**D1. Python Challenge: -**

Python Basics

1. Create python script kaggle titanic challenge

2. Read csv file

3. Write into two separate files: -

1. Reverse order of columns ABCD - DCBA

2. Second all the columns as it

Download train.csv and test.csv from below link and keep it in PythonApplication21.py folder

Input Link from Kaggle :- <https://www.kaggle.com/c/titanic/data>

**Script: -**

import pandas as pd

import os

import datetime

import shutil

import csv

class loader (object) :

# constructor

def \_\_init\_\_(self,\*args):

if args[0].endswith('.csv'):

self.in\_filename = args[0]

# reverse filename

self.out\_filename ="Reverse-Copy-"+str(datetime.date.today())+"-"+self.in\_filename

print('Rerse data filename :-',self.out\_filename)

#copy of filename

self.out\_filename\_copy= "Copy-"+str(datetime.date.today())+"-"+self.in\_filename

print('copy of file', self.out\_filename\_copy)

else :

raise Exception('Incorrect file format')

# function to load data

def load(self) :

try :

# checking if file exists

if os.path.isfile(self.in\_filename) :

with open(self.in\_filename) as csvFile :

reader = csv.reader(csvFile)

data = [row for row in reader]

# create copy of file

shutil.copy(self.in\_filename, self.out\_filename\_copy)

# reverse the colunms of file

with open(self.out\_filename, 'w+', newline='') as csvFileW:

lines = csv.writer(csvFileW)

for row in data:

lines.writerow(row[::-1])

csvFileW.close()

else :

raise()

except :

print('Please check if file exists...')

#main function

def main () :

in\_filename = input('Enter input file name :- ')

obj = loader(in\_filename)

obj.load()

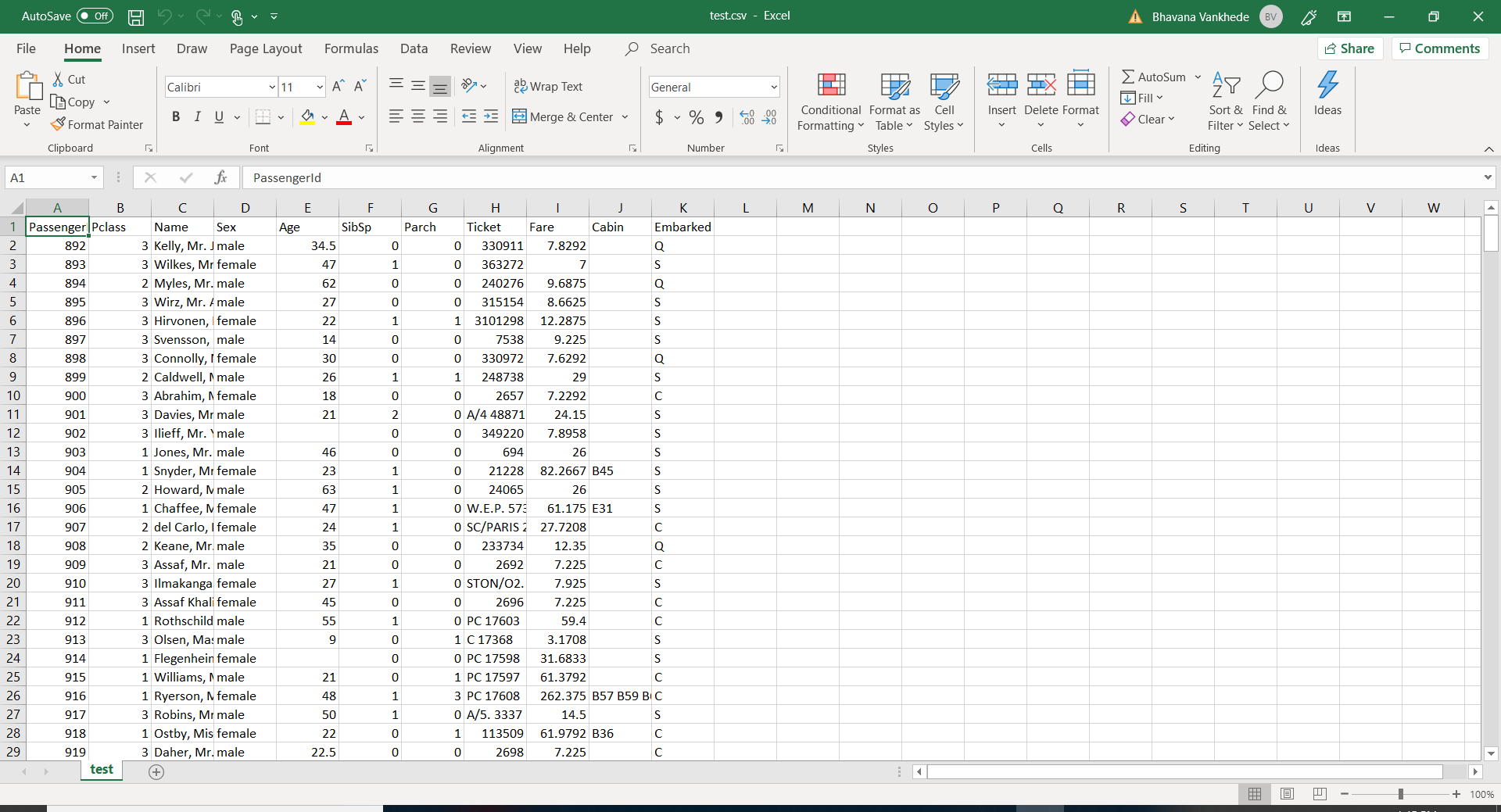
if \_\_name\_\_ =="\_\_main\_\_" :

