Report

Оглавление

[Creating database wih tables (Uglovskaya,Chudinova,Rudish) 2](#_Toc121855468)

[Data preparation: (Uglovskaya,Chudinova) 3](#_Toc121855469)

[Task 1.1 4](#_Toc121855470)

[Count the number of unique Locations.(Rudish) 4](#_Toc121855471)

[Find all Customers whose LightSen is in the range between 4500-7000.(Rudish) 5](#_Toc121855472)

[Find all Customer that contains letters: LC and NC. (The letters go in this order, but between them there may be other letters and symbols. Letters are not case-sensitive).(Rudish) 6](#_Toc121855473)

[Display a list of all unique customers. (Chudinova) 7](#_Toc121855474)

[Find all Customers with GyrSen in the range between 22-24.(Chudinova) 7](#_Toc121855475)

[Task 1.2 8](#_Toc121855476)

[Find all Customers with a Number of sensor in the range between 25-27, and for all devices with STM or ESP, regardless of digits further instead of Yes in the iOSApp column display "1", and for No is 0 (Uglovskaya) 8](#_Toc121855477)

[Create a table price, which will indicate the cost of devices: for PLC – set 1000, CNC-10000, iot-100 for all the others arbitrarily (Chudinova) 9](#_Toc121855478)

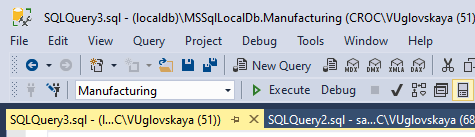
[Find all iosapp Customers that do not support wi-fi and update them to “yes” (Uglovskaya) 10](#_Toc121855479)

[Move all identifiers from customer and location to the corresponding tables and remove their mention in the name ( Uglovskaya) 10](#_Toc121855480)

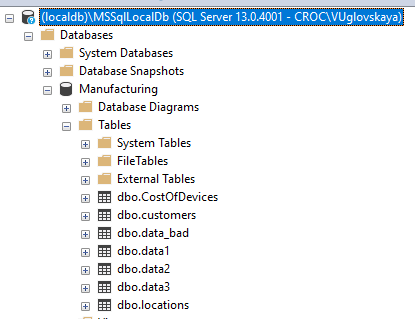
[Task 2 12](#_Toc121855481)

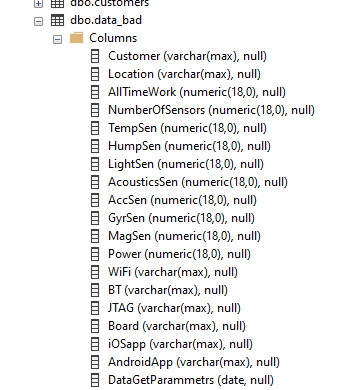
# Creating database wih tables (Uglovskaya,Chudinova,Rudish)

For fulfillment our project , MS Sql Server Management was used. We created database, which is called ‘manufacturing’



With the help of task import we downloaded information from CSV file to our database into sepearate tables.





# Data preparation: (Uglovskaya,Chudinova)

Analyzing File Bad\_Data we found anomaly(with different symbols) in such columns:

* Wi-Fi
* BT
* JTAG
* Board
* iOSapp
* AndroidApp

For data preprocessing let’s complete Bad\_data with information from other tables.

BEGIN TRANSACTION

MERGE data\_bad T1

USING data1 T2

ON T1.Customer = T2.Customer

AND T1.Location = T2.Location

WHEN MATCHED THEN

UPDATE SET WiFi = T2.WiFi,

BT= T2.BT,

JTAG =T2.JTAG,

Board= T2.Board,

iOSapp =T2.iOSapp,

AndroidApp = T2.AndroidApp;

MERGE data\_bad T1

USING data2 T3

ON T1.Customer = T3.Customer

AND T1.Location = T3.Location

WHEN MATCHED THEN

UPDATE SET WiFi = T3.WiFi,

BT= T3.BT,

JTAG =T3.JTAG,

Board= T3.Board,

iOSapp =T3.iOSapp,

AndroidApp = T3.AndroidApp;

MERGE data\_bad T1

USING data3 T4

ON T1.Customer = T4.Customer

AND T1.Location = T4.Location

WHEN MATCHED THEN

UPDATE SET WiFi = T4.WiFi,

BT= T4.BT,

JTAG =T4.JTAG,

Board= T4.Board,

iOSapp =T4.iOSapp,

AndroidApp = T4.AndroidApp;

COMMIT TRANSACTION

After the merge of data, we still have lots of rows with unclear data. So lets remove this data.

