALIAS* command to change the column name as shown in the image below. **Capture your SQL command and how many rows are produced**

	ID_KATEGORI	NAMA_PRODUK	NAMA_KATEGORI	STATUS
1	8	Product ACRVI	Seafood	Campaign Products
2	8	Product AQOKR	Seafood	Campaign Products
3	8	Product CBRRL	Seafood	Campaign Products
4	8	Product CKEDC	Seafood	Campaign Products
5	8	Product EVFFA	Seafood	Campaign Products
6	8	Product GMKIJ	Seafood	Campaign Products
7	8	Product LYERX	Seafood	Campaign Products
8	8	Product POXFU	Seafood	Campaign Products
9	8	Product TTEEX	Seafood	Campaign Products

Question 7: Filtering Data for 'Seafood' Category Only and Renaming Columns

To filter only the 'Seafood' category and use the alias command to rename the columns, the SQL query would look like this:

8

sql

SELECT

p.categoryid AS Category_ID,

p.productname AS Product_Name,

CASE

WHEN p.categoryid = 8 THEN 'Seafood'

END AS Category_Name,

CASE

WHEN p.categoryid IN (1, 7, 8) THEN 'Campaign Products'

ELSE 'Non-Campaign Products'


END AS Is_Campaign

FROM

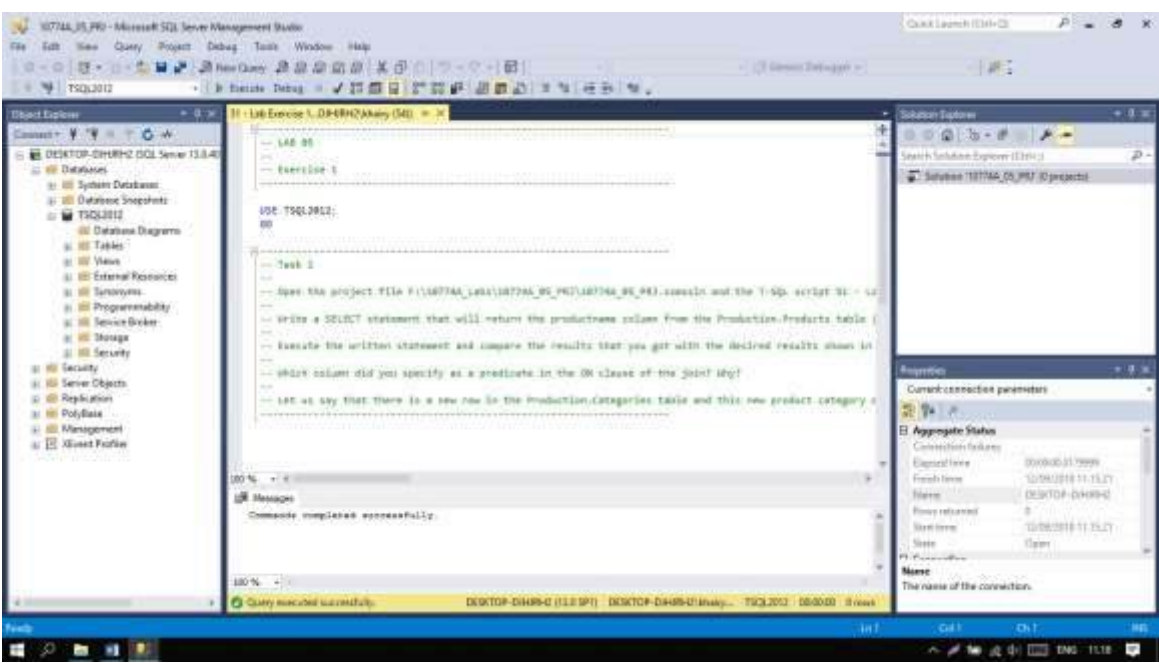
Production.Products AS p

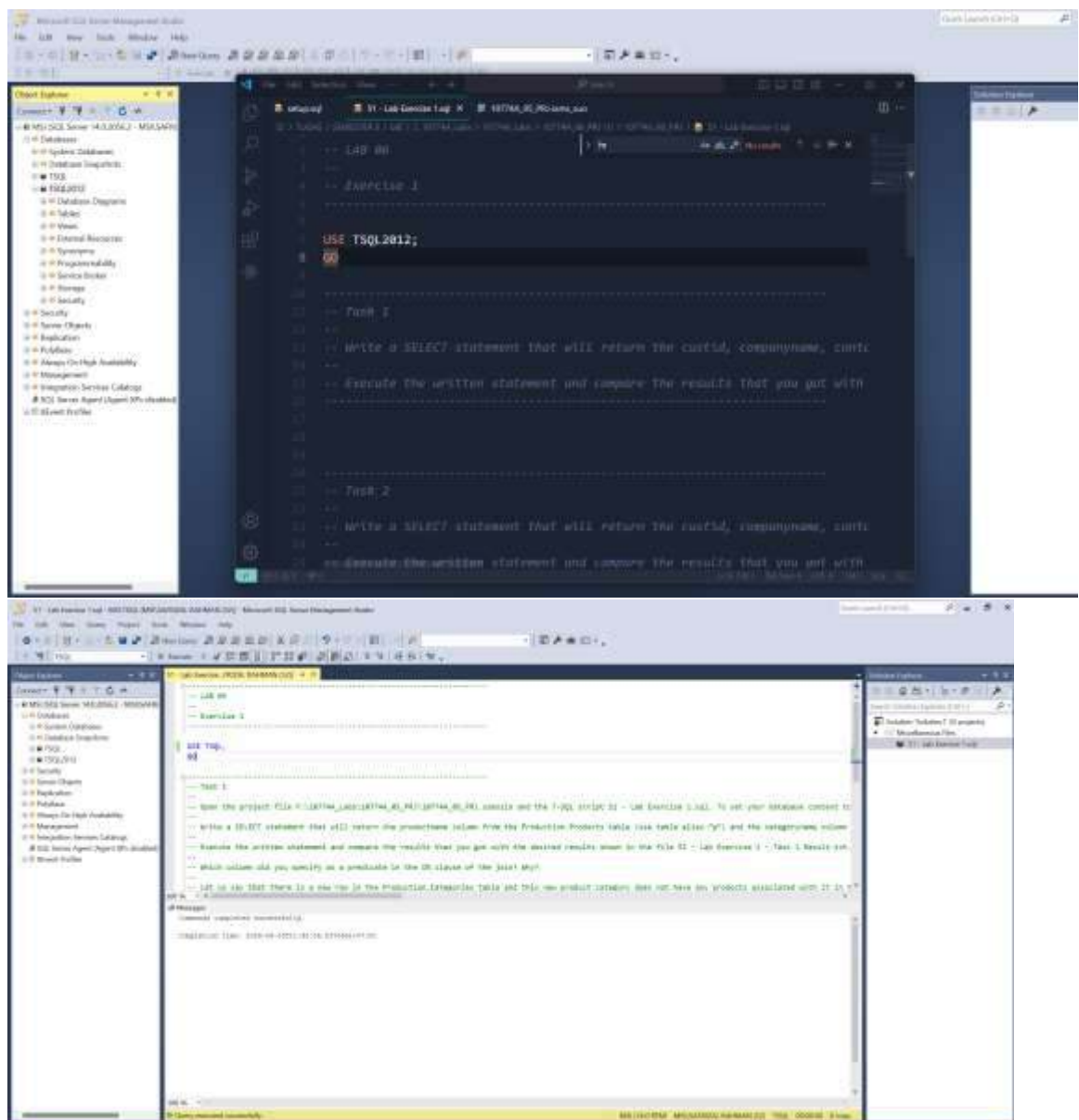
WHERE



	
--	--

Practical – Part 6 : Creating an Inner Join Query

Step	Information
1	<p>To experiment on this jobsheet, first log in to SQL Server Management Studio (SSMS). Then open the project \10774A Labs\10774A_05_PRJ\10774A_05_PRJ.ssmssln and the T-SQL script 51 - Lab Exercise 1.sql. Make sure the database is connected to “TSQL”.</p>
	



2

[Question- 9] Write a T-SQL SELECT that will display the productname column from the Production.Products table (use the alias table "p") and the categoryname column from the Production.Categories table (use the alias table "c") using inner join.
SELECT p.productname, c.categoryname
FROM Production.Products AS p
INNER JOIN Production.Categories AS c ON p.categoryid = c.categoryid;



3

Compare the results in step 2 with the file 52 - Lab Exercise 1 - Task 1 Result.txt. If they are the same then T-SQL wrote correct.

52 - Lab Exercise 1 - Task 1 Result.txt		51 - Lab Exercise 1...RI-PC\TOSHIBA (52))	
productname		categoryname	
Product HHYDP		Beverages	
Product RECZE		Beverages	
Product IMEHJ		Condiments	
...			
...			
...			
Product BWRLG		Beverages	
Product JYGFE		Beverages	
Product LUNZZ		Condiments	
(77 row(s) affected)			



Results Messages

	productname	categoryname
1	Product HHYDP	Beverages
2	Product RECZE	Beverages
3	Product IMEHJ	Condiments
4	Product KSBRM	Condiments
5	Product EPEIM	Condiments
6	Product VAIIV	Condiments
7	Product HMLNI	Produce
8	Product WVJFP	Condiments
9	Product AOZBW	Meat/Poultry
10	Product YHXGE	Seafood
11	Product QMVUN	Dairy Products
12	Product OSFNS	Dairy Products
13	Product POXFU	Seafood
14	Product PWCJB	Produce
15	Product KSZOI	Condiments
16	Product PAFRH	Confections
17	Product BLCAX	Meat/Poultry
18	Product CKEDC	Seafood
19	Product YKXDC	Confections

4

[Question- 10] Which column is specified as a predicate in the ON join clause? Why?

```
SELECT p.productname, c.categoryname
```

```
FROM Production.Products AS p
```

```
INNER JOIN Production.Categories AS c ON p.categoryid = c.categoryid;
```

Common Key: The categoryid column is a common key that exists in both the Production.Products table and the Production.Categories table.

Relationship: It represents the relationship between products and their categories. Each product has a categoryid that links it to a specific category in the Categories table.

Data Integrity: Using categoryid ensures that the join operation correctly matches each product with its corresponding category, maintaining data integrity and providing meaningful results.





5

Conclusion : After carrying out this part of the practicum, students know and understand how to perform an INNER JOIN on two tables.



Practical – Part

7 : Creating an Inner Join Query on Multiple Tables

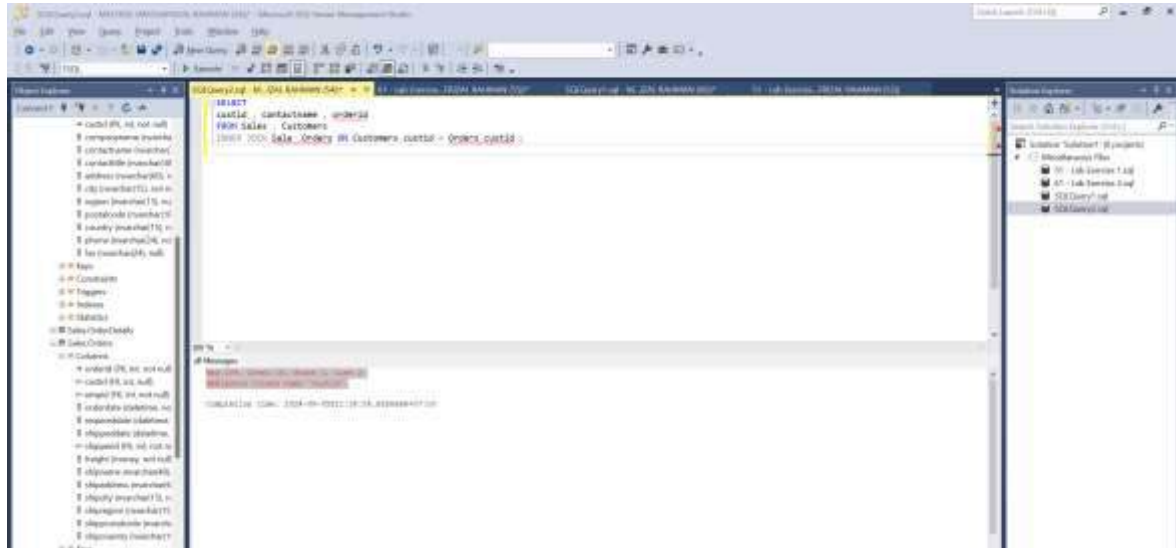
Step	Information
1	<p>A <i>developer</i> will often be asked to run T-SQL files obtained from various departments . For example, the sales department wants a sales report of all customers for at least one order , with detailed information about each order. Then <i>the developer</i> will prepare the initialization of the SELECT statement to retrieve the custid and contactname columns in the Sales.Orders table. In accordance with the case study, this part 2 practicum will be carried out.</p> <p>Open the project \10774A Labs\10774A_05_PRJ\10774A_05_PRJ.ssmssln and the T-SQL script 61 - Lab Exercise 2.sql. Make sure the database is connected with “TSQL”.</p>  

The developer will write T-SQL:

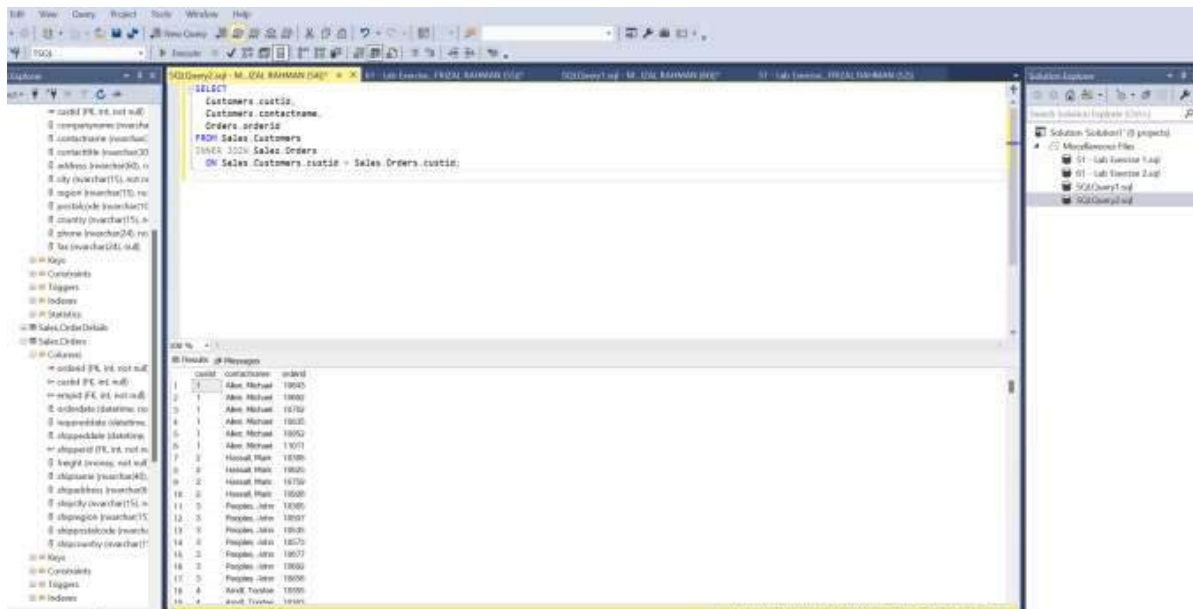
```
SELECT
custid , contactname , orderid
FROM Sales . Customers
INNER JOIN Sales . Orders ON Customers . custid = Orders . custid ;
```

Execute the T-SQL , and observe the results!