main()

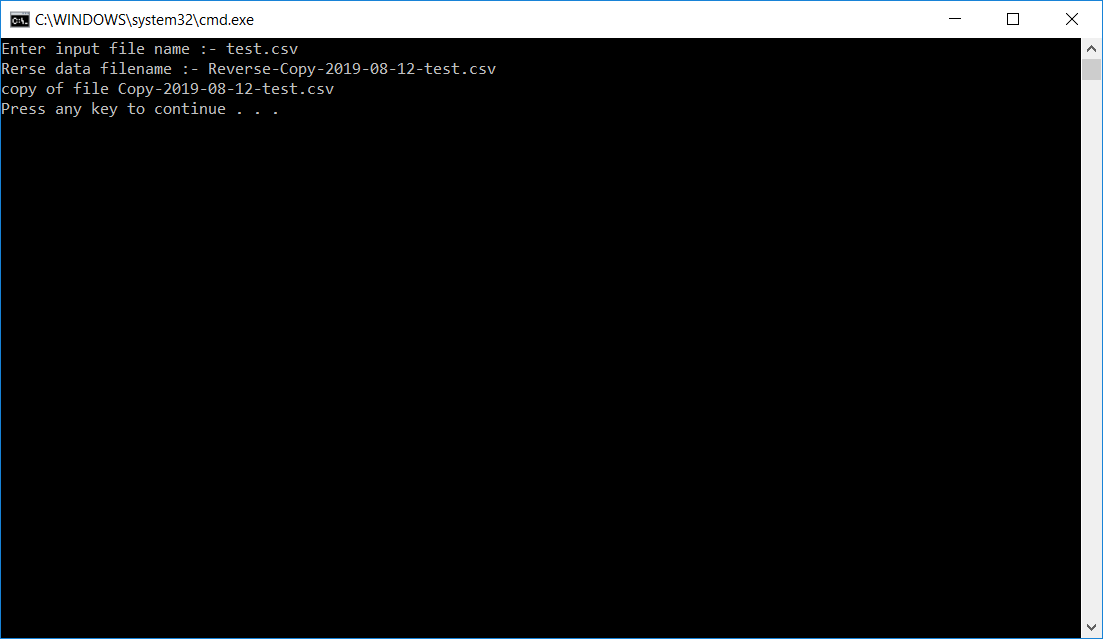
**Input File Attachment: -**

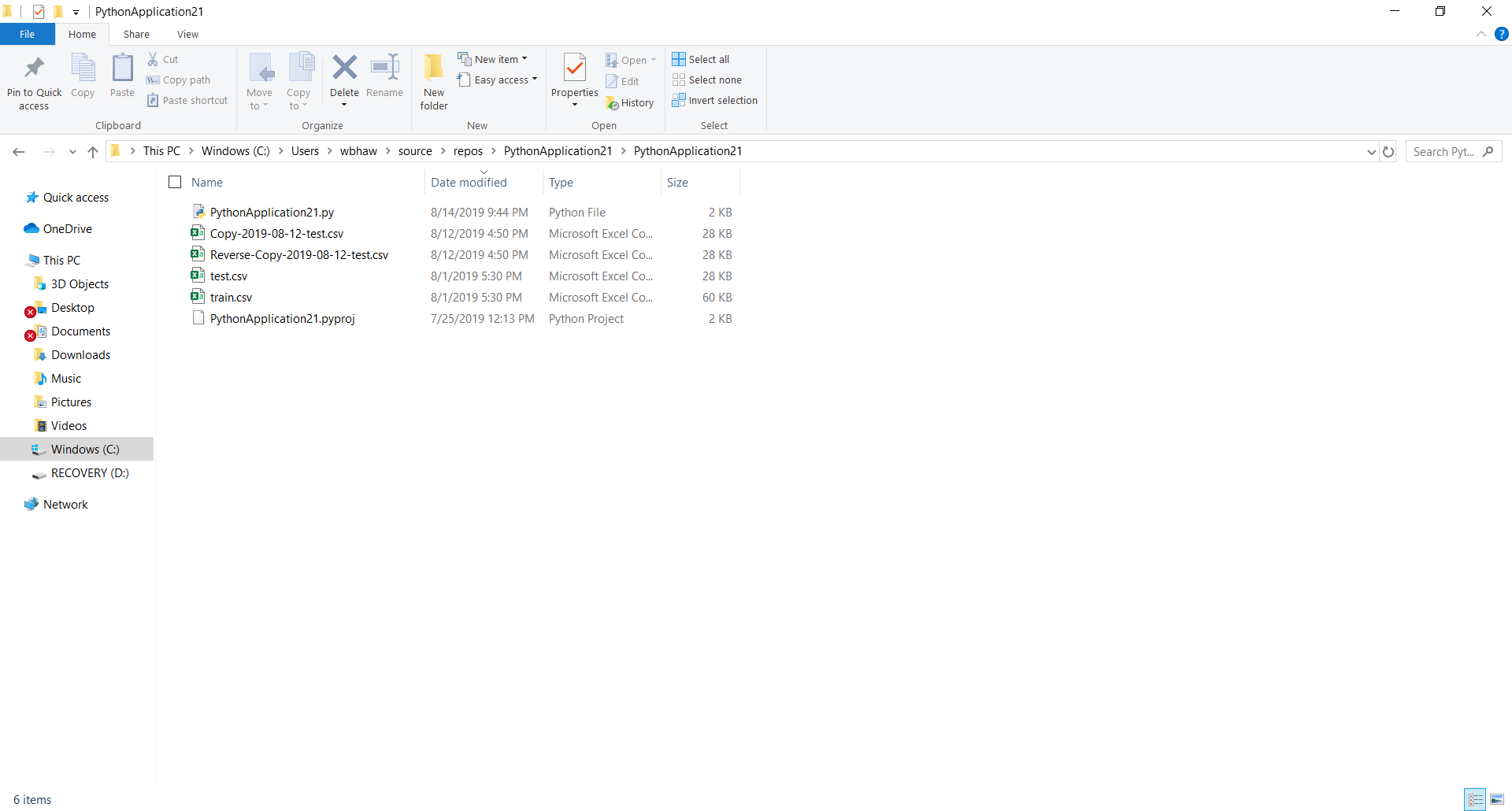


**Original file: -**

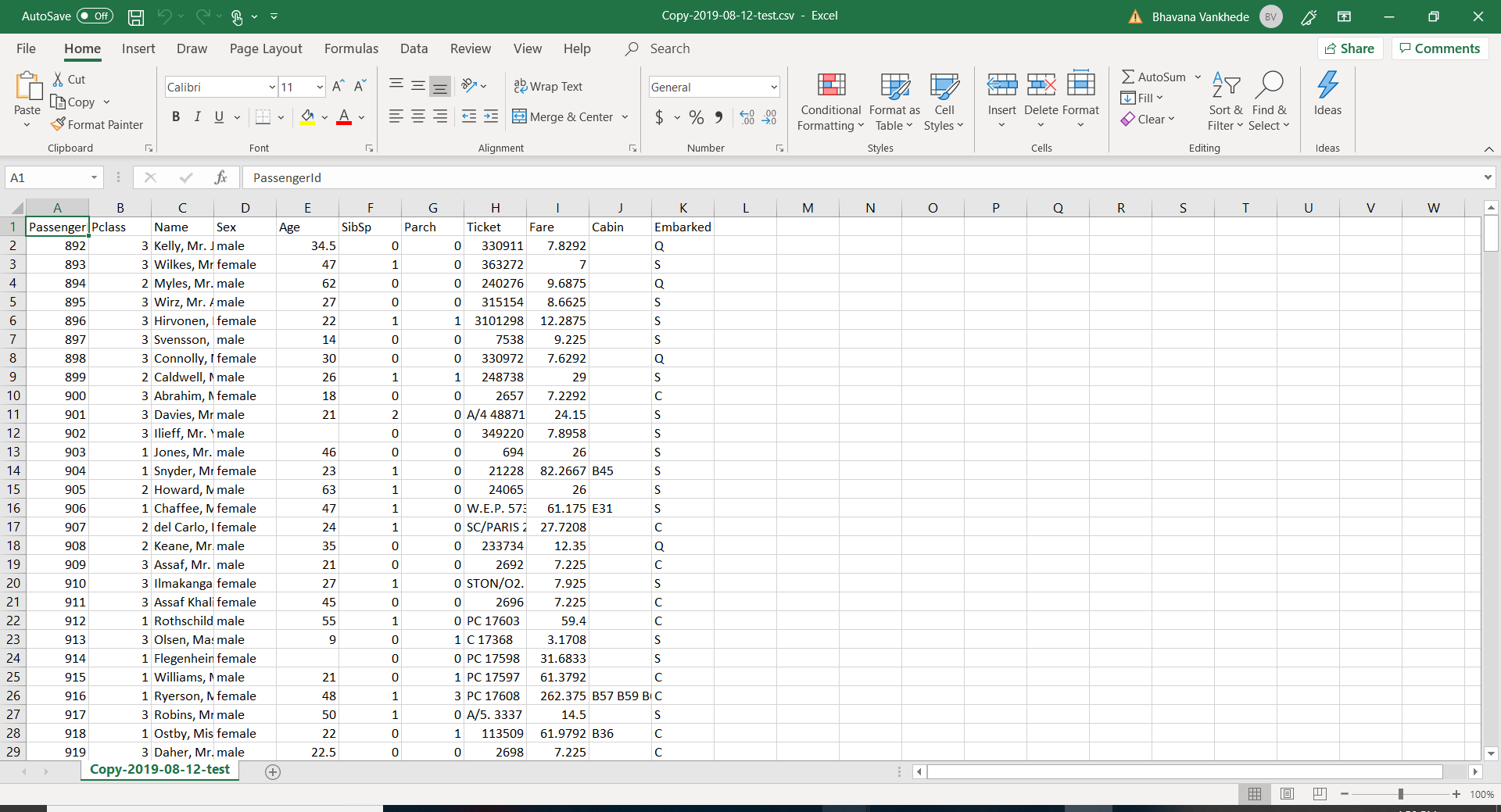


**Output: -**

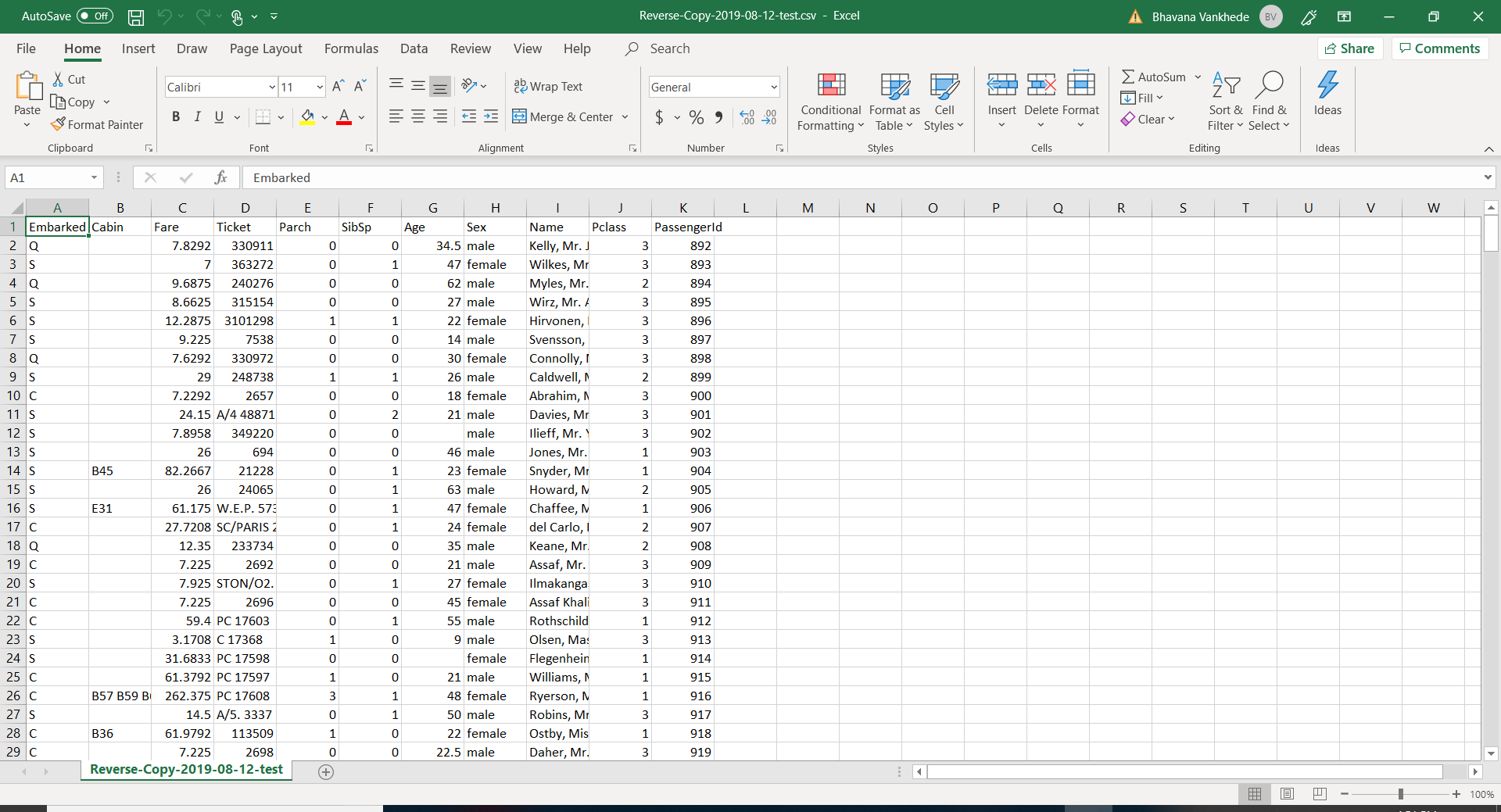




**Copy of file: -**



**Reverse File:-**



**D2. Flask Challenge: -**

Pull data from alphavantage.co using API query use time series data and apply Microsoft code load that data into excel file. Create flask UI to download data from mentioned server using cloud.