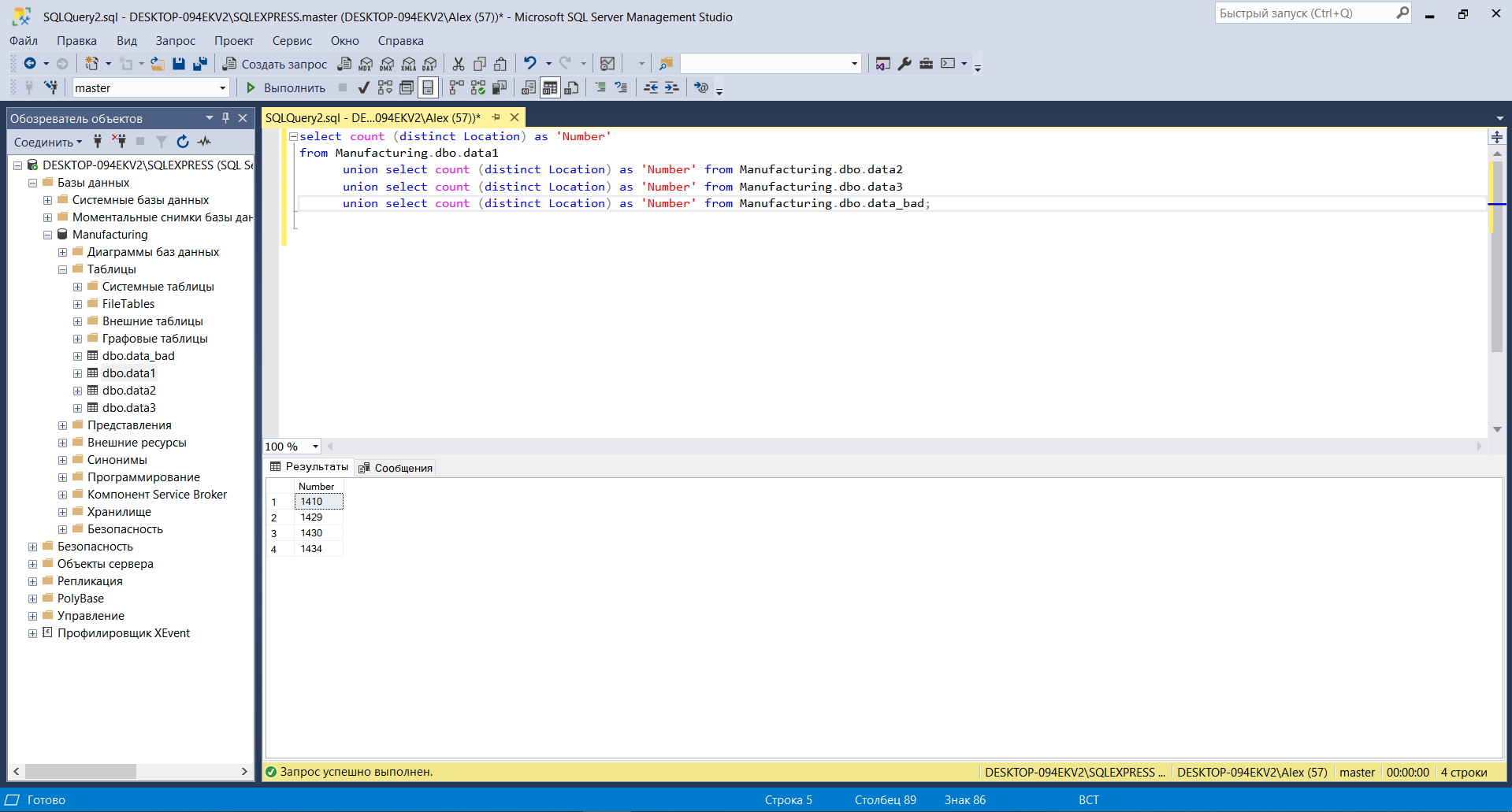
With the Replace function, we will remove unnecessary values. Example:

UPDATE data\_bad

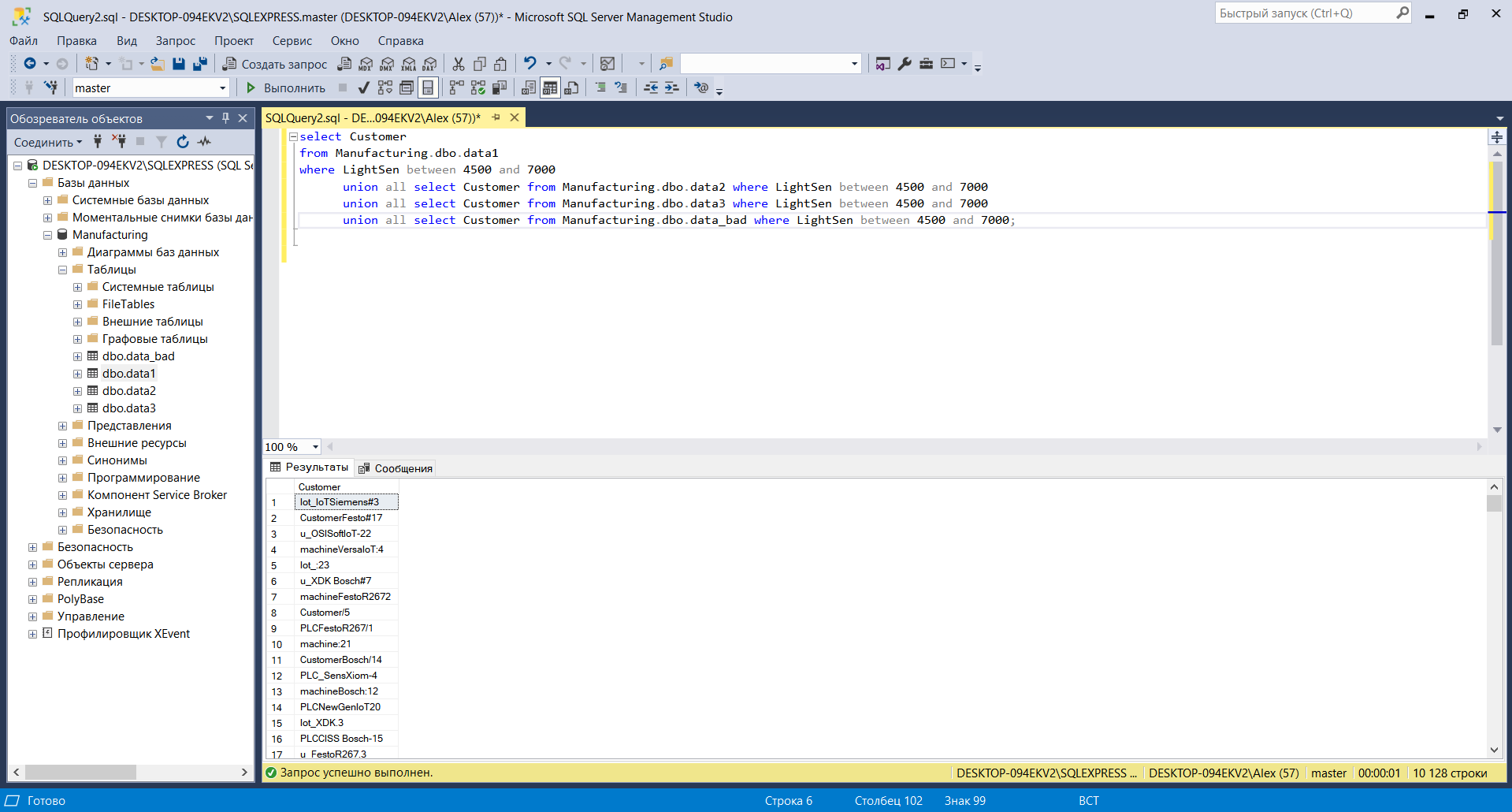
SET WiFi = REPLACE(WiFi, ';', '');