2



NO error



3

[Question- 11] After the 2nd stage of the experiment is carried out, an error will appear. What is the content of the error message? Why can this error occur? Explain!

The error message will be "Ambiguous column name 'custid'". This error happens because both the Sales.Customers and Sales.Orders tables have a custid column. SQL Server doesn't know which one to



Practical – Part

use in the SELECT query since both tables are part of the JOIN. Thus, you need to explicitly reference the table from which you want to retrieve custid by using the table name or alias.

[Question- 12] In this 4th trial, fix the error that occurred in the 3rd stage trial which explains that all table names have their own table identities.

In the 4th stage, you resolved the ambiguity in the custid column by explicitly referring to the table names in the SELECT query.

SELECT

Customers.custid,

Customers.contactname,

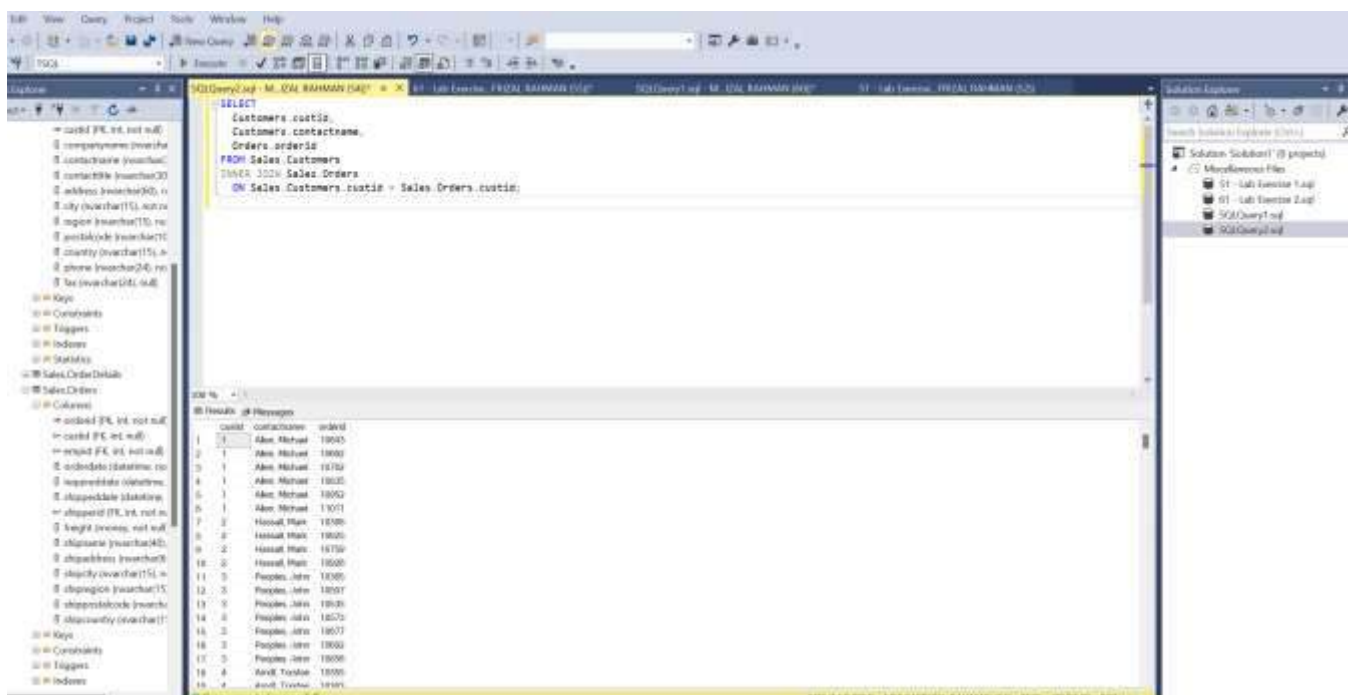
Orders.orderid

FROM Sales.Customers

INNER JOIN Sales.Orders

ON Sales.Customers.custid = Sales.Orders.custid;

4



5

Observe and compare the results of the 4th stage trial with the file 62 - Lab Exercise 2 - Task 2 Result.txt. If the results are the same, then your answer is correct.

Team Teaching Advanced Database



62 - Lab Exercise 2 - Task 2 Result.txt x 61 - Lab Exercise 2...RI-PC\TOSHIBA (52) *

custid	contactname	orderid
1	Allen, Michael	10643
1	Allen, Michael	10692
1	Allen, Michael	10702
...		
...		
91	Conn, Steve	10906
91	Conn, Steve	10998
91	Conn, Steve	11044

(830 row(s) affected)

00 %

Results Messages

	custid	contactname	orderid
1	1	Allen, Michael	10643
2	1	Allen, Michael	10692
3	1	Allen, Michael	10702
4	1	Allen, Michael	10835
5	1	Allen, Michael	10952
6	1	Allen, Michael	11011
7	2	Hassall, Mark	10308
8	2	Hassall, Mark	10625
9	2	Hassall, Mark	10759
10	2	Hassall, Mark	10926
11	3	Peoples, John	10365
12	3	Peoples, John	10507
13	3	Peoples, John	10535
14	3	Peoples, John	10573
15	3	Peoples, John	10677
16	3	Peoples, John	10682
17	3	Peoples, John	10856
18	4	Arndt, Torsten	10355
19	4	Arndt, Torsten	10383

2 Query executed successfully.


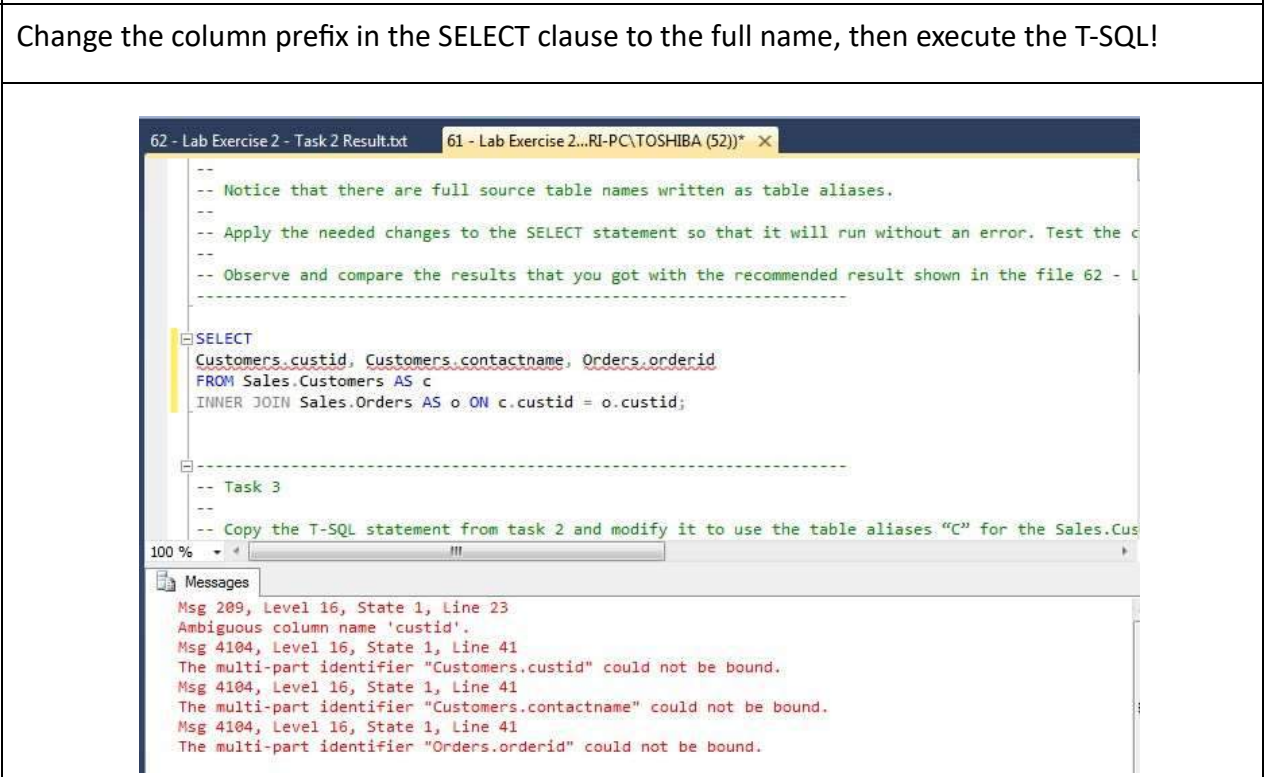
6

[Question- 13] Copy the T-SQL in the 4th stage of the test and modify it by using the alias table " c " to Sales.Customers table and " o " for Sales.Orders table.

```
sql
Copy code
SELECT
    c.custid,
    c.contactname,
    o.orderid
FROM Sales.Customers AS c
INNER JOIN Sales.Orders AS o
    ON c.custid = o.custid;
```

This query now uses table aliases c for Sales.Customers and o for Sales.Orders, which makes the query more concise and easier to read.

Practical – Part

	
7	<p>Execute T-SQL on stage-6 test and compare the result with the result of stage 4 execution! If the result is the same then your T-SQL is correct.</p>
8	<p>Change the column prefix in the SELECT clause to the full name, then execute the T-SQL!</p> 
9	<p>[Question- 14] Why does the execution result of T-SQL stage 8 produce an error?</p> <ol style="list-style-type: none"> Ambiguous Column Name: <ul style="list-style-type: none"> Error message "Ambiguous column name 'custid'" suggests that the column custid exists in both Sales.Customers and Sales.Orders tables, and without further clarification, SQL does not know which table you are referring to.



	<p>2. Multi-part Identifier Error:</p> <ul style="list-style-type: none"> o The error "The multi-part identifier 'Customers.custid' could not be bound" means that SQL Server cannot resolve the reference to the Customers.custid, Customers.contactname, and Orders.orderid because the table Customers and Orders are aliased as c and o, respectively. <p>To fix this:</p> <ol style="list-style-type: none"> 1. Change the column references in the SELECT clause to use the table aliases c and o for Customers and Orders, as you did in the FROM clause. You should update your query as follows: <pre>sql Copy code SELECT c.custid, c.contactname, o.orderid FROM Sales.Customers AS c INNER JOIN Sales.Orders AS o ON c.custid = o.custid;</pre> <ol style="list-style-type: none"> 2. This query should execute without any errors because the column references in the SELECT clause now correctly correspond to the aliases provided in the FROM clause. The error occurred because the columns were referenced with full table names (Customers and Orders) instead of their aliases (c and o) defined in the FROM clause. SQL Server could not bind those full table names as no such references existed in the query.
10	<p>[Question- 15] Change the column name prefix in the T-SQL test step 8 with its alias name, then display the execution results!</p> <p>You need to replace the full table names with their corresponding aliases (as shown in the corrected query above), and then re-run the query to get the expected result.</p>
11	<p>Conclusion : After carrying out this part of the practicum, you should now know and understand the importance of using table alias names and how to JOIN multiple tables (more than two tables).</p>

8 : Creating a Self-Join Query

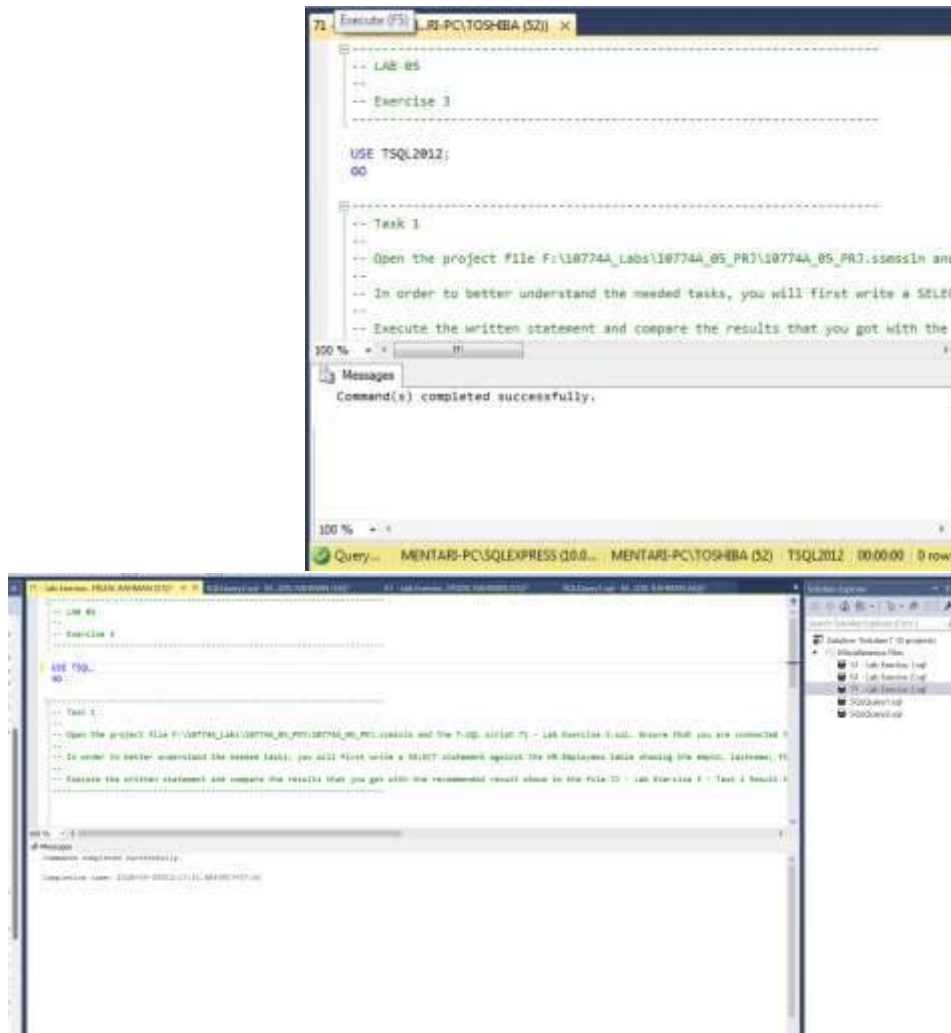
Step	Information
------	-------------

Practical – Part

This practicum uses a case study in an HR department that wants to display reports on employees and managers. Some of the things that want to be displayed are the lastname, firstname, and title columns of the HR.Employees table for employees and managers.