Please refer to folder structure for more details.s

**Link :-**

<http://flask-env1.uj3ttyczem.us-east-1.elasticbeanstalk.com/application>

**Script:-**

**Views.py**

"""

Routes and views for the flask application.

"""

from datetime import datetime

from flask import render\_template, request

from FlaskWebProject1 import application

@application.route('/')

@application.route('/home')

def home():

"""Renders the home page."""

return render\_template(

'index.html',

title='Home Page',

year=datetime.now().year,

)

@application.route('/contact')

def contact():

"""Renders the contact page."""

return render\_template(

'contact.html',

title='Contact',

year=datetime.now().year,

message='Your contact page.'

)

@application.route('/about')

def about():

"""Renders the about page."""

if request.method == 'POST' :

if request.form['submit\_button'] == 'submit' :

pass

return render\_template(

'about.html',

title='About',

year=datetime.now().year,

message='Your application description page.'

)

@application.route('/application')

def application():

"""Renders the about page."""

return render\_template(

'application.html',

title='Application',

year=datetime.now().year,

message='Hello World!'

)

**Application.html**

{% extends "layout.html" %}

{% block content %}

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title>Hello World</title>

</head>

<body>

<h6> Hello World ! </h6>

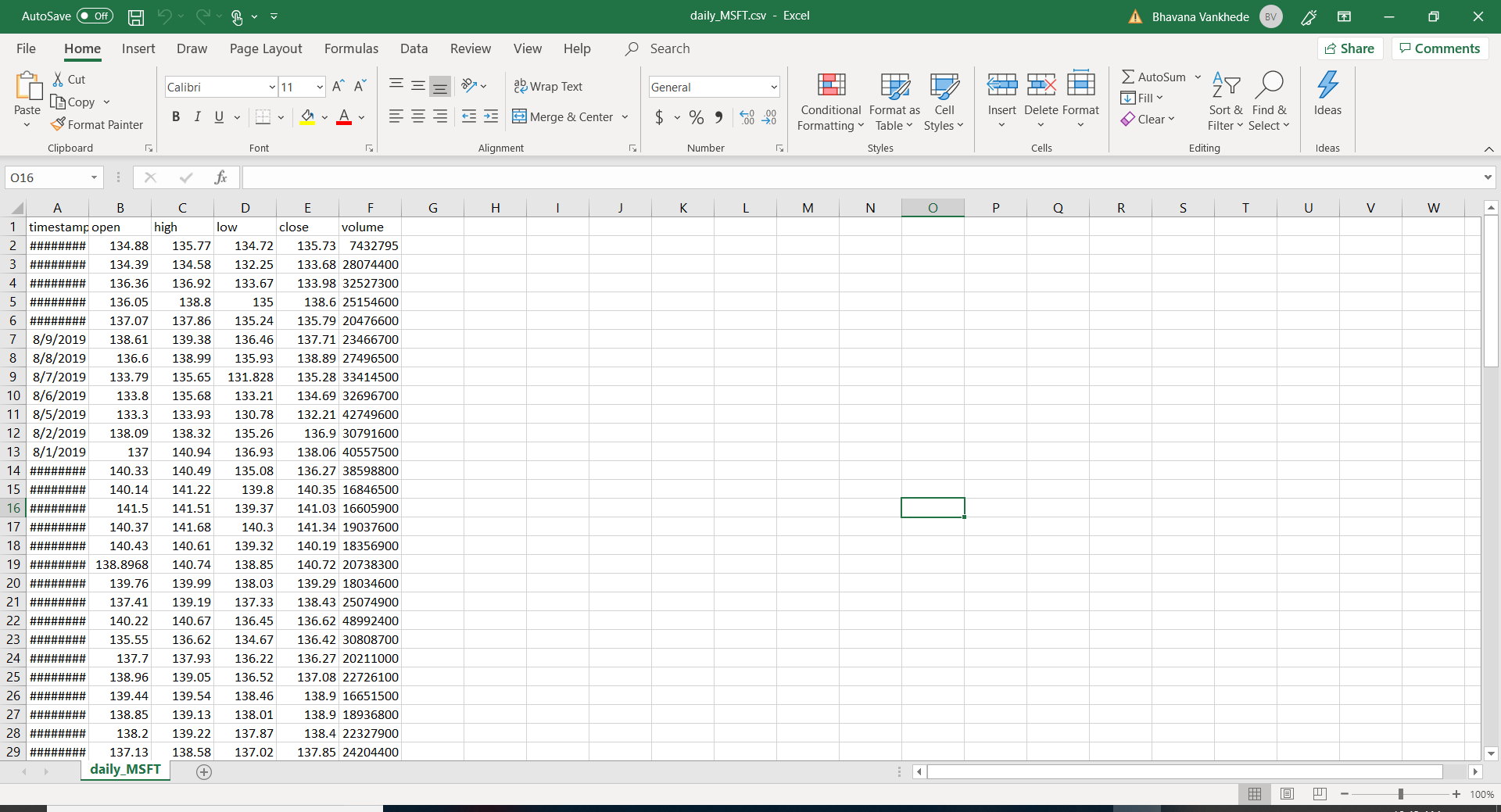
<input type="button" onclick="parent.location ='https://www.alphavantage.co/query?function=TIME\_SERIES\_DAILY&symbol=MSFT&apikey=demo&datatype=csv'" value="Download">