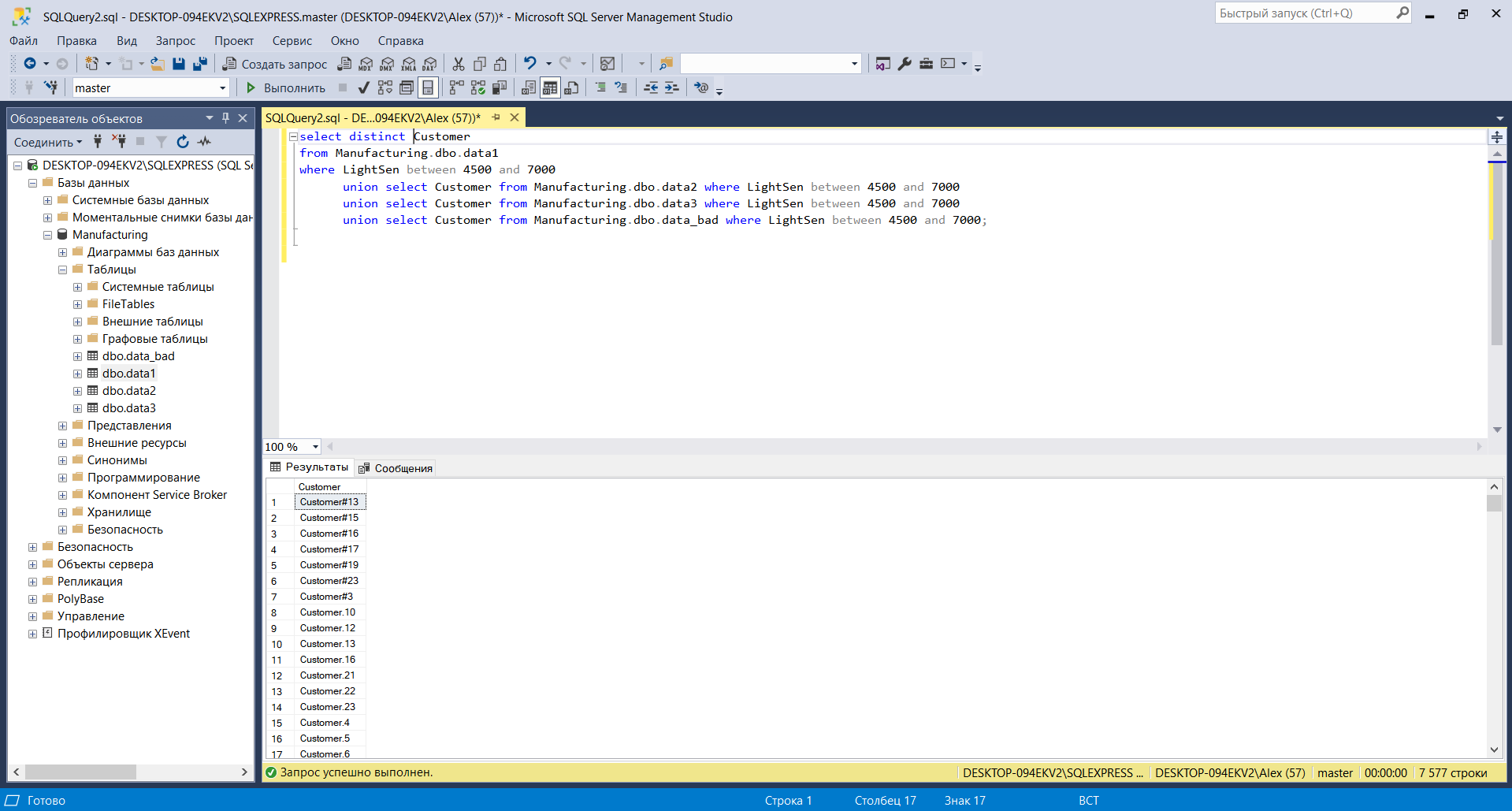
# Task 1.1

## Count the number of unique Locations.(Rudish)

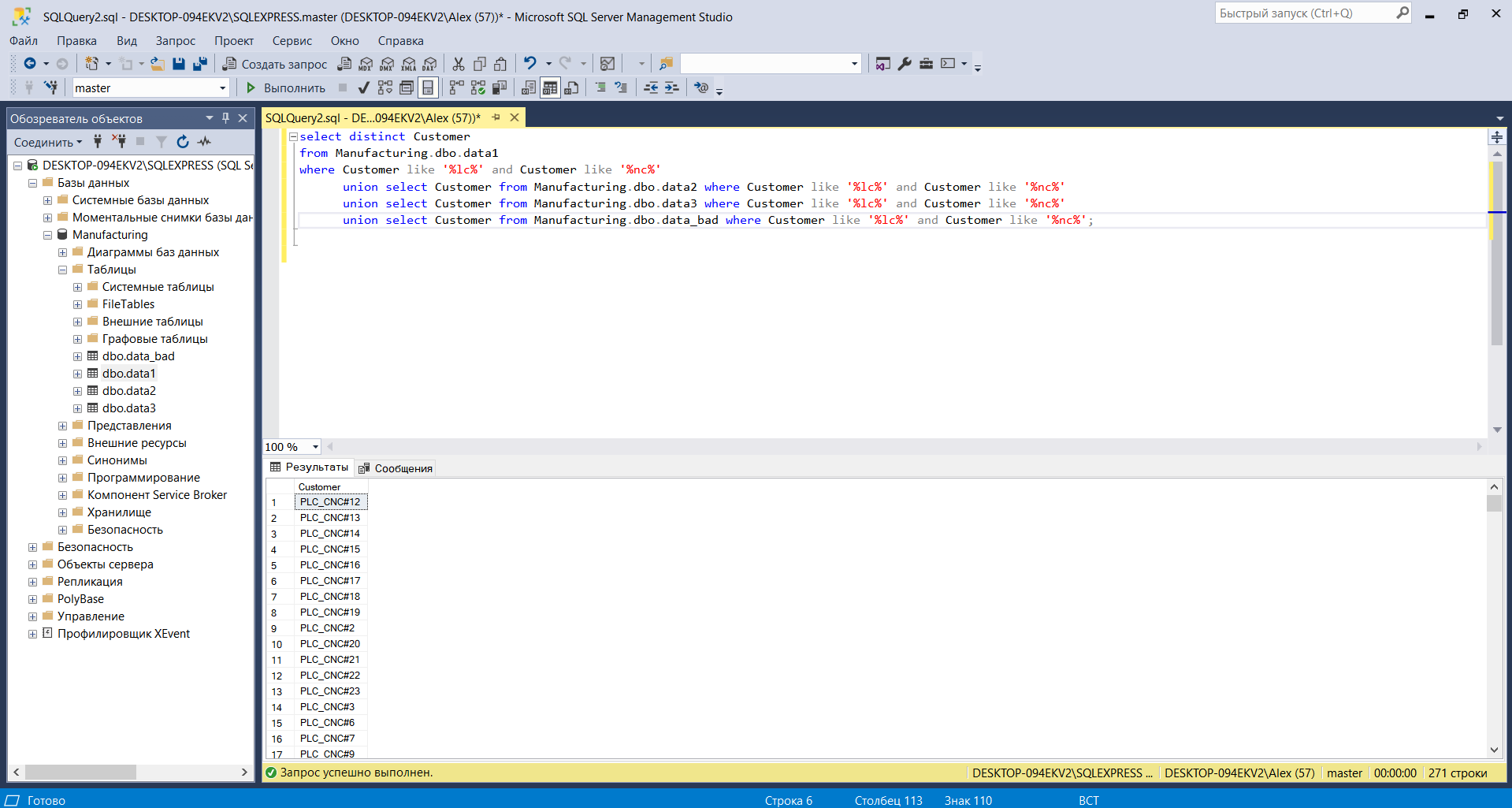