Open the project \10774A Labs\10774A_05_PRJ\10774A_05_PRJ.ssmssln and the T-SQL script 71 - Lab Exercise 3.sql. Make sure the database is connected with “TSQL”.

1



2

[Question- 16] Write T-SQL using SELECT clause to display empid, lastname, firstname, title, and mgrid columns. on the table HR.Employees by giving the alias name “e” for the HR.Employees table.

To display the columns empid, lastname, firstname, title, and mgrid from the HR.Employees table using the alias e, you can write the T-SQL SELECT query as follows:

```
sql
Copy code
SELECT
    e.empid,
    e.lastname,
    e.firstname,
```



```
e.title,  
e.mgrid  
FROM  
HR.Employees AS e;
```

This query will retrieve the specified columns from the HR.Employees table with the alias e for the table, making the query easier to read, especially when joining multiple tables or working with large datasets.

The screenshot shows the SQL Developer interface. The top pane displays a SQL query: `SELECT e.empid, e.lastname, e.firstname, e.title, e.mgrid FROM HR.Employees AS e;`. The bottom pane shows the query results in a table with 5 columns: empid, lastname, firstname, title, and mgrid. The results list 14 employees, including names like 'Muller', 'Pyne', 'Liao', 'Hart', 'Mark', 'Tobias', 'King', 'Coble', and 'Chapman', along with their titles and manager IDs.

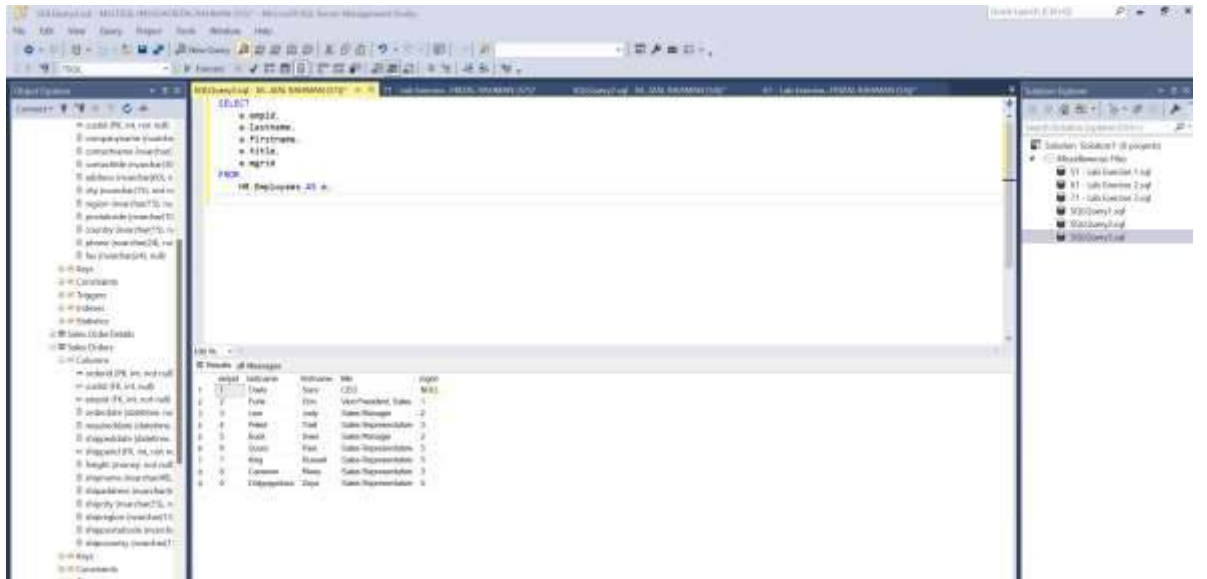
empid	lastname	firstname	title	mgrid
1	Muller	Steven	HR Representative	100
2	Pyne	John	HR Representative	100
3	Liao	John	HR Representative	100
4	Hart	John	HR Representative	100
5	Mark	Mark	HR Representative	100
6	Tobias	Paul	HR Representative	100
7	King	David	HR Representative	100
8	Coble	Michael	HR Representative	100
9	Chapman	David	HR Representative	100
10				
11				
12				
13				
14				

[Question- 17] Execute the 2nd stage of the test and compare it with 72 - Lab Exercise 3 - Task 1 Result.txt . If the results are the same, then your test is correct.

3

```
SELECT  
e.empid,  
e.lastname,  
e.firstname,  
e.title,  
e.mgrid  
FROM  
HR.Employees AS e;
```

Practical – Part

	 <p>The screenshot shows the SQL Server Enterprise Manager interface. A query window is open with the following SQL code:</p> <pre>SELECT e.empid, e.lastname, e.firstname, e.title, e.mgrid FROM HR.Employees AS e</pre> <p>The results grid below the query shows the following data:</p> <table><tr><th>empid</th><th>lastname</th><th>firstname</th><th>title</th><th>mgrid</th></tr><tr><td>1</td><td>Davis</td><td>Jane</td><td>CEO</td><td>NULL</td></tr><tr><td>2</td><td>Furke</td><td>John</td><td>Vice President, Sales</td><td>1</td></tr><tr><td>3</td><td>Lee</td><td>Andy</td><td>Sales Manager</td><td>2</td></tr><tr><td>4</td><td>Patel</td><td>Neal</td><td>Sales Representative</td><td>3</td></tr><tr><td>5</td><td>Arch</td><td>Dean</td><td>Sales Manager</td><td>2</td></tr><tr><td>6</td><td>Quinn</td><td>Pam</td><td>Sales Representative</td><td>5</td></tr><tr><td>7</td><td>King</td><td>Robert</td><td>Sales Representative</td><td>6</td></tr><tr><td>8</td><td>Chen</td><td>Renee</td><td>Sales Representative</td><td>3</td></tr><tr><td>9</td><td>Edwards</td><td>Steph</td><td>Sales Representative</td><td>6</td></tr></table>	empid	lastname	firstname	title	mgrid	1	Davis	Jane	CEO	NULL	2	Furke	John	Vice President, Sales	1	3	Lee	Andy	Sales Manager	2	4	Patel	Neal	Sales Representative	3	5	Arch	Dean	Sales Manager	2	6	Quinn	Pam	Sales Representative	5	7	King	Robert	Sales Representative	6	8	Chen	Renee	Sales Representative	3	9	Edwards	Steph	Sales Representative	6
empid	lastname	firstname	title	mgrid																																															
1	Davis	Jane	CEO	NULL																																															
2	Furke	John	Vice President, Sales	1																																															
3	Lee	Andy	Sales Manager	2																																															
4	Patel	Neal	Sales Representative	3																																															
5	Arch	Dean	Sales Manager	2																																															
6	Quinn	Pam	Sales Representative	5																																															
7	King	Robert	Sales Representative	6																																															
8	Chen	Renee	Sales Representative	3																																															
9	Edwards	Steph	Sales Representative	6																																															
4	<p>[Question- 18] Copy the T-SQL in step 2 then modify it by adding columns about manager information, namely lastname, firstname using SELF-JOIN. Use the aliases mgrlastname and mgrfirstname to distinguish the names of managers and employees.</p> <pre>SELECT e.empid, e.lastname, e.firstname, e.title, e.mgrid, m.lastname AS mgrlastname, m.firstname AS mgrfirstname FROM HR.Employees AS e LEFT JOIN HR.Employees AS m ON e.mgrid = m.empid;</pre>																																																		



```
SQLQuery4.sql - M...IZAL RAHMAN (61))+ X SQLQuery3.sql - M...IZAL RAHMAN (51))+ 71 - Lab Exercise...FRIZAL RAHMAN (
SELECT
    e.empid,
    e.lastname,
    e.firstname,
    e.title,
    e.mgrid,
    m.lastname AS mgrlastname,
    m.firstname AS mgrfirstname
FROM
    HR.Employees AS e
LEFT JOIN
    HR.Employees AS m ON e.mgrid = m.empid;
```

100 %

Results Messages

	empid	lastname	firstname	title	mgrid	mgrlastname	mgrfirstname
1	1	Davis	Sara	CEO	NULL	NULL	NULL
2	2	Funk	Don	Vice President, Sales	1	Davis	Sara
3	3	Lew	Judy	Sales Manager	2	Funk	Don
4	4	Peled	Yael	Sales Representative	3	Lew	Judy
5	5	Buck	Sven	Sales Manager	2	Funk	Don
6	6	Suurs	Paul	Sales Representative	5	Buck	Sven
7	7	King	Russell	Sales Representative	5	Buck	Sven
8	8	Cameron	Maria	Sales Representative	3	Lew	Judy
9	9	Dolgopyatova	Zoya	Sales Representative	5	Buck	Sven

This query will display the employee's data along with the manager's name (mgrlastname and mgrfirstname). It uses a **LEFT JOIN** because some employees (like the CEO) may not have a manager (mgrid is NULL).

5

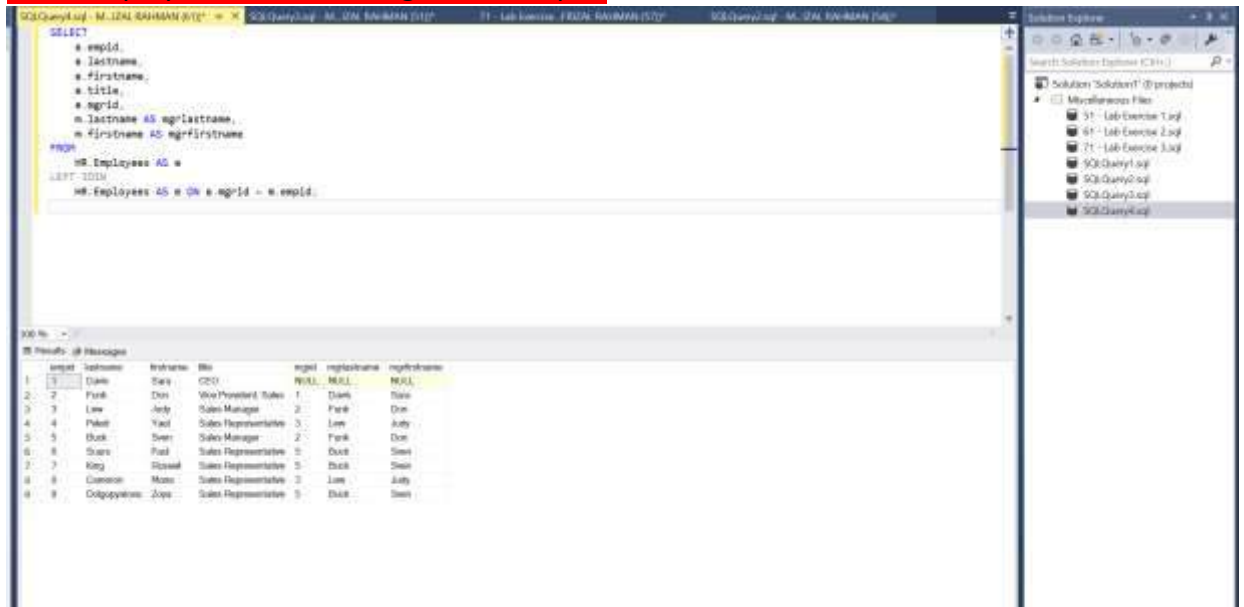
Execute the 2nd stage of the test and compare it with 7 3 - Lab Exercise 3 - Task 2 Result.txt . If the results are the same, then your test is correct.

```
SELECT
    e.empid,
    e.lastname,
    e.firstname,
    e.title,
    e.mgrid,
    m.lastname AS mgrlastname,
    m.firstname AS mgrfirstname
FROM
    HR.Employees AS e
LEFT JOIN
```



Practical – Part

HR.Employees AS m ON e.mgrid = m.empid;



[Question- 20] Is it mandatory to write the table alias name when executing the SELF-JOIN command? Can the original table name be used as an alias name? Explain!

2. Is it mandatory to use an alias in a SELF-JOIN?

- No, it is not mandatory** to use a table alias in a SELF-JOIN, but it is highly recommended to do so. When you join a table to itself, using aliases helps distinguish between the different instances of the same table, making your query more readable and understandable.

2. Can the original table name be used as an alias?

- Yes, the original table name** can be used as an alias, but it's uncommon and can lead to confusion. Using a different alias (like e for employees and m for managers) is more readable and makes it clear which instance of the table you're referring to in the query.



SQLQuery4.sql - M...IZAL RAHMAN (61))
SQLQuery3.sql - M...IZAL RAHMAN (51))
71 - Lab Exercise...FRIZAL RAHMAN (57))

```
SELECT
  HR.Employees.empid,
  HR.Employees.lastname,
  HR.Employees.firstname,
  HR.Employees.title,
  HR.Employees.mgrid,
  manager.lastname AS mgrlastname,
  manager.firstname AS mgrfirstname
FROM
  HR.Employees
LEFT JOIN
  HR.Employees AS manager ON HR.Employees.mgrid = manager.empid;
```

100 %

Results Messages

	empid	lastname	firstname	title	mgrid	mgrlastname	mgrfirstname
1	1	Davis	Sara	CEO	NULL	NULL	NULL
2	2	Funk	Don	Vice President, Sales	1	Davis	Sara
3	3	Lew	Judy	Sales Manager	2	Funk	Don
4	4	Peled	Yael	Sales Representative	3	Lew	Judy
5	5	Buck	Sven	Sales Manager	2	Funk	Don
6	6	Suurs	Paul	Sales Representative	5	Buck	Sven
7	7	King	Russell	Sales Representative	5	Buck	Sven
8	8	Cameron	Maria	Sales Representative	3	Lew	Judy
9	9	Dolgopiatova	Zoya	Sales Representative	5	Buck	Sven