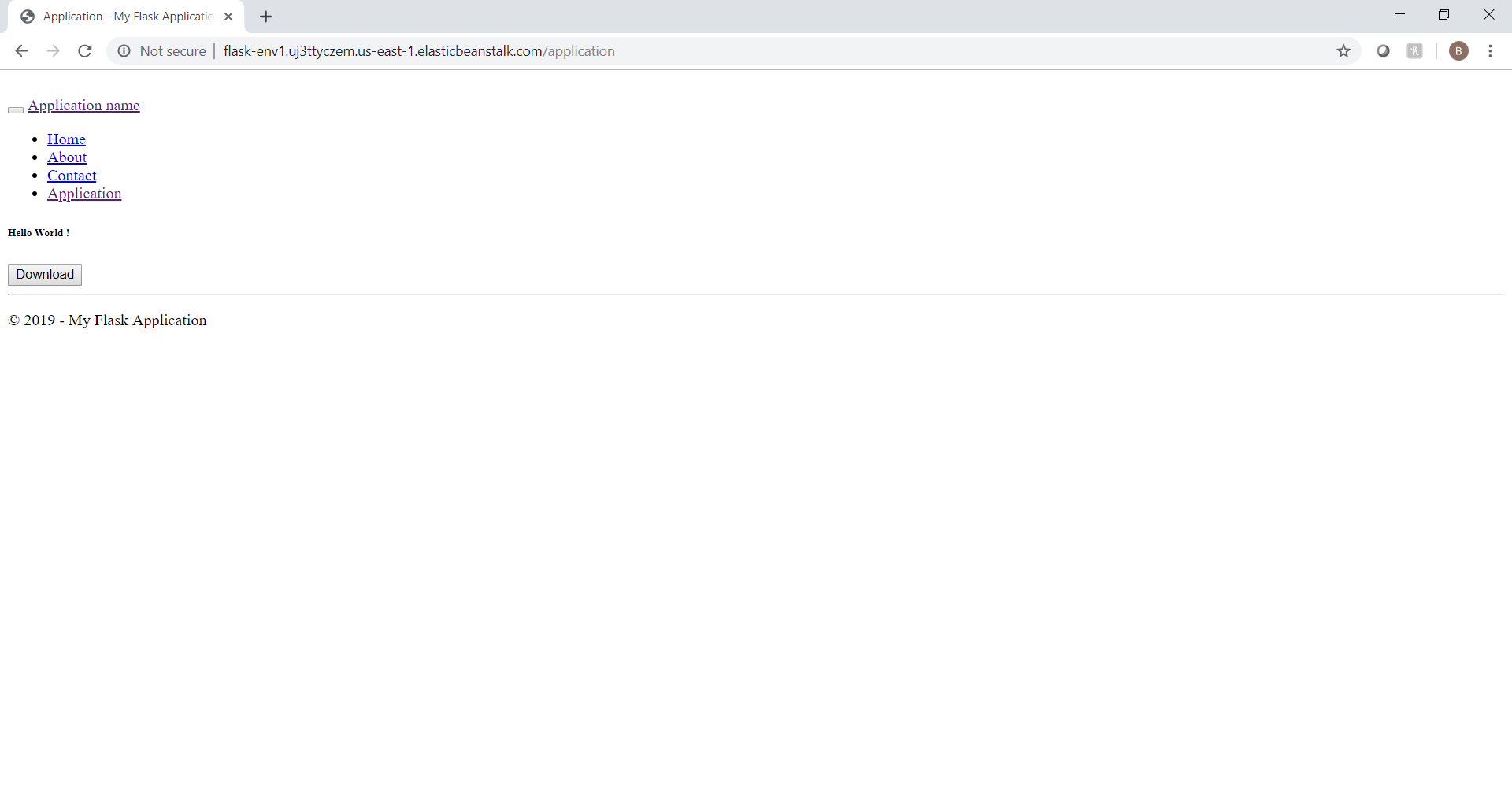
</body>

</html>

{% endblock %}

**Output :-**





**D3. Java Challenge: -**

Create hello world application in eclipse in program demonstrate breakpoint, add option to force exception and it should what causing exception from stack traces

**Script:-**

**public** **class** HelloWorld {

**public** **static** **void** main(String args[]) {

**try** {

System.***out***.println ("Hello World");

**throw** **new** IndexOutOfBoundsException();

}

**catch**(IndexOutOfBoundsException e) {

e.printStackTrace();