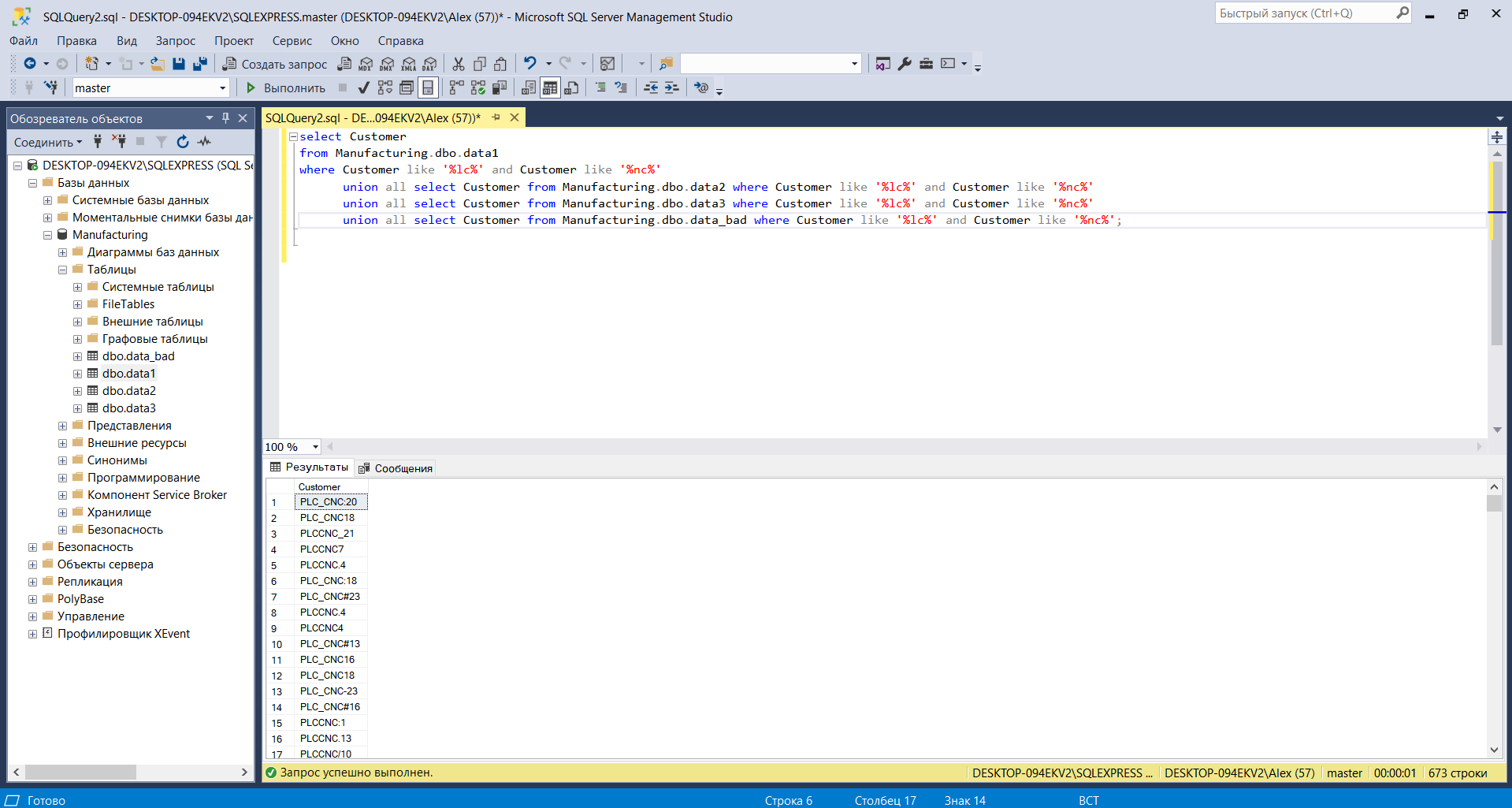
## Find all Customers whose LightSen is in the range between 4500-7000.(Rudish)