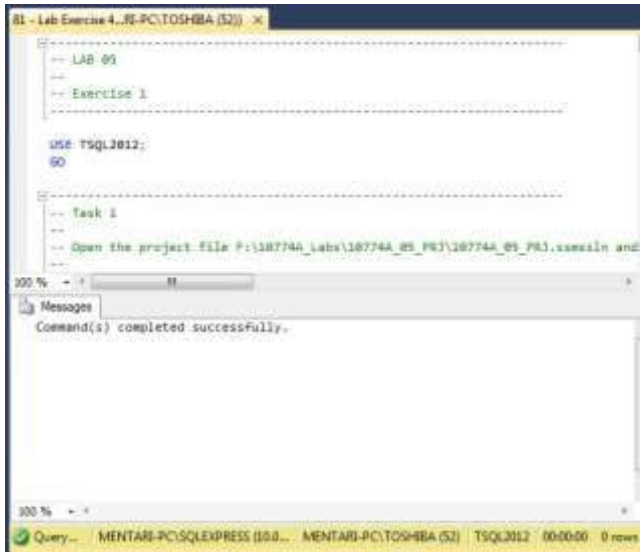
7

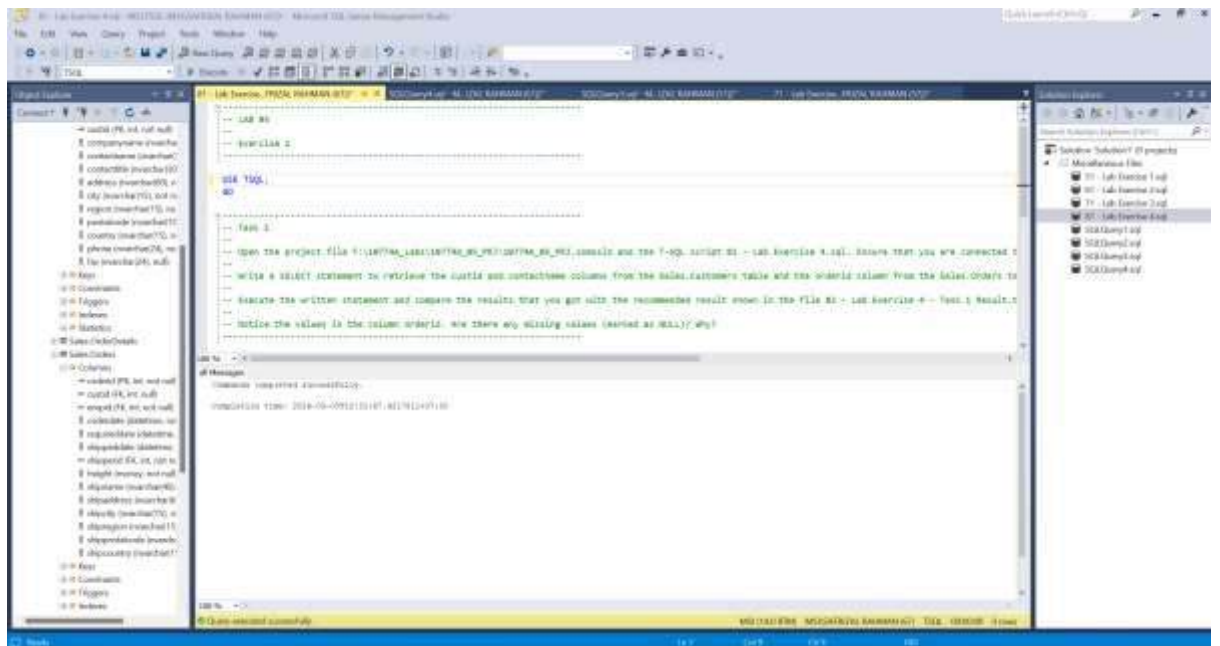
Conclusion : After doing this part of the practicum, you should understand how to write a T-SQL SELF-JOIN statement.



Practical – Part

9 : Creating Outer-Join Query

Step	Information
1	<p>The case study used in this practicum part 4 continues the practicum in part 3. The sales department is quite satisfied with the report that has been made. Then the sales department wants to change the report to show all customers, even though the customer does not have an order history or customers who have an order history. Therefore, a SELECT clause is needed to retrieve all rows from the Sales.Customers table (custid and contactname columns) and the orderid column From the Sales.Orders table.</p> <p>Open the project \10774A Labs\10774A_05_PRJ\10774A_05_PRJ.ssmssln and the T-SQL script 81 - Lab Exercise 4.sql. Make sure the database is connected with "TSQL".</p> 



[Question- 21] Write a T-SQL command with a SELECT clause to retrieve the custid and contactname columns from the table Sales.Customers and the orderid column from the Sales.Orders table . The command created must retrieve all rows from the Sales.Customers table .

To retrieve all rows from the Sales.Customers table, including those customers who might not have any orders, you should use a **LEFT JOIN**. This ensures that even customers without orders (where orderid might be NULL) are included in the result.

Here's the T-SQL command:

sql

Copy code

SELECT

c.custid,

c.contactname,

o.orderid

FROM

Sales.Customers AS c

LEFT JOIN


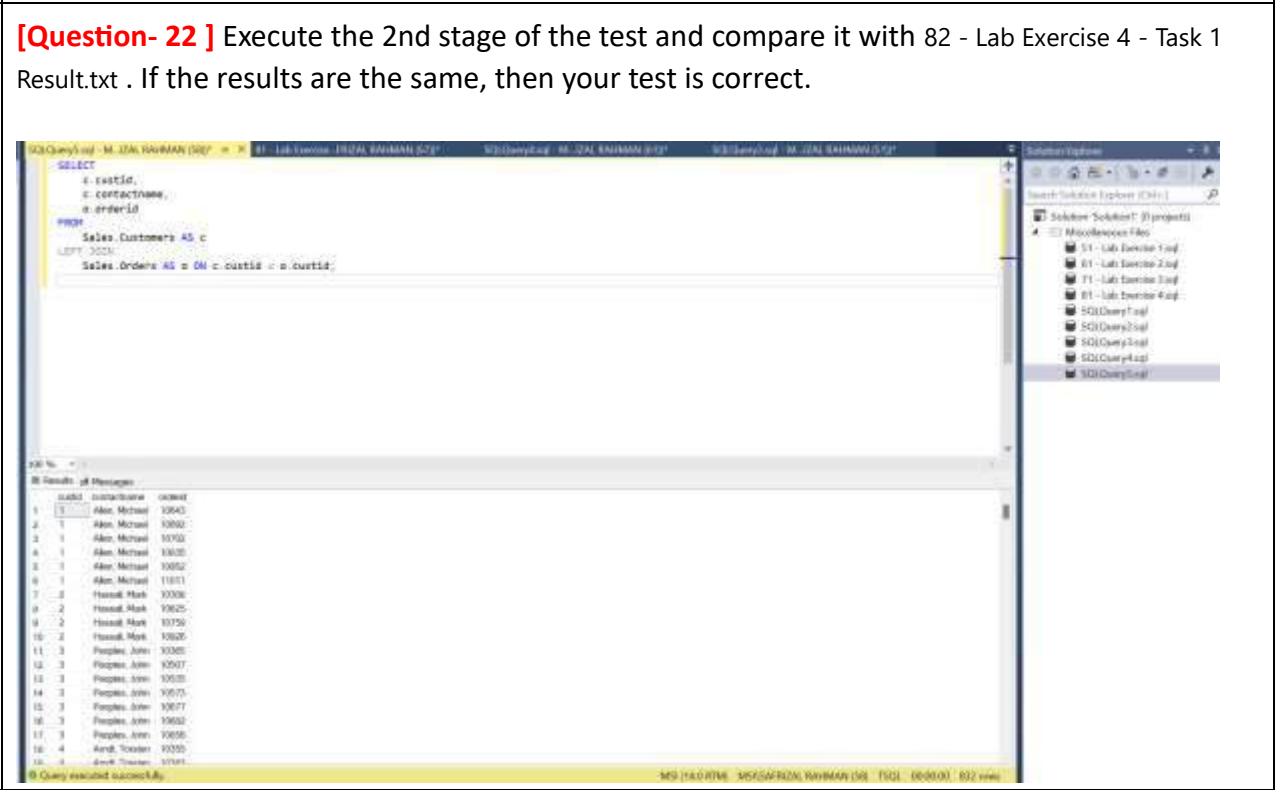
Sales.Orders AS o ON c.custid = o.custid;

- **c:** Alias for Sales.Customers table.

- **o:** Alias for Sales.Orders table.

- **LEFT JOIN:** Ensures that all rows from the Sales.Customers table are retrieved, even if there are no corresponding orderid values in Sales.Orders.


Practical – Part

	
3	<p>[Question- 22] Execute the 2nd stage of the test and compare it with 82 - Lab Exercise 4 - Task 1 Result.txt . If the results are the same, then your test is correct.</p> 
4	<p>[Question- 23] Pay attention to the values in the orderid column . Are there any missing values (NULL)? Why?</p> <p>Are there any missing values (NULL)?</p> <ul style="list-style-type: none"> Yes, there might be NULL values in the orderid column. These would occur for customers who have not placed any orders. <p>Why are there NULL values?</p>



	<ul style="list-style-type: none"> The query uses a LEFT JOIN between Sales.Customers and Sales.Orders. A LEFT JOIN retrieves all rows from the left table (Sales.Customers) and matches them with the right table (Sales.Orders). If a customer does not have any corresponding orders, the orderid column will be NULL for that customer.
5	Conclusion : After doing this part of the practicum, you should understand how to write the TSQL OUTER-JOIN statement .

10 : Creating a Cross-Join Query

Step	Information
1	<p>This case study begins with the HR department wanting to set up a personal calendar for each employee. The IT department will provide a T-SQL code that generates all days in the past year. Therefore, <i>the developer</i> will use the SELECT clause to return all rows from the calendar table for each row in the HR.Employees table.</p> <p>Open the project \10774A Labs\10774A_05_PRJ\10774A_05_PRJ.ssmssln and the T-SQL script 91 - Lab Exercise 5.sql. Make sure the database is connected with "TSQL".</p> 

Practical – Part



[Question- 24] Run the T-SQL code under task 1. Display the output! (Don't worry if you don't understand the T-SQL code. The next step will provide a more concrete example of how CROSSJOIN is implemented.) The T-SQL code under Task 1 creates and populates the HR.Calendar table with dates for the current year. Here is the code:

SQL

SET NOCOUNT ON;

IF OBJECT_ID('HR.Calendar') IS NOT NULL

DROP TABLE HR.Calendar;

CREATE TABLE HR.Calendar (

calendardate DATE CONSTRAINT PK_Calendar PRIMARY KEY

);

DECLARE

@startdate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 1, 1),

@enddate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 12, 31);

WHILE @startdate <= @enddate

BEGIN

INSERT INTO HR.Calendar (calendardate)

VALUES (@startdate);

SET @startdate = DATEADD(DAY, 1, @startdate);

2



END;

SET NOCOUNT OFF;

GO

-- Observe the HR.Calendar table

SELECT

 calendardate

FROM HR.Calendar;

The screenshot displays a SQL Server Enterprise Manager window. The top pane shows a query script for creating and populating the HR.Calendar table. The script includes a SET NOCOUNT ON statement, a check for the table's existence, a CREATE TABLE statement with a primary key, and a WHILE loop that inserts dates from 2024-01-01 to 2024-01-18. The bottom pane shows the results of the query, which is a list of dates from 2024-01-01 to 2024-01-18. A status bar at the bottom indicates that the query was executed successfully.

```
SQLQuery1.sql - M...IZAL RAHMAN (56)*  X  91 - Lab Exercise...FRIZAL RAHMAN (53)*
SET NOCOUNT ON;

IF OBJECT_ID('HR.Calendar') IS NOT NULL
    DROP TABLE HR.Calendar;

CREATE TABLE HR.Calendar (
    calendardate DATE CONSTRAINT PK_Calendar PRIMARY KEY
);

DECLARE
    @startdate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 1, 1),
    @enddate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 12, 31);

WHILE @startdate <= @enddate
BEGIN
    INSERT INTO HR.Calendar (calendardate)
    VALUES (@startdate);

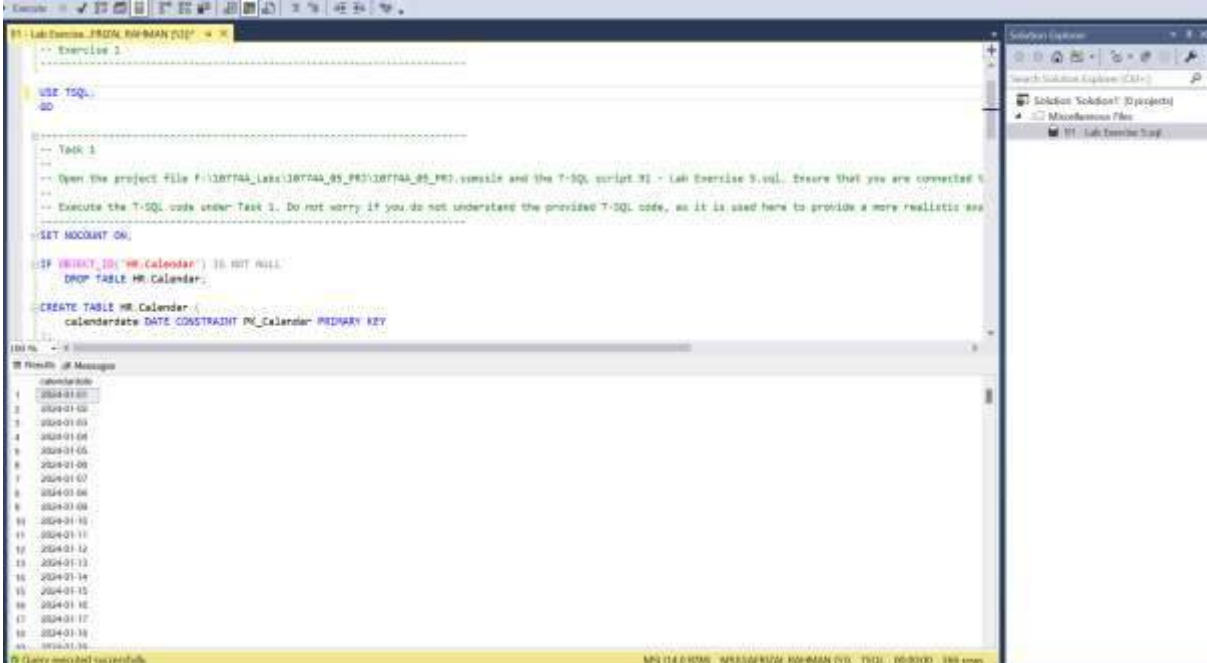
    SET @startdate = DATEADD(DAY, 1, @startdate);
END;
```

	calendardate
1	2024-01-01
2	2024-01-02
3	2024-01-03
4	2024-01-04
5	2024-01-05
6	2024-01-06
7	2024-01-07
8	2024-01-08
9	2024-01-09
10	2024-01-10
11	2024-01-11
12	2024-01-12
13	2024-01-13
14	2024-01-14
15	2024-01-15
16	2024-01-16
17	2024-01-17
18	2024-01-18
19	2024-01-19