}

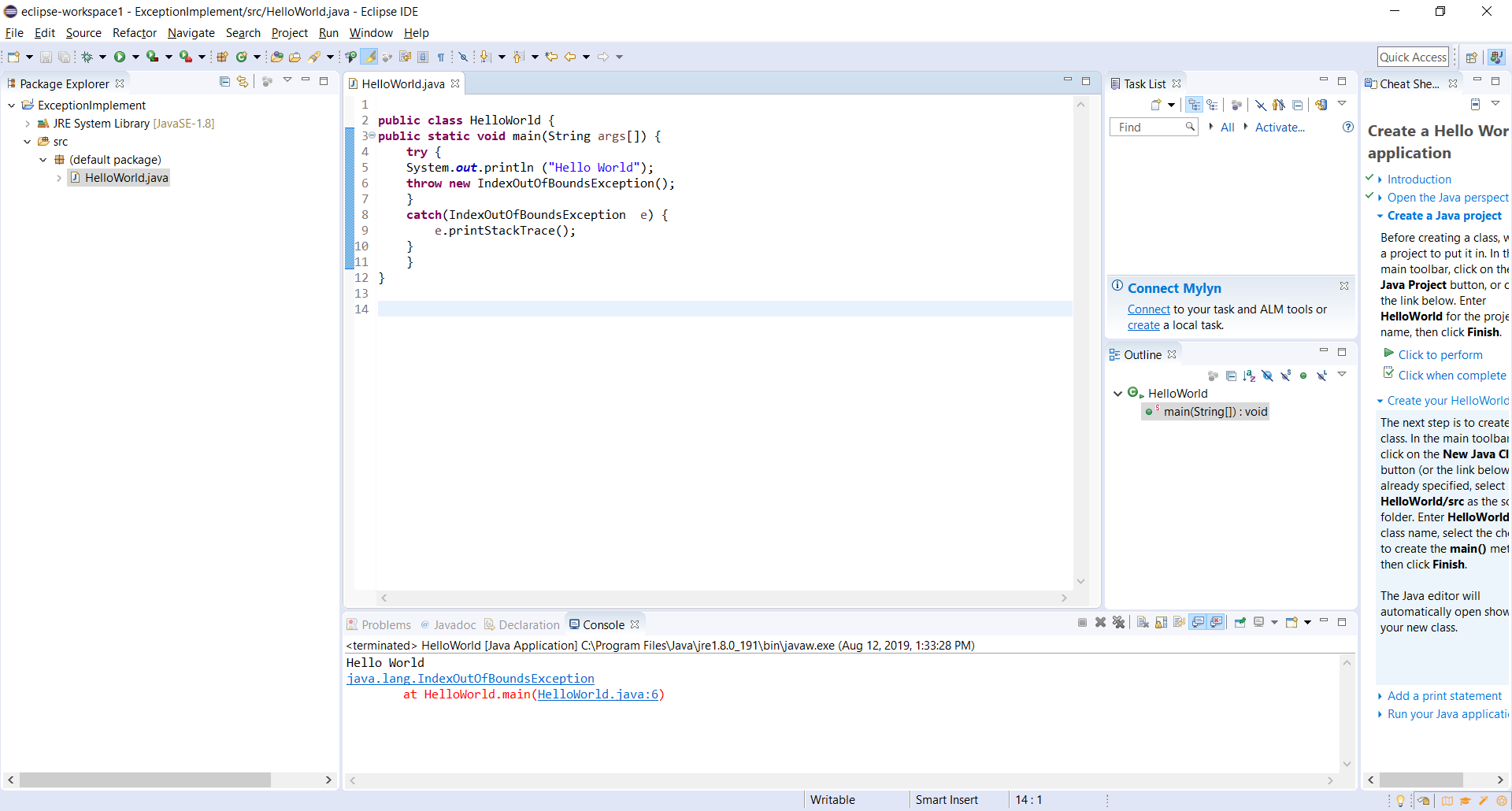
}

}

**File Attachment:-**

****

**Output:-**



**D4. Shell Script Challenge: -**

Create shell script to create ec2 instance on AWS

**Script:-**

#!/bin/bash

echo "\*\*\* Please Enter AWS Details to Launch EC2 Instance\*\*\*"

read -p "Access id " access\_key

read -p "Secret Key " access\_secret\_key

read -p "Instance Image ID " image\_id

read -p "Instance Type " instance\_type

read -p "Region " region

#check input

if [[ -z "$access\_key" && "$access\_secret\_key" && "$image\_id" && "$instance\_type" && "$region" ]] ; then