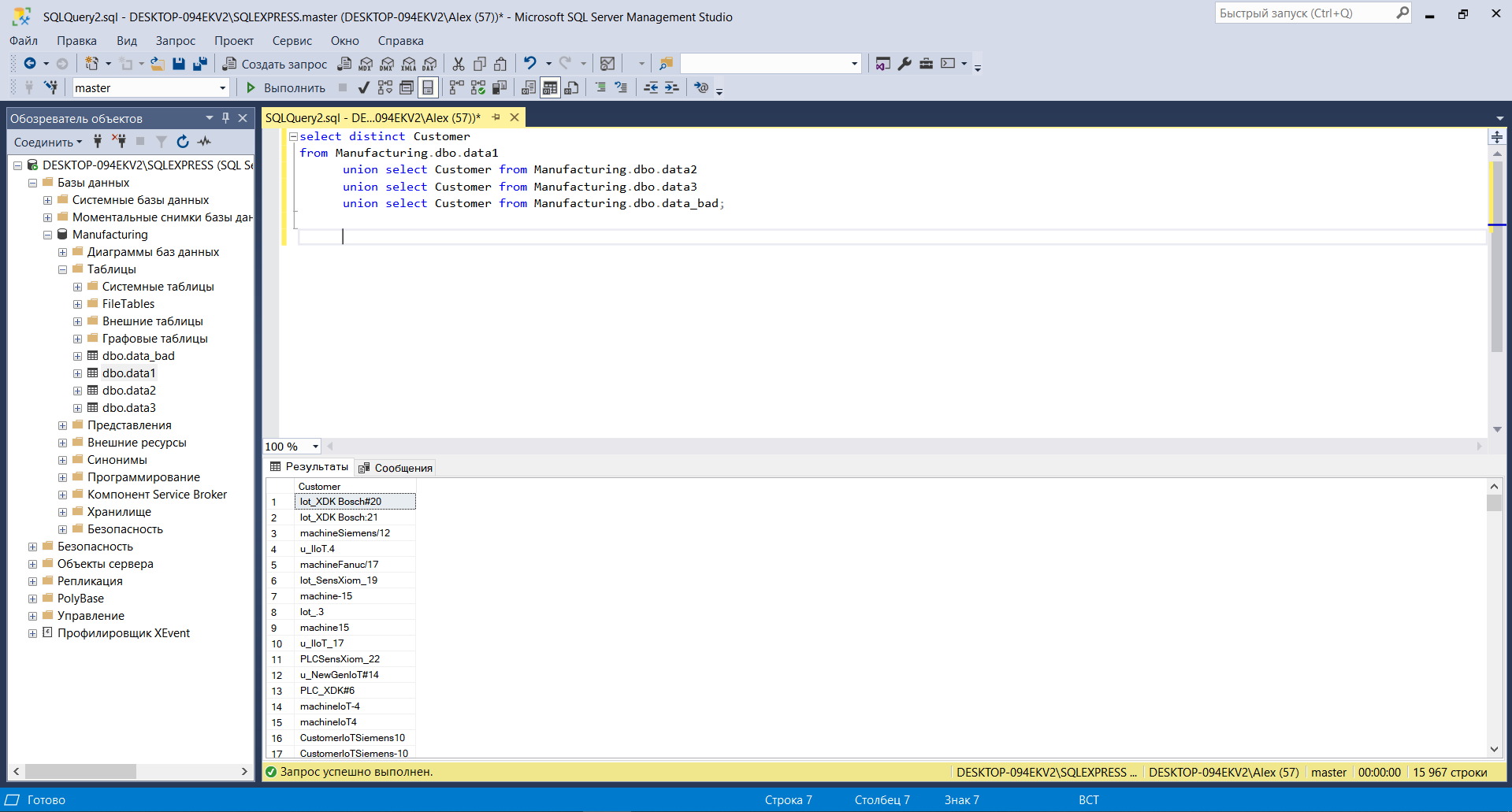


## Find all Customer that contains letters: LC and NC. (The letters go in this order, but between them there may be other letters and symbols. Letters are not case-sensitive).(Rudish)