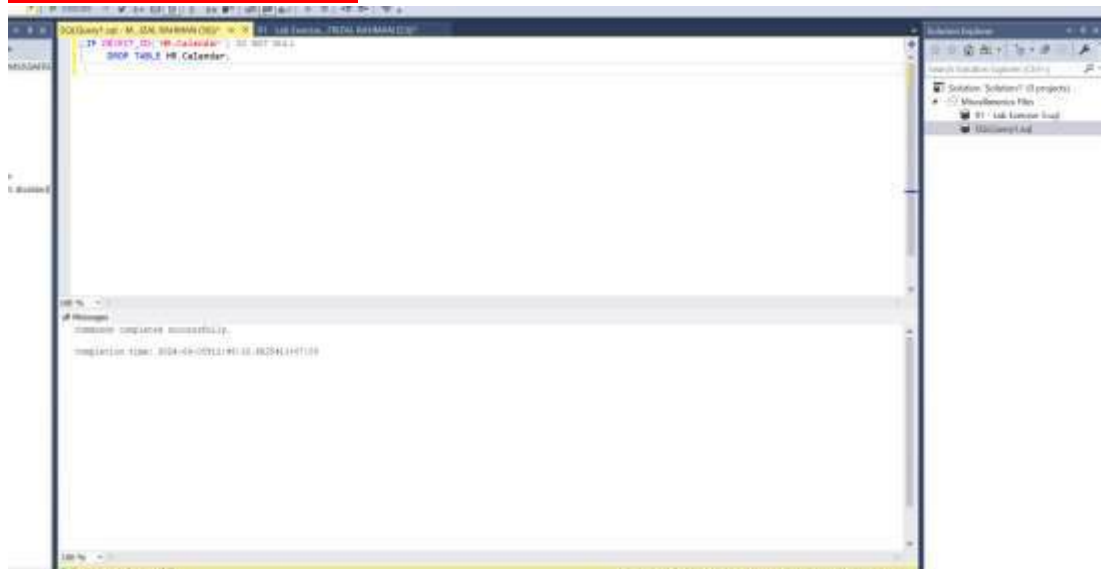
Query executed successfully. MSI



Practical – Part

	 <pre>-- Task 1 -- Open the project file F:\187TAA_Laba\187TAA_05_P03\187TAA_05_P03.ssmfile and the T-SQL script 03 - Lab Exercise 3.sql. Ensure that you are connected to -- Execute the T-SQL code under Task 1. Do not worry if you do not understand the provided T-SQL code, as it is used here to provide a more realistic scenario. SET NOCOUNT ON; IF (EXISTS (SELECT * FROM HR.Calendar)) IS NOT NULL DROP TABLE HR.Calendar; CREATE TABLE HR.Calendar (calendardate DATE CONSTRAINT PK_Calendar PRIMARY KEY);</pre> <p>Results: 0 Messages</p> <table><tr><th>Calendardate</th></tr><tr><td>2024-01-01</td></tr><tr><td>2024-01-02</td></tr><tr><td>2024-01-03</td></tr><tr><td>2024-01-04</td></tr><tr><td>2024-01-05</td></tr><tr><td>2024-01-06</td></tr><tr><td>2024-01-07</td></tr><tr><td>2024-01-08</td></tr><tr><td>2024-01-09</td></tr><tr><td>2024-01-10</td></tr><tr><td>2024-01-11</td></tr><tr><td>2024-01-12</td></tr><tr><td>2024-01-13</td></tr><tr><td>2024-01-14</td></tr><tr><td>2024-01-15</td></tr><tr><td>2024-01-16</td></tr><tr><td>2024-01-17</td></tr><tr><td>2024-01-18</td></tr><tr><td>2024-01-19</td></tr><tr><td>2024-01-20</td></tr><tr><td>2024-01-21</td></tr><tr><td>2024-01-22</td></tr><tr><td>2024-01-23</td></tr><tr><td>2024-01-24</td></tr><tr><td>2024-01-25</td></tr><tr><td>2024-01-26</td></tr><tr><td>2024-01-27</td></tr><tr><td>2024-01-28</td></tr><tr><td>2024-01-29</td></tr><tr><td>2024-01-30</td></tr><tr><td>2024-01-31</td></tr></table> <p>MD 114.8.1015 MS SQL SERVER 2019 (V1) TSQL: P03-0001 188 rows</p>	Calendardate	2024-01-01	2024-01-02	2024-01-03	2024-01-04	2024-01-05	2024-01-06	2024-01-07	2024-01-08	2024-01-09	2024-01-10	2024-01-11	2024-01-12	2024-01-13	2024-01-14	2024-01-15	2024-01-16	2024-01-17	2024-01-18	2024-01-19	2024-01-20	2024-01-21	2024-01-22	2024-01-23	2024-01-24	2024-01-25	2024-01-26	2024-01-27	2024-01-28	2024-01-29	2024-01-30	2024-01-31
Calendardate																																	
2024-01-01																																	
2024-01-02																																	
2024-01-03																																	
2024-01-04																																	
2024-01-05																																	
2024-01-06																																	
2024-01-07																																	
2024-01-08																																	
2024-01-09																																	
2024-01-10																																	
2024-01-11																																	
2024-01-12																																	
2024-01-13																																	
2024-01-14																																	
2024-01-15																																	
2024-01-16																																	
2024-01-17																																	
2024-01-18																																	
2024-01-19																																	
2024-01-20																																	
2024-01-21																																	
2024-01-22																																	
2024-01-23																																	
2024-01-24																																	
2024-01-25																																	
2024-01-26																																	
2024-01-27																																	
2024-01-28																																	
2024-01-29																																	
2024-01-30																																	
2024-01-31																																	
3	<p>[Question- 25] Write a SELECT command to retrieve values from the empid, firstname, and lastname columns from the HR.Employees table and the calendardate column from the HR.Calendar table</p> <pre>SELECT e.empid, e.firstname, e.lastname, c.calendardate FROM HR.Employees e CROSS JOIN HR.Calendar c;</pre>																																



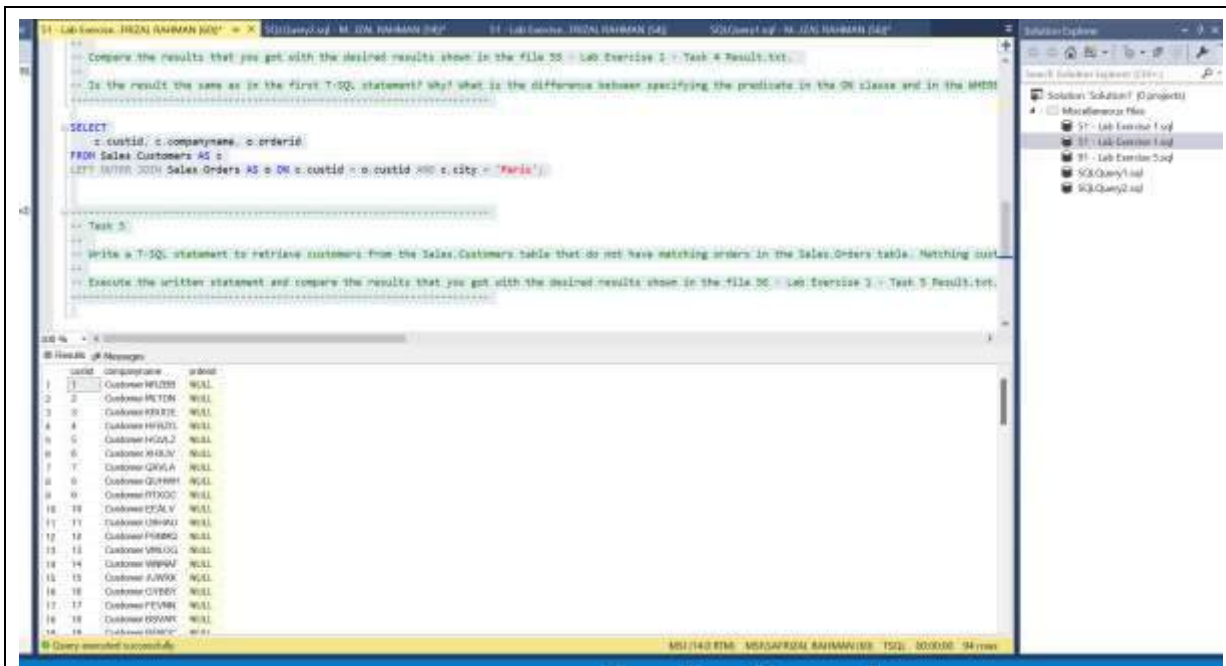
	
4	<p>[Question-2 6] Execute the 3rd stage test and compare it with the file 92 - Lab Exercise 5 - Task 2 Result.txt . If the results are the same, then your test is correct.</p> <p>By completing these tasks, you've demonstrated your understanding of how to write and execute T-SQL CROSS JOIN queries. If you have any further questions or need additional assistance, feel free to ask!</p> 
5	Drop the HR.Calendar table by executing the T-SQL code below task 3.

**Practical – Part**

6	Conclusion : After completing this practical section, you will understand how to write T-SQL CROSS-JOIN code .

Practical – Part 11 : Writing Queries Who Will Filter Data with WHERE clause

Step	Information
1	<p>The scenario in this practicum uses the problems in the marketing department. The marketing department is working on several campaigns for old customers. The marketing staff needs a different customer list according to several business rules. Therefore, <i>the developer</i> will write a SELECT command to retrieve the desired rows from the Sales.Customers table.</p> <p>Open the project \10774A Labs\10774A_06_PRJ\10774A_06_PRJ.ssmssln and the T-SQL script 51 - Lab Exercise 1.sql. Make sure the database is connected with “TSQL”.</p>



Write a SELECT statement that will return the column values from a table, Then filter the results to only customers who are from "Brazil"!

```

SELECT
  custid, companyname, contactname, address, city, country,
  telephone FROM Sales . Customers WHERE
  country = 'Brazil' ;

```

Use of the N prefix for literal characters ('N'Brazil'). This prefix is used because the country column is a Unicode data type. When expressing Unicode characters literally, the N character (for National) is specified as the prefix.

```

SELECT
  custid,
  companyname,
  contactname,
  address,
  city,
  country,
  telephone
FROM Sales.Customers
WHERE country = N'Brazil';

```

```

SELECT
  custid,
  companyname,
  contactname,
  address,

```



Practical – Part

city,

country,

phone -- Replace 'phone' with the correct column name if 'telephone' is incorrect

FROM Sales.Customers

WHERE country = 'Brazil';

```

SELECT
  custid,
  companyname,
  contactname,
  address,
  city,
  country,
  phone
FROM Sales.Customers
WHERE country = 'Brazil';

```

	custid	companyname	contactname	address	city	country	phone
1	35	Customer J.BORG	Richardson, Sharon	Av. dos Laranjeiros, 8789	Sao Paulo	Brazil	(11) 812-2408
2	21	Customer K.DRY	Rosen, George	Rua Orto, 3456	Sao Paulo	Brazil	(11) 456-7890
3	31	Customer Y.KING	Cheng, Yan Gang	Av. Street, 5678	Campinas	Brazil	(11) 957-8901
4	34	Customer W.WIG	Gomez, Lily	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
5	81	Customer W.LAW	Petrovic, Miroslav	Piazza Parkside, 1234	Rio de Janeiro	Brazil	(21) 876-5432
6	82	Customer W.PIC	Miles, Anna	Avenida dos Capetins, 1234	Sao Paulo	Brazil	(11) 901-2345
7	87	Customer G.VEL	Garcia, Ivan	Av. Copacabana, 6789	Rio de Janeiro	Brazil	(21) 345-6789
8	81	Customer G.QWY	Nguyen, Jean Philippe	Av. Friburgo Centro, 1234	Sao Paulo	Brazil	(11) 123-4567
9	90	Customer G.QWY	Li, Yan	Rua do Mercado, 4567	Florianopolis	Brazil	(48) 234-5678

[Question- 27] Execute the 2nd stage of the test and compare it with the file 52 - Lab Exercise 1 - Task 1 Result.txt . If the results are the same, then your test is correct.

3

```

SELECT
  custid,
  companyname,
  contactname,
  address,
  city,
  country,
  phone
FROM Sales.Customers
WHERE country IN ('Brazil', 'UK', 'USA');

```

	custid	companyname	contactname	address	city	country	phone
1	4	Customer H.BIG	Arnell, Stephen	7890 Market St.	London	UK	(773) 456-7890
2	11	Customer L.BANK	Julia, David	Paumotu Circle 4567	London	UK	(773) 789-0123
3	19	Customer J.AWOL	Richardson, Sharon	Av. dos Laranjeiros, 8789	Sao Paulo	Brazil	(11) 812-2408
4	16	Customer D.DRIF	Henry, David	Bowling Green 0123 Brewery	London	UK	(773) 234-5678
5	14	Customer H.BANK	Bowman, Harold	5678 King George	London	UK	(773) 345-6789
6	21	Customer K.DRY	Rosen, George	Rua Orto, 3456	Sao Paulo	Brazil	(11) 456-7890
7	31	Customer Y.KING	Cheng, Yan Gang	Av. Street, 5678	Campinas	Brazil	(11) 957-8901
8	34	Customer W.WIG	Kishner, Verity	6789 Baker Blvd	Eugene	USA	(503) 355-0122
9	36	Customer W.WIG	Gomez, Lily	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
10	88	Customer L.VIRO	Baile, Denise	City Center Plaza 2345 Main St	Elgin	USA	(803) 345-6789
11	38	Customer L.VIRO	Lee, David	San Jose Avenue Center Hwy 3456	Costa	USA	(408) 345-6789
12	43	Customer L.PICU	Dudynski, Anna	1801 Orchard Terrace	Waco, Texas	USA	(800) 555-0119
13	45	Customer G.VEL	Sankaranarayanan, Krishna	1234 Park St Suite 5	Sao Francisco	USA	(415) 555-0118
14	49	Customer G.VEL	Sankaranarayanan, Krishna	9812 Chaucerum Rd.	Portland	USA	(503) 555-0117
15	85	Customer G.VEL	Muller, Kim	South House 1234 Queensbridge	London	UK	(773) 889-1234
16	88	Customer G.VEL	Engelard Muller, Anna	7890 Market St	Amsterdam	USA	(800) 555-0115
17	81	Customer G.VEL	Petrovic, Miroslav	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
18	82	Customer G.VEL	Miles, Anna	Avenida dos Capetins, 1234	Sao Paulo	Brazil	(11) 901-2345
19	88	Customer G.VEL	Muller, Kim	7890 Market St	Amsterdam	USA	(800) 555-0115



Execute

SQLQuery3.sql - M...IZAL RAHMAN (57) - X 51 - Lab Exercise...FRIZAL RAHMAN (60) - SQLQuery2.sql - M...IZAL RAHMAN (59) - 51 - Lab Exercise...FRIZAL RAHMAN (54)

```
SELECT
    custid,
    companyname,
    contactname,
    address,
    city,
    country,
    phone -- Correct column name based on your output
FROM Sales.Customers
WHERE country = 'Brazil';
```

100 %

Results Messages

	custid	companyname	contactname	address	city	country	phone
1	15	Customer JAWOK	Richardson Steven	Av. dos Lusitãos, 6789	Sao Paulo	Brazil	(11) 012-3456
2	21	Customer KIDPX	Ruikeo, Clotilde	Rua Ordo, 3456	Sao Paulo	Brazil	(11) 456-7890
3	31	Customer YJCBX	Cheng, Yao-Gang	Av. Brasil, 5678	Campanas	Brazil	(11) 567-8901
4	34	Customer BVRGL	Cohen, Shy	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
5	61	Customer WULWD	Flarczyk, Krzysztof	Rua da Penitenciaría, 1234	Rio de Janeiro	Brazil	(21) 678-9012
6	62	Customer WFGZJ	Hawec, Anna	Alameda dos Cavaleiros, 1234	Sao Paulo	Brazil	(11) 901-2345
7	67	Customer QVEPD	Gardier, Elish	Av. Copacabana, 4567	Rio de Janeiro	Brazil	(21) 345-6789
8	81	Customer YQZWW	Nagel, Jean Philippe	Av. Infã de Castro, 1234	Sao Paulo	Brazil	(11) 123-4567
9	88	Customer SRQVM	Li, Yan	Rua do Mercado, 4567	Ribeirão	Brazil	(14) 234-5678



4

[Question- 28] Write a SELECT command that will return values in the custid, companyname, contactname, address, city, columns. country, and phone in the Sales.Customers table , then filter the results only for “Brazil, UK and USA” (Use the IN predicate in the WHERE clause).

```
SQLQuery3.sql - M...IZAL RAHMAN (57) - X
SELECT
    custid,
    companyname,
    contactname,
    address,
    city,
    country,
    phone -- Correct column name based on your output
FROM Sales.Customers
WHERE country = N'Brazil';
```

100 %

Results Messages

	custid	companyname	contactname	address	city	country	phone
1	15	Customer JAWOK	Richardson, Shawn	Av. dos Lusitâos, 6769	Sao Paulo	Brazil	(11) 012-3456
2	21	Customer KIDPX	Russo, Giuseppe	Rua Orós, 3456	Sao Paulo	Brazil	(11) 456-7890
3	31	Customer YACBX	Cheng, Yao-Guang	Av. Brasil, 5678	Campanas	Brazil	(11) 567-8901
4	34	Customer BVRGQ	Cohen, Shy	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
5	81	Customer WULWD	Flarczyk, Krzysztof	Rua da Padiscadora, 1234	Rio de Janeiro	Brazil	(21) 678-9012
6	62	Customer WFIJZ	Mawo, Anna	Alameda dos Canôes, 1234	Sao Paulo	Brazil	(11) 901-2345
7	67	Customer QVSPD	Gardien, Eskin	Av. Copacabana, 8769	Rio de Janeiro	Brazil	(21) 345-6789
8	81	Customer YQQWW	Nagel, Jean Philippe	Av. Infês de Castro, 1234	Sao Paulo	Brazil	(11) 123-4567
8	88	Customer SRQVM	Li, Yan	Rua do Mercado, 4567	Ressende	Brazil	(14) 234-5678

5

[Question-2 9] Execute the 3rd stage test and compare it with file 53 - Lab Exercise 1 - Task 2 Result.txt . If the results are the same, then your test is correct.

```
SQLQuery3.sql - M...IZAL RAHMAN (57) - X
SELECT
    custid,
    companyname,
    contactname,
    address,
    city,
    country,
    phone
FROM Sales.Customers
WHERE contactname LIKE 'AN';
```

100 %

Results Messages

	custid	companyname	contactname	address	city	country	phone
1	1	Customer NFGZBB	Allen, Michael	Obers Str. 6123	Berlin	Germany	030-3456789
2	4	Customer HFBZG	Andt, Torsten	7890 Hannover Str.	London	UK	(171) 456-7890



SQLQuery3.sql - M. IZAL RAHMAN (57) * X 51 - Lab Exercise - FRIZAL RAHMAN (50) * SQLQuery2.sql - M. IZAL RAHMAN (59) * 51 - Lab Exercise - FRIZAL RAHMAN (54)

```

SELECT
    custid,
    companyname,
    contactname,
    address,
    city,
    country,
    phone
FROM Sales.Customers
WHERE country IN ('Brazil', 'UK', 'USA');

```

100 %

Results Messages

	custid	companyname	contactname	address	city	country	phone
1	4	Customer HFBZG	Amel, Tordson	7890 Hanover Sq	London	UK	(171) 456-7890
2	11	Customer UBHAU	Jaffe, David	Fauntleroy Circus 4567	London	UK	(171) 789-0123
3	15	Customer JUWKK	Richardson, Shawn	Av. dos Laranjeiros, 6789	Sao Paulo	Brazil	(11) 012-3456
4	18	Customer GYBSY	Bekky, Dana	Berkeley Gardens, 0123 Brewery	London	UK	(171) 234-5678
5	19	Customer RFBQC	Boseman, Randall	5678 King George	London	UK	(171) 345-6789
6	21	Customer KIDPX	Russo, Giuseppe	Rua Ordo, 3456	Sao Paulo	Brazil	(11) 456-7890
7	31	Customer YJCBX	Cheng, Yao-Qiang	Av. Brasil, 5678	Campanas	Brazil	(11) 567-8901
8	32	Customer YSIGX	Kristian, Vicky	6789 Baker Blvd.	Eugene	USA	(503) 555-0122
9	34	Customer EYVIG	Cohen, Shy	Rua do Paço, 7890	Rio de Janeiro	Brazil	(21) 789-0123
10	36	Customer LVJSD	Smith, Denise	City Center Plaza 2345 Main St.	Eden	USA	(603) 555-0126
11	38	Customer LJJCA	Lee, Frank	Garden House Crowther Way 3456	Cowes	UK	(198) 567-8901
12	43	Customer UHSOI	Despanda, Anu	8901 Orchestra Terrace	Walla Walla	USA	(509) 555-0119
13	45	Customer QKPYT	Sankummaral, Krishna	1234 Pole St, Suite 5	San Francisco	USA	(415) 555-0118
14	48	Customer QVTMB	Szymczak, Radoslaw	8012 Chocomauro Rd.	Portland	USA	(503) 555-0117
15	53	Customer GCJSG	Muller, Ken	South House, 1234 Queensbridge	London	UK	(171) 990-1234
16	55	Customer KZQZT	Egeland-Muller, Anja	7890 Beijing St.	Anchorage	USA	(907) 555-0115
17	61	Customer WULWD	Florczyk, Krzysztof	Rua da Panificadora, 1234	Rio de Janeiro	Brazil	(21) 678-9012
18	62	Customer WFLZJ	Meyer, Anna	Alameda dos Canibos, 1234	Sao Paulo	Brazil	(11) 901-2345

The IT department has written T-SQL code to return values in the custid, companyname columns in the Sales.Customers table and the orderid column in the Sales.Orders table as below: `SELECT c . custid , c . companyname , o . orderid FROM Sales . Customers AS c LEFT OUTER JOIN Sales . Orders AS o ON c . custid = o . custid AND c . city = 'Paris' ;`

SQLQuery3.sql - M. IZAL RAHMAN (57) * X 51 - Lab Exercise - FRIZAL RAHMAN (50) * SQLQuery2.sql - M. IZAL RAHMAN (59) * 51 - Lab Exercise - FRIZAL RAHMAN (54)

```

SELECT
    c . custid , c . companyname , o . orderid
FROM Sales . Customers AS c
LEFT OUTER JOIN Sales . Orders AS o ON c . custid = o . custid AND c . city = 'Paris' ;

```

100 %

Results Messages

	custid	companyname	orderid
1	8	Customer NRZBB	NULL
2	2	Customer MLTDN	NULL
3	3	Customer KBUEE	NULL
4	4	Customer HFBZG	NULL
5	5	Customer HCVLZ	NULL
6	6	Customer XHQUV	NULL
7	7	Customer QKXLA	NULL
8	8	Customer QJHWH	NULL
9	9	Customer RTXQC	NULL
10	10	Customer EEALV	NULL
11	11	Customer UBHAU	NULL
12	12	Customer PSNMQ	NULL
13	13	Customer VNEOG	NULL
14	14	Customer WNMAF	NULL
15	15	Customer JUWKK	NULL
16	16	Customer GYBSY	NULL
17	17	Customer FEVNN	NULL
18	18	Customer PSWAE	NULL



7

Query execution in the 7th stage of the trial. Note two things, first the query will retrieve all rows in the Sales.Customers table . Second, the use of the comparison operator with the ON clause makes the city column more specific, namely the same as the value "Paris".

```
SQL - Lab Exercise 1..RI-PC\TOSHIBA (52) >
-- Is the result the same as in the first T-SQL statement? Why? What is the difference?

SELECT
  c.custid, c.companyname, o.orderid
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o ON c.custid = o.custid AND c.city = 'Paris';

-- Task 3
-- Write a T-SQL statement to retrieve customers from the Sales.Customers table
```

custid	companyname	orderid
1	Customer NRJZBB	NULL
2	Customer MLTDN	NULL
3	Customer KBUDG	NULL
4	Customer HFBZG	NULL
5	Customer HGVUZ	NULL
6	Customer XHGVV	NULL
7	Customer GXXVA	NULL
8	Customer QJHWH	NULL
9	Customer RTXGC	NULL


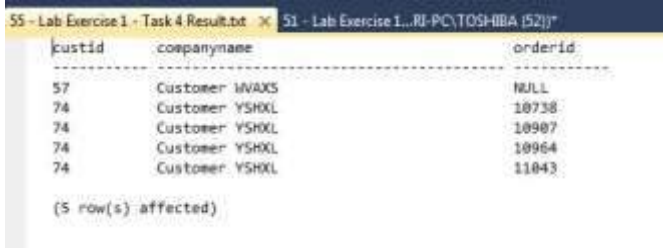
8

[Question-30] Copy the T-SQL Code in step 7 then modify it with the comparison operator for the city column in the WHERE clause. After that execute the code, show the result!

```
SQLQuery4.sql - M...IZAL RAHMAN (55) >
SELECT
  c.custid, c.companyname, o.orderid
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o ON c.custid = o.custid
WHERE c.city = 'Paris';
```

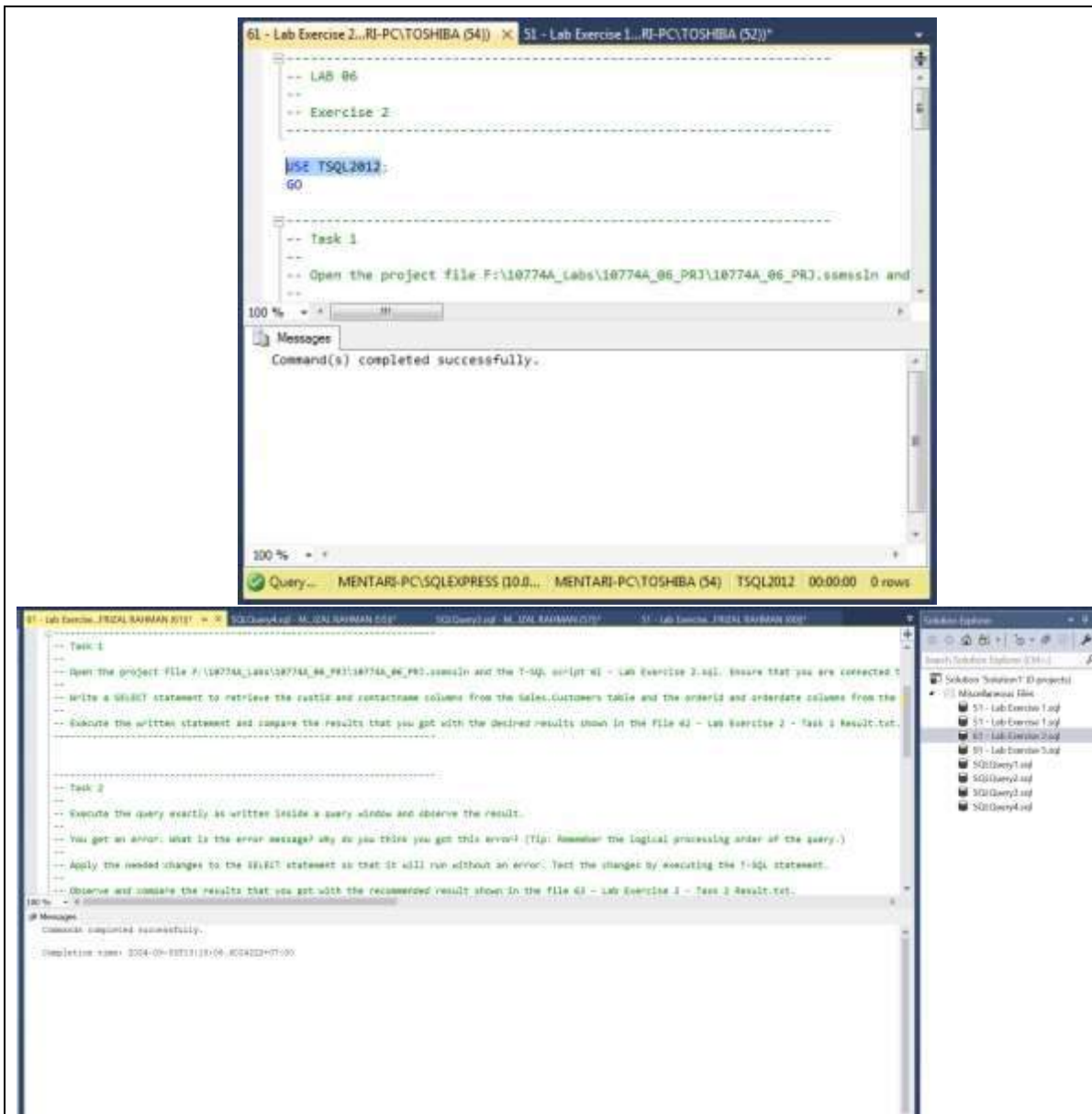
custid	companyname	orderid
57	Customer WVAXS	NULL
74	Customer YSHXL	10738
74	Customer YSHXL	10907
74	Customer YSHXL	10964
74	Customer YSHXL	11043



	 <pre>SELECT c.custid, c.companyname FROM Sales.Customers AS c LEFT OUTER JOIN Sales.Orders AS o ON c.custid = o.custid WHERE o.custid IS NULL;</pre> <p>Results</p> <table><thead><tr><th></th><th>custid</th><th>companyname</th></tr></thead><tbody><tr><td>1</td><td>22</td><td>Customer DTDNN</td></tr><tr><td>2</td><td>57</td><td>Customer WVAKS</td></tr></tbody></table>		custid	companyname	1	22	Customer DTDNN	2	57	Customer WVAKS									
	custid	companyname																	
1	22	Customer DTDNN																	
2	57	Customer WVAKS																	
9	<p>Compare the results of step 9 with file 55 - Lab Exercise 1 - Task 4 Result.txt . If the results are the same, then your test is correct.</p>  <table><thead><tr><th>custid</th><th>companyname</th><th>orderid</th></tr></thead><tbody><tr><td>57</td><td>Customer WVAKS</td><td>NULL</td></tr><tr><td>74</td><td>Customer YSHXL</td><td>10738</td></tr><tr><td>74</td><td>Customer YSHXL</td><td>10907</td></tr><tr><td>74</td><td>Customer YSHXL</td><td>10964</td></tr><tr><td>74</td><td>Customer YSHXL</td><td>11043</td></tr></tbody></table> <p>(5 row(s) affected)</p>	custid	companyname	orderid	57	Customer WVAKS	NULL	74	Customer YSHXL	10738	74	Customer YSHXL	10907	74	Customer YSHXL	10964	74	Customer YSHXL	11043
custid	companyname	orderid																	
57	Customer WVAKS	NULL																	
74	Customer YSHXL	10738																	
74	Customer YSHXL	10907																	
74	Customer YSHXL	10964																	
74	Customer YSHXL	11043																	
10	<p>Conclusion : After completing the practicum and answering the questions in this section, you should understand how to filter data rows from one or more tables using the WHERE clause with logical operator predicates.</p>																		

Practical – Part 11 : Writing Queries Which Will Sort Data with clause ORDER BY

Step	Information
1	<p>The case study in this lab is based on a problem in the sales department. The sales department wants to create a report that shows all orders with some customer information. In addition, there is an additional request to sort the data based on order dates and the customer IDs. The order rows in the previous lab were displayed without using the ORDER BY clause, therefore specifically for this lab section the WHERE command will be followed by the ORDER BY clause.</p> <p>Open the project \10774A Labs\10774A_06_PRJ\10774A_06_PRJ.ssmssln and the T-SQL script 61 - Lab Exercise 2.sql . Make sure the database is connected with "TSQL".</p>



2

[Question- 31] Write a SELECT command to retrieve the custid, custname columns from the Sales.Customers table and the orderid, orderdate columns from the Sales.Orders table ! Filter the results only for orders on or after April 1, 2008. Then sort the results based on orderdate in descending order and custid in ascending order!