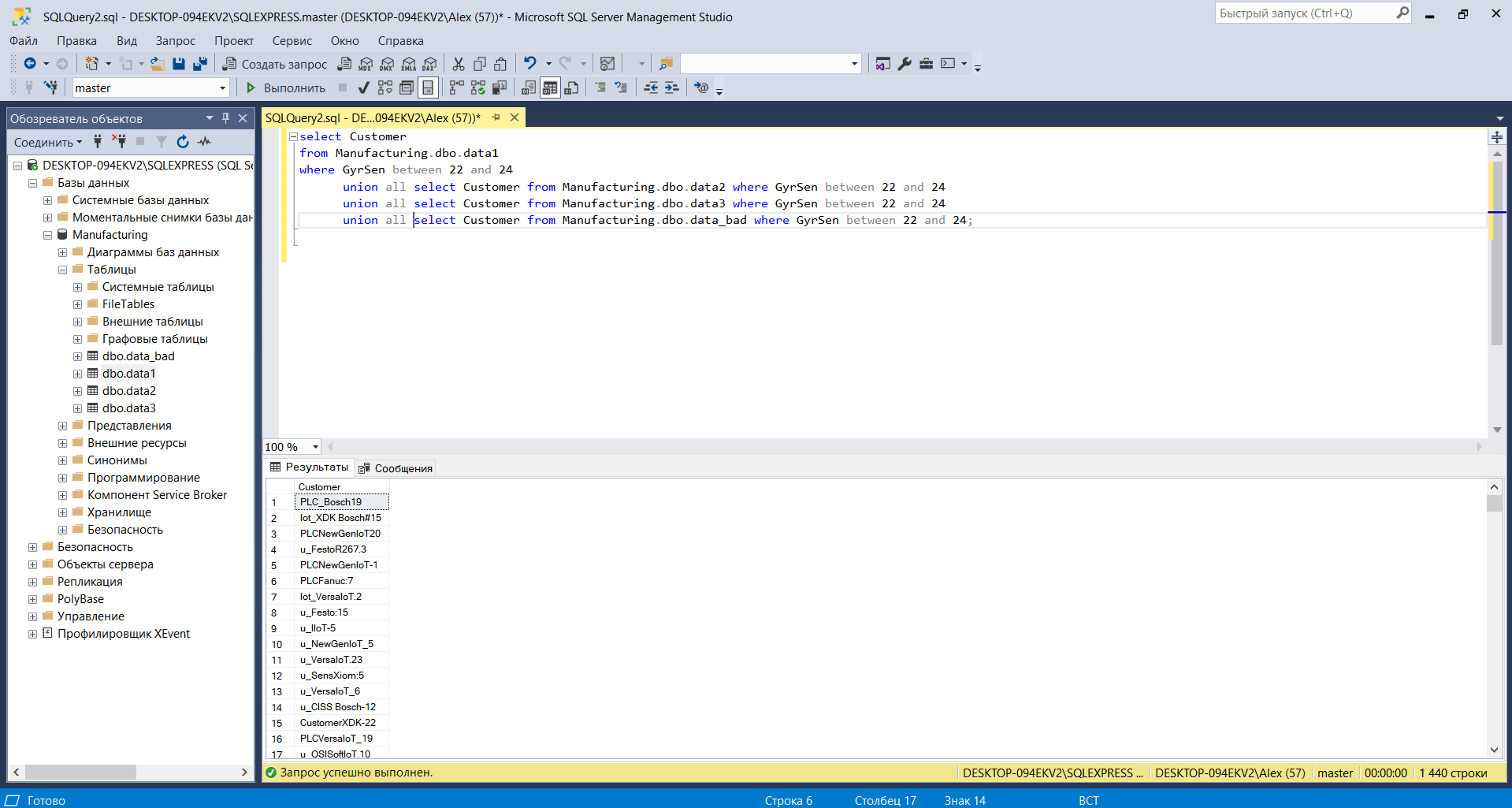


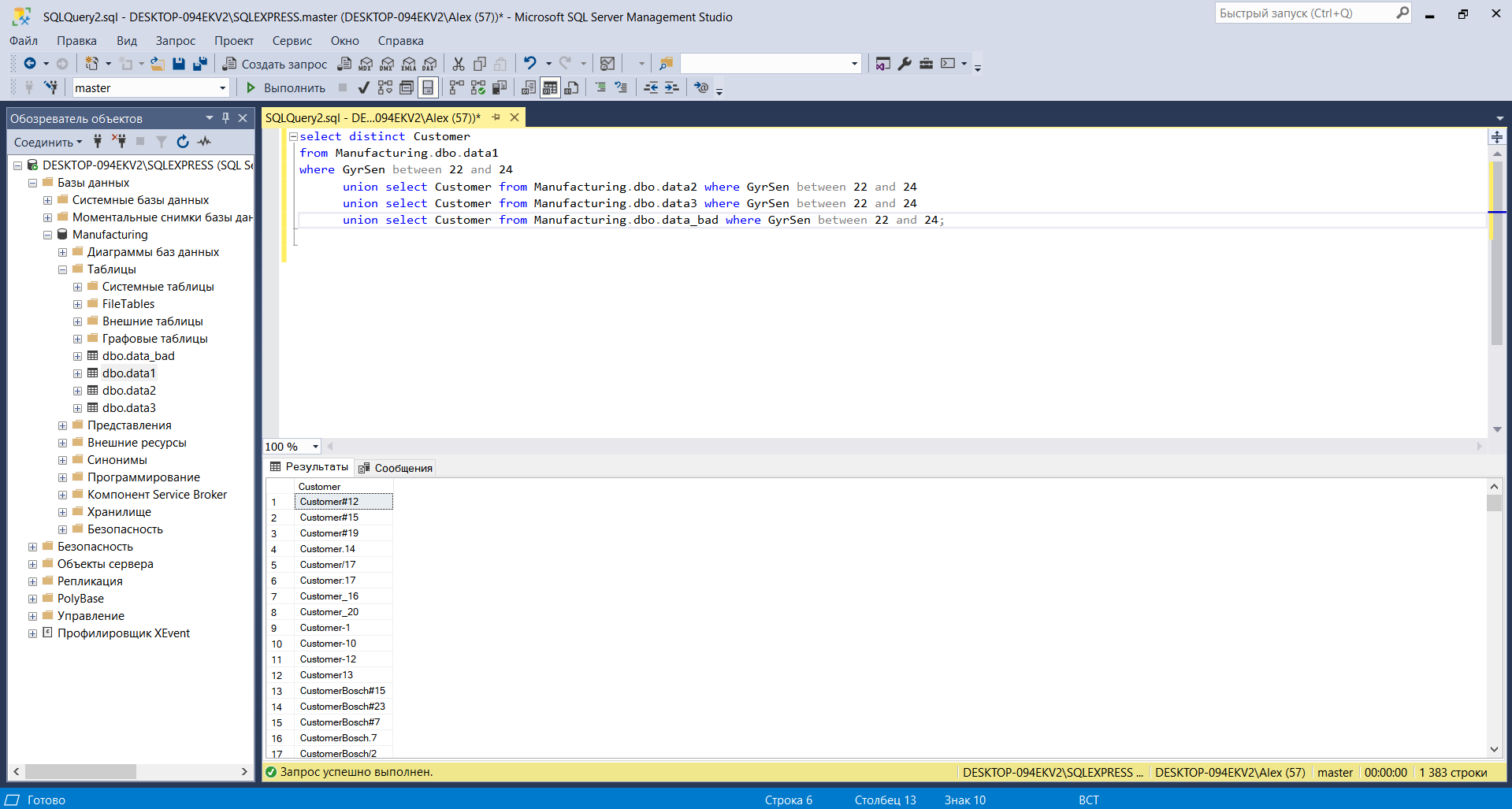


## Display a list of all unique customers. (Chudinova)



## Find all Customers with GyrSen in the range between 22-24.(Chudinova)





# Task 1.2

## Find all Customers with a Number of sensor in the range between 25-27, and for all devices with STM or ESP, regardless of digits further instead of Yes in the iOSApp column display "1", and for No is 0 (Uglovskaya)

SELECT Customer,

CASE WHEN iOSapp = 'Yes' THEN '1'

ELSE '0' END AS SupportIOSapp

FROM data\_bad