The screenshot shows a SQL Server Enterprise window with a query editor and a results pane. The query is as follows:

```
SELECT
    c.custid,
    c.contactname,
    o.orderid,
    o.orderdate
FROM Sales.Customers AS c
INNER JOIN Sales.Orders AS o ON c.custid = o.custid
WHERE o.orderdate >= '2008-04-01'
ORDER BY o.orderdate DESC, c.custid ASC;
```

The results pane displays a table with the following data:

	custid	contactname	orderid	orderdate
1	9	Raghu, Armitazh	11076	2008-05-06 00:00:00.000
2	65	Moore, Michael	11077	2008-05-06 00:00:00.000
3	68	Mycha, Jacob	11075	2008-05-06 00:00:00.000
4	73	Gonzalez, Nuri	11074	2008-05-06 00:00:00.000
5	20	Kane, John	11072	2008-05-05 00:00:00.000
6	44	Louwerds, George	11070	2008-05-05 00:00:00.000
7	48	Dressler, Marlies	11071	2008-05-05 00:00:00.000
8	58	Fakhouri, Fadi	11073	2008-05-05 00:00:00.000
9	17	Jones, Tamera	11067	2008-05-04 00:00:00.000
10	62	Miles, Anna	11068	2008-05-04 00:00:00.000
11	80	Geschwendner, Jens	11069	2008-05-04 00:00:00.000
12	46	Dressler, Marlies	11065	2008-05-01 00:00:00.000
13	71	Navarro, Tomaz	11064	2008-05-01 00:00:00.000
14	88	Smith Jr., Ronaldo	11066	2008-05-01 00:00:00.000
15	27	Schmölke, Martin	11060	2008-04-30 00:00:00.000
16	32	Krehman, Verley	11061	2008-04-30 00:00:00.000
17	37	Cyrous, Odedu V	11063	2008-04-30 00:00:00.000
18	66	Voss, Florian	11062	2008-04-30 00:00:00.000
19	6	Robert, Charlie	11058	2008-04-30 00:00:00.000

The status bar at the bottom indicates: MSN (14.0) RTM - MSN (SA) RAHMAN (52) - TSQL - 00:01:00 - 88 rows.

[Question- 32] Execute the 2nd stage of the test and compare it with the file 62 - Lab Exercise 2 - Task 1 Result.txt . If the results are the same, then your test is correct.

The screenshot shows a SQL Server Enterprise window with a query editor and a results pane. The query is as follows:

```
SELECT
    c.custid,
    c.contactname,
    o.orderid,
    o.orderdate
FROM Sales.Customers AS c
INNER JOIN Sales.Orders AS o ON c.custid = o.custid
WHERE o.orderdate >= '2008-04-01'
ORDER BY o.orderdate DESC, c.custid ASC;
```

The results pane displays a table with the following data:

	custid	contactname	orderid	orderdate
1	9	Raghu, Armitazh	11076	2008-05-06 00:00:00.000
2	65	Moore, Michael	11077	2008-05-06 00:00:00.000
3	68	Mycha, Jacob	11075	2008-05-06 00:00:00.000
4	73	Gonzalez, Nuri	11074	2008-05-06 00:00:00.000
5	20	Kane, John	11072	2008-05-05 00:00:00.000
6	44	Louwerds, George	11070	2008-05-05 00:00:00.000
7	48	Dressler, Marlies	11071	2008-05-05 00:00:00.000
8	58	Fakhouri, Fadi	11073	2008-05-05 00:00:00.000
9	17	Jones, Tamera	11067	2008-05-04 00:00:00.000
10	62	Miles, Anna	11068	2008-05-04 00:00:00.000
11	80	Geschwendner, Jens	11069	2008-05-04 00:00:00.000
12	46	Dressler, Marlies	11065	2008-05-01 00:00:00.000
13	71	Navarro, Tomaz	11064	2008-05-01 00:00:00.000
14	88	Smith Jr., Ronaldo	11066	2008-05-01 00:00:00.000
15	27	Schmölke, Martin	11060	2008-04-30 00:00:00.000
16	32	Krehman, Verley	11061	2008-04-30 00:00:00.000
17	37	Cyrous, Odedu V	11063	2008-04-30 00:00:00.000
18	66	Voss, Florian	11062	2008-04-30 00:00:00.000
19	6	Robert, Charlie	11058	2008-04-30 00:00:00.000



The T-SQL command from the previous practicum followed by the WHERE command is as follows:

```
SELECT
e . empid , e . lastname , e . firstname , e . title , e . mgrid , m
. lastname AS mgrlastname , m . firstname AS mgrfirstname FROM HR .
Employees AS e
INNER JOIN HR . Employees AS m ON e . mgrid = m . empid WHERE
mgrlastname = N'Buck' ;
```

[Question- 33] Execute the T-SQL command at stage 3. Did an error occur? What is the error message? What do you think is the cause?

Msg 207, Level 16, State 1, Line 4

Invalid column name 'mgrlastname'.

[Question-3 4] Make changes to the T-SQL command to fix the error in the 3rd trial, then execute it! Compare the execution results with the file 63 - Lab Exercise 2 - Task 2 Result.txt. If the same, then the test result is correct.

4

The screenshot displays a SQL Server Enterprise Manager interface. The top pane shows a T-SQL query with the following text:

```
--SELECT
e . empid,
e . lastname,
e . firstname,
e . title,
e . mgrid,
m . lastname AS mgrlastname,
m . firstname AS mgrfirstname
FROM HR . Employees AS e
INNER JOIN HR . Employees AS m ON e . mgrid = m . empid
WHERE m . lastname = N'Buck' ;
```

The bottom pane shows the results of the query, which are three rows of data:

empid	lastname	firstname	title	mgrid	mgrlastname	mgrfirstname
6	Suurs	Paul	Sales Representative	5	Buck	Sean
7	King	Russell	Sales Representative	5	Buck	Sean
9	Dolgopyatova	Zoya	Sales Representative	5	Buck	Sean

Below the results window, there is a file named "63 - Lab Exercise 2 - Task 2 Result.txt" which contains the same data as the results window.



[Question- 35] Copy the T-SQL command in experiment 4, and modify it to produce all employees ORDER BY manager's first name. Initially test using the table's original name, then test using the table's alias name! Execute the T-SQL and compare the results to the 64 - Lab Exercise 2 - Task 3

Result.txt file . If the results are the same, then the experiment was correct.

64 - Lab Exercise 2 - Task 3 Result.txt

empid	lastname	firstname	title	mgrid	mgrlastname
3	Lew	Judy	Sales Manager	2	Funk
5	Buck	Sven	Sales Manager	2	Funk
4	Peled	Yael	Sales Representative	3	Lew
8	Cameron	Maria	Sales Representative	3	Lew
2	Funk	Don	Vice President, Sales	1	Davis
6	Suurs	Paul	Sales Representative	5	Buck
7	King	Russell	Sales Representative	5	Buck
9	Dolgopiatova	Zoya	Sales Representative	5	Buck

(8 row(s) affected)

5

SQLQuery5.sql - M. ZAL RAHMAN (320) ...

```

SELECT
    e.empid,
    e.lastname,
    e.firstname,
    e.title,
    e.mgrid,
    e.lastname AS mgrlastname,
    e.firstname AS mgrfirstname
FROM HR.Employees AS e
INNER JOIN HR.Employees AS m ON e.mgrid = m.empid
ORDER BY m.firstname;

```

100%

Results

empid	lastname	firstname	title	mgrid	mgrlastname	mgrfirstname
3	Lew	Judy	Sales Manager	2	Funk	Don
5	Buck	Sven	Sales Manager	2	Funk	Don
4	Peled	Yael	Sales Representative	3	Lew	Judy
8	Cameron	Maria	Sales Representative	3	Lew	Judy
2	Funk	Don	Vice President, Sales	1	Davis	Sara
6	Suurs	Paul	Sales Representative	5	Buck	Sven
7	King	Russell	Sales Representative	5	Buck	Sven
9	Dolgopiatova	Zoya	Sales Representative	5	Buck	Sven

6

[Question-3 6] Why can we use column names according to the original table name or use table alias names?

```

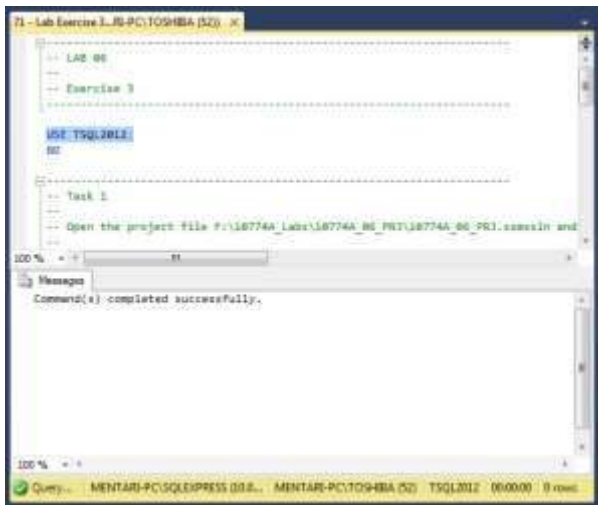
SELECT
    e.empid,
    e.lastname,

```



	<pre> e.firstname, e.title, e.mgrid, m.lastname AS mgrlastname, m.firstname AS mgrfirstname FROM HR.Employees AS e INNER JOIN HR.Employees AS m ON e.mgrid = m.empid ORDER BY mgrfirstname;</pre> <p>Clarity and Precision: Using table aliases helps make your queries clearer, especially when joining multiple tables. It ensures that the columns are correctly referenced and avoids ambiguity.</p> <p>Flexibility: SQL allows both approaches for flexibility. If table aliases are used, they provide a more concise way to write queries, and they can be used interchangeably with original table names if needed.</p>
7	<p>Conclusion : After working on the practical work and questions in this section, you should now understand how to use the ORDER BY clause .</p>

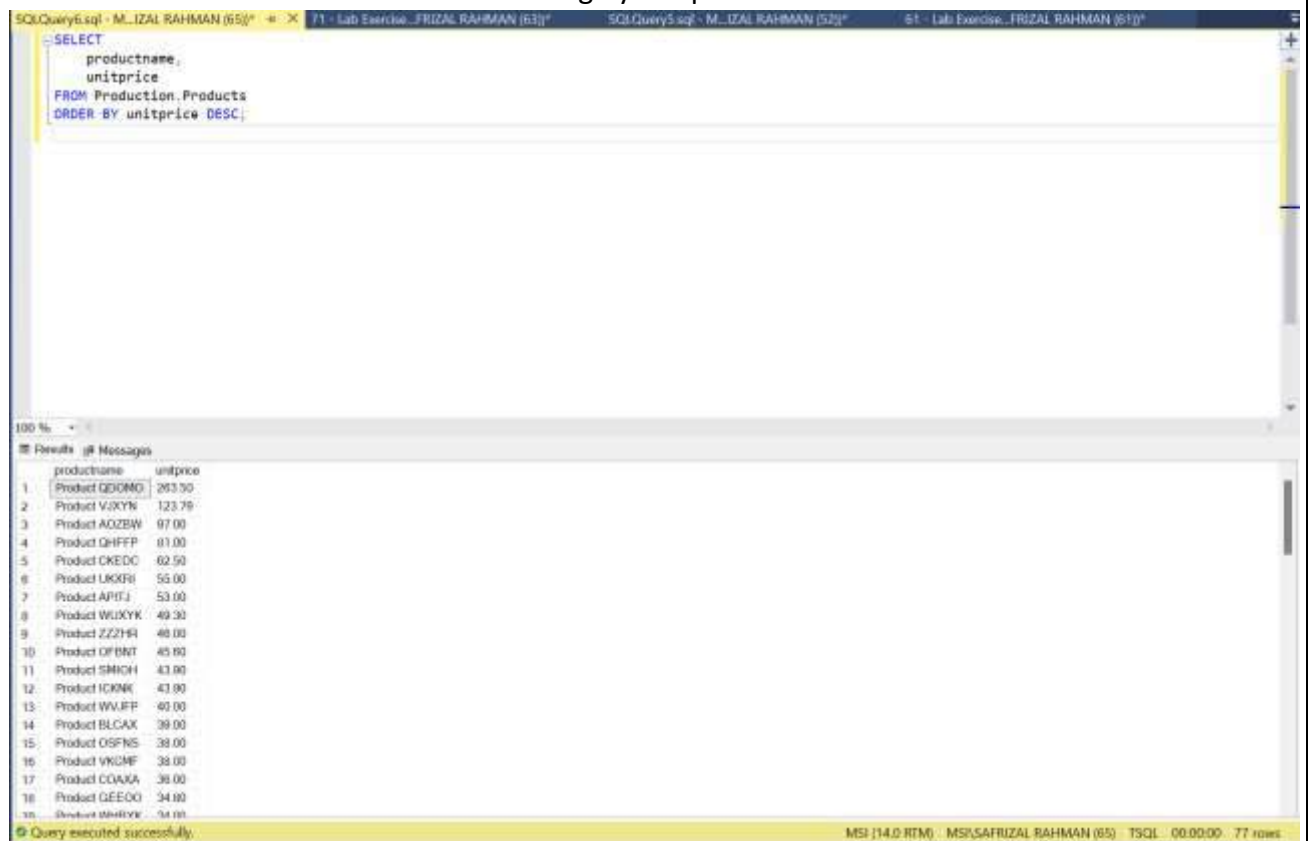
Practical – Part 12 : Writing Queries Who Will Do Data Filtering with clauses TOP

Step	Information
1	<p>Part 8 of the lab uses a case study on the sales department. The sales department wants to create an additional report that shows the order invoices and the 10 percent of the most expensive products that have been sold.</p> <p>Open the project \10774A Labs\10774A_06_PRJ\10774A_06_PRJ.ssmssln and the T-SQL script 71 - Lab Exercise 3.sql . Make sure the database is connected with "TSQL".</p> 



[Question- 37] Write a SELECT command to display the productname and unitprice columns in the Production.Products table sorted descending by unitprice! Show the execution results!