where (NumberOfSensors between 25 and 27) and (Board like 'ESP%' or Board like 'STM%')

union all SELECT Customer,

CASE WHEN iOSapp = 'Yes' THEN '1'

ELSE '0' END AS SupportIOSapp

FROM data2

where (NumberOfSensors between 25 and 27) and (Board like 'ESP%' or Board like 'STM%')

union all SELECT Customer,

CASE WHEN iOSapp = 'Yes' THEN '1'

ELSE '0' END AS SupportIOSapp

FROM data1

where (NumberOfSensors between 25 and 27) and (Board like 'ESP%' or Board like 'STM%')

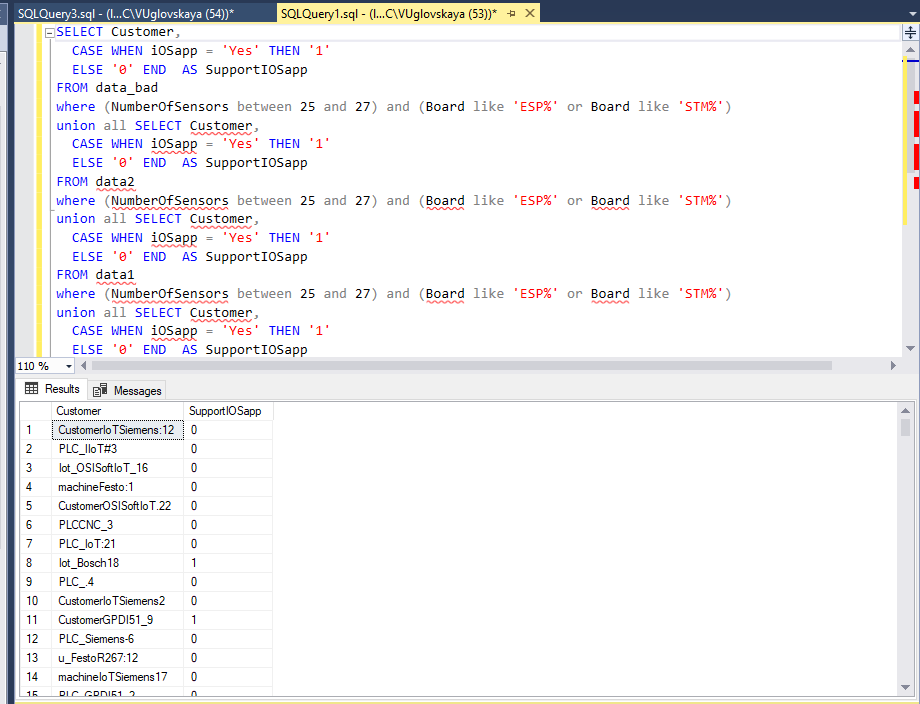
union all SELECT Customer,

CASE WHEN iOSapp = 'Yes' THEN '1'