2





[Question- 38] Copy and modify the T-SQL command in trial 2 with the limitation that only 10 percent of the child products are displayed based on unitprice ordering! Execute the command, and compare whether it is in accordance with the file 73 - Lab Exercise 3 - Task 2 Result.txt.

productname	unitprice
Product Q00M0	263.50
Product VJXYN	123.79
Product A0ZBW	97.00
Product QHFFP	81.00
Product CKEDC	62.50
Product UKXRI	55.00
Product APITJ	53.00
Product WUXYK	49.30

(8 row(s) affected)

```
WITH SortedProducts AS (  
    SELECT  
        productname,  
        unitprice,  
        ROW_NUMBER() OVER (ORDER BY unitprice DESC) AS RowNum,  
        COUNT(*) OVER () AS TotalRows  
    FROM Production.Products  
)  
SELECT  
    productname,  
    unitprice  
FROM SortedProducts  
WHERE RowNum <= (TotalRows * 0.10);
```

productname	unitprice
Product Q00M0	263.50
Product VJXYN	123.79
Product A0ZBW	97.00
Product QHFFP	81.00
Product CKEDC	62.50
Product UKXRI	55.00
Product APITJ	53.00
Product WUXYK	49.30

(8 row(s) affected)

[Question- 39] Is it possible to implement the 5 trial T-SQL command using the OFFSET-FETCH clause?



Yes, you can use the OFFSET-FETCH clause to achieve similar results, but it works with a fixed number of rows or pages, not percentages directly. However, you can combine it with a row count calculation. For instance, if you know the total number of rows, you can calculate the number of rows to fetch for 10% and use OFFSET-FETCH to retrieve that subset:

sql

Copy code

```
;WITH SortedProducts AS (  
    SELECT  
        productname,  
        unitprice,  
        ROW_NUMBER() OVER (ORDER BY unitprice DESC) AS RowNum,  
        COUNT(*) OVER () AS TotalRows  
    FROM Production.Products  
)  
SELECT  
    productname,  
    unitprice  
FROM SortedProducts  
ORDER BY unitprice DESC  
OFFSET 0 ROWS  
FETCH NEXT CAST((TotalRows * 0.10) AS INT) ROWS ONLY;
```

In this example, OFFSET 0 ROWS starts at the beginning, and FETCH NEXT CAST((TotalRows * 0.10) AS INT) ROWS ONLY limits the number of rows fetched to approximately 10% of the total rows. Note that FETCH works with exact row numbers, so you need to compute the number of rows based on the total rows available.

-- First, get the total number of rows

```
DECLARE @TotalRows INT;
```

```
SELECT @TotalRows = COUNT(*)
```

```
FROM Production.Products;
```

-- Now, use OFFSET-FETCH to get the top 10% rows

```
WITH SortedProducts AS (  
    SELECT
```

```
        productname,
```

```
        unitprice,
```

```
        ROW_NUMBER() OVER (ORDER BY unitprice DESC) AS RowNum
```

```
    FROM Production.Products
```

```
)
```

```
SELECT
```

```
    productname,
```



```

unitprice
FROM SortedProducts
ORDER BY unitprice DESC
OFFSET 0 ROWS
FETCH NEXT CAST(@TotalRows * 0.10 AS INT) ROWS ONLY;

```

```

--DECLARE @TotalRows INT;

--SELECT @TotalRows = COUNT(*)
--FROM Production.Products;

-- Now, use OFFSET-FETCH to get the top 10% rows.
WITH SortedProducts AS (
    SELECT
        productname,
        unitprice,
        ROW_NUMBER() OVER (ORDER BY unitprice DESC) AS RowNum
    FROM Production.Products
)
SELECT
    productname,
    unitprice
FROM SortedProducts
ORDER BY unitprice DESC
OFFSET 0 ROWS
FETCH NEXT CAST(@TotalRows * 0.10 AS INT) ROWS ONLY;

```

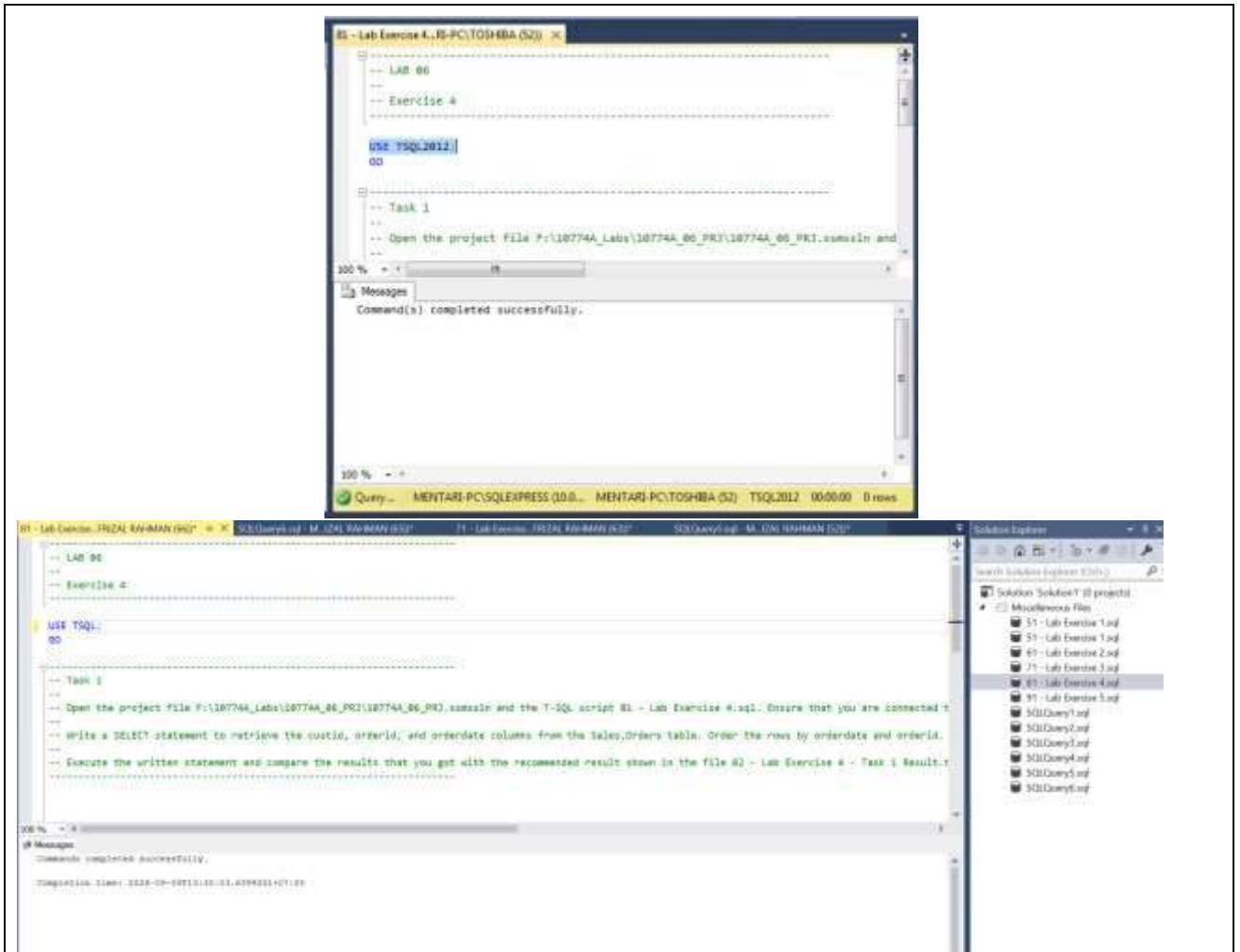
	productname	unitprice
1	Product GDORMO	263.50
2	Product VJOYV	123.79
3	Product AQZBW	87.00
4	Product QHFFP	81.00
5	Product QREDC	62.50
6	Product UROPH	55.00
7	Product APITJ	53.00

5

Conclusion : After completing the practical work and questions in this section, you should now understand how to apply the TOP option to the SELECT clause of the T-SQL command.

Practical – Part 13 : Writing Queries Who Will Filter Data with OFFSET-FETCH clause

Step	Information
1	<p>Practical part 9 will implement paging solution to display rows from Sales.Orders table , because the number of rows is too many. On each report page, user can only see 20 rows.</p> <p>Open the project \10774A Labs\10774A_06_PRJ\10774A_06_PRJ.ssmssln and the T-SQL script 81 - Lab Exercise 4.sql . Make sure the database is connected with “TSQL”.</p>



2

[Question- 40] Write a SELECT command to display the custid, orderid, and orderdate columns in the Sales.Orders table . Sort the rows by orderdate and orderid. Take the first 20 rows. Execute the command and compare the results with the file 82 - Lab Exercise 4 - Task 1 Result.txt. If the results are the same, then your test is correct.



SQLQuery7.sql - M. IZAL RAHMAN (72) * × 81 - Lab Exercise...FRIZAL RAHMAN (85) * SQLQuery6.sql - M. IZAL RAHMAN (85) * T1 - Lab Exercise...FRIZAL RAHMAN (83) *

```

SELECT
    custid,
    orderid,
    orderdate
FROM Sales.Orders
ORDER BY orderdate, orderid
OFFSET 0 ROWS
FETCH NEXT 20 ROWS ONLY;

```

100 %

Results Messages

	custid	orderid	orderdate
1	85	10248	2006-07-04 00:00:00.000
2	79	10249	2006-07-05 00:00:00.000
3	34	10250	2006-07-06 00:00:00.000
4	84	10251	2006-07-06 00:00:00.000
5	76	10252	2006-07-06 00:00:00.000
6	34	10253	2006-07-10 00:00:00.000
7	14	10254	2006-07-11 00:00:00.000
8	88	10255	2006-07-12 00:00:00.000
9	88	10256	2006-07-15 00:00:00.000
10	35	10257	2006-07-16 00:00:00.000
11	20	10258	2006-07-17 00:00:00.000
12	13	10259	2006-07-18 00:00:00.000
13	56	10260	2006-07-19 00:00:00.000
14	61	10261	2006-07-19 00:00:00.000
15	05	10262	2006-07-22 00:00:00.000
16	20	10263	2006-07-23 00:00:00.000
17	24	10264	2006-07-24 00:00:00.000
18	7	10265	2006-07-25 00:00:00.000
19	87	10266	2006-07-26 00:00:00.000
20	25	10267	2006-07-29 00:00:00.000

82 - Lab Exercise 4 - Task 1 Result.txt × 81 - Lab Exercise 4...RI-PC\TOSHIBA (52)

custid	orderid	orderdate
85	10248	2006-07-04 00:00:00.000
79	10249	2006-07-05 00:00:00.000
34	10250	2006-07-06 00:00:00.000
...		
...		
...		
7	10265	2006-07-25 00:00:00.000
87	10266	2006-07-26 00:00:00.000
25	10267	2006-07-29 00:00:00.000

(20 row(s) affected)

ORDER BY orderdate, orderid sorts the result by orderdate first and then by orderid.

OFFSET 0 ROWS skips no rows.

FETCH NEXT 20 ROWS ONLY retrieves the next 20 rows after the offset.

3

[Question- 41] Write a SELECT statement to display the same results as question no. 43, skip the first 20 rows, and continue with the next 20 rows using the OFFSET-FETCH clause! Execute the statement and compare 83 - Lab Exercise 4 - Task 2 Result.txt. If the results are the same, then your test is correct.

ORDER BY orderdate, orderid sorts the result by orderdate and orderid.

OFFSET 20 ROWS skips the first 20 rows.



2. **FETCH NEXT 20 ROWS ONLY** retrieves the next 20 rows after the offset.

The screenshot shows a SQL Server Enterprise Manager interface. The query window displays the following T-SQL query:

```
SELECT
    custid,
    orderid,
    orderdate
FROM Sales.Orders
ORDER BY orderdate, orderid
OFFSET 20 ROWS
FETCH NEXT 20 ROWS ONLY;
```

The results pane shows the following data:

custid	orderid	orderdate
33	10268	2006-07-30 00:00:00.000
89	10269	2006-07-31 00:00:00.000
87	10270	2006-08-01 00:00:00.000
73	10271	2006-08-01 00:00:00.000
85	10272	2006-08-02 00:00:00.000
83	10273	2006-08-05 00:00:00.000
85	10274	2006-08-06 00:00:00.000
49	10275	2006-08-07 00:00:00.000
80	10276	2006-08-08 00:00:00.000
52	10277	2006-08-08 00:00:00.000
5	10278	2006-08-12 00:00:00.000
44	10279	2006-08-13 00:00:00.000
6	10280	2006-08-14 00:00:00.000
89	10281	2006-08-14 00:00:00.000
88	10282	2006-08-15 00:00:00.000
88	10283	2006-08-16 00:00:00.000
44	10284	2006-08-19 00:00:00.000
83	10285	2006-08-20 00:00:00.000
81	10286	2006-08-21 00:00:00.000
81	10287	2006-08-22 00:00:00.000

The status bar at the bottom indicates: "MSD (14.0) RTM - MSSASPRJZALRAHMAN (72) - TSQL - 00:00:00 - 20 rows".

83 - Lab Exercise 4 - Task 2 Result.txt

custid	orderid	orderdate
33	10268	2006-07-30 00:00:00.000
89	10269	2006-07-31 00:00:00.000
87	10270	2006-08-01 00:00:00.000
...		
...		
63	10285	2006-08-20 00:00:00.000
63	10286	2006-08-21 00:00:00.000
67	10287	2006-08-22 00:00:00.000

(20 row(s) affected)

4

Conclusion : After working on the practical work and questions in this section, you should now understand how to use the OFFSET-FETCH clause in T-SQL commands.

-- Have a great time doing it -