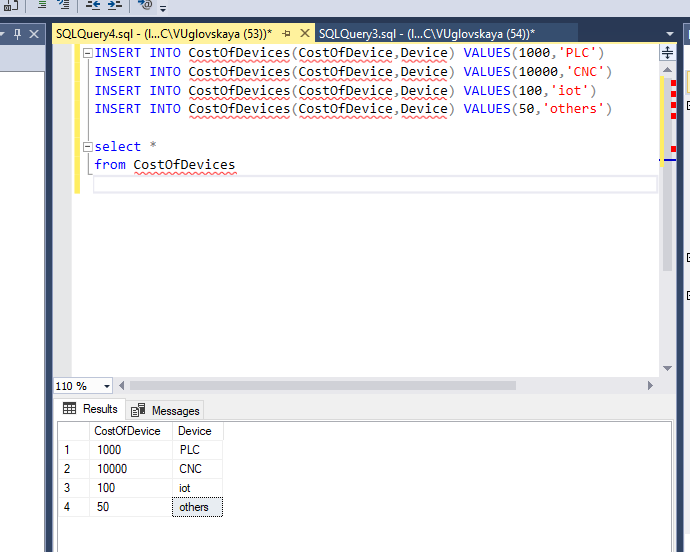
ELSE '0' END AS SupportIOSapp

FROM data3

where (NumberOfSensors between 25 and 27) and (Board like 'ESP%' or Board like 'STM%')



## Create a table price, which will indicate the cost of devices: for PLC – set 1000, CNC-10000, iot-100 for all the others arbitrarily (Chudinova)



INSERT INTO CostOfDevices(CostOfDevice,Device) VALUES(1000,'PLC')

INSERT INTO CostOfDevices(CostOfDevice,Device) VALUES(10000,'CNC')

INSERT INTO CostOfDevices(CostOfDevice,Device) VALUES(100,'iot')

INSERT INTO CostOfDevices(CostOfDevice,Device) VALUES(50,'others')

## Find all iosapp Customers that do not support wi-fi and update them to “yes” (Uglovskaya)

BEGIN TRANSACTION

update Data1

set WiFi = 'Yes'

where iOSapp = 'Yes'

update Data2

set WiFi = 'Yes'

where iOSapp = 'Yes'

update Data3

set WiFi = 'Yes'

where iOSapp = 'Yes'

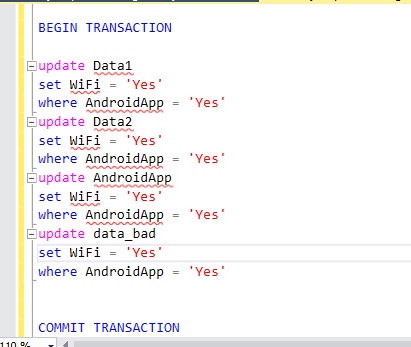
update data\_bad

set WiFi = 'Yes'

where iOSapp = 'Yes'

COMMIT TRANSACTION

We have done the same thing for the Android App



## Move all identifiers from customer and location to the corresponding tables and remove their mention in the name ( Uglovskaya)

BEGIN TRANSACTION;

CREATE TABLE customers (

id int IDENTITY (1, 1),

name varchar(500),

);

INSERT INTO Customers (Name)

SELECT distinct(Customer)

FROM Data1

union all SELECT distinct(Customer)

FROM Data2

union all SELECT distinct(Customer)

FROM Data3

union all SELECT distinct(Customer)

FROM data\_bad

CREATE TABLE locations (

id int IDENTITY (1, 1),

name varchar(500));

select \*

from locations

INSERT INTO locations (Name)

SELECT distinct(location)

FROM Data1

union all SELECT distinct(location)

FROM Data2

union all SELECT distinct(location)

FROM Data3

union all SELECT distinct(location)

FROM data\_bad

CREATE TABLE customers (

id int IDENTITY (1, 1),

name varchar(500),

locationId varchar(30)

);

update Data1

set Data1.clientid = customers.id

from Data1

inner join customers on

Data1.customer = customers.name

SELECT distinct(Customer)

FROM Data1

union all SELECT distinct(Customer)

FROM Data2

union all SELECT distinct(Customer)

FROM Data3

union all SELECT distinct(Customer)

FROM data\_bad

ALTER TABLE data1

ADD clientid INT NULL

ALTER TABLE data2

ADD clientid INT NULL

ALTER TABLE data3

ADD clientid INT NULL

ALTER TABLE data\_bad

ADD clientid INT NULL

ALTER TABLE data1

ADD Locationid INT NULL

ALTER TABLE data2

ADD Locationid INT NULL

ALTER TABLE data3

ADD Locationid INT NULL

ALTER TABLE data\_bad

ADD Locationid INT NULL

update Data1

set Data1.Locationid = locations.id

from Data1

inner join locations on

Data1.Location = locations.name

update Data1

set Data1.clientid = customers.id

from Data1

inner join customers on

Data1.customer = customers.name

update Data2

set Data2.Locationid = locations.id

from Data1

inner join locations on

Data2.Location = locations.name

update Data2

set Data2.clientid = customers.id

from Data2

inner join customers on

Data2.customer = customers.name

update Data3

set Data3.clientid = customers.id

from Data3

inner join customers on

Data3.customer = customers.name

update data\_bad

set data\_bad.Locationid = locations.id

from data\_bad

inner join locations on

data\_bad.Location = locations.name

update data\_bad

set data\_bad.clientid = customers.id

from data\_bad

inner join customers on

data\_bad.customer = customers.name

COMMIT TRANSACTION

# Task 2

- Egor Nikishin: primary data analysis and visual anomaly detection

- Andrey Chubin: automated anomaly detection based on distance

- Anastasiia Kuznetsova: automated anomaly detection based on machine learning approach (Isolation Forest)

GITHUB: <https://github.com/suova/Manufacturing-Data-Collection-and-Analytics/blob/main/DC_HW.ipynb>