echo "Instance creation failed, please enter all details correctly ..."

exit 1

else

# setting up environment variables

aws configure set aws\_access\_key\_id $access\_key

aws configure set aws\_secret\_access\_key $access\_secret\_key

aws configure set region $region

# create ec2 instance

aws ec2 run-instances --image-id $image\_id --instance-type $instance\_type --region $region

echo "Instance created sucessfully, please verify ..."

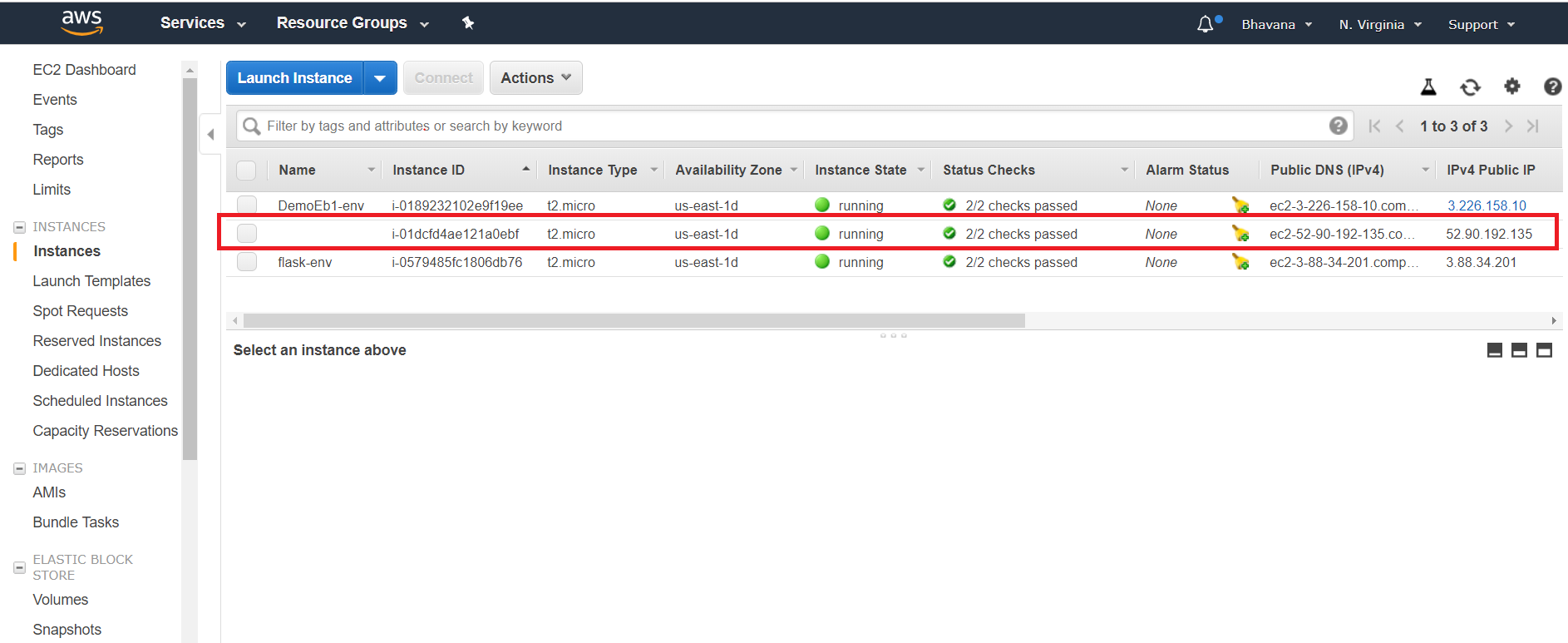
exit 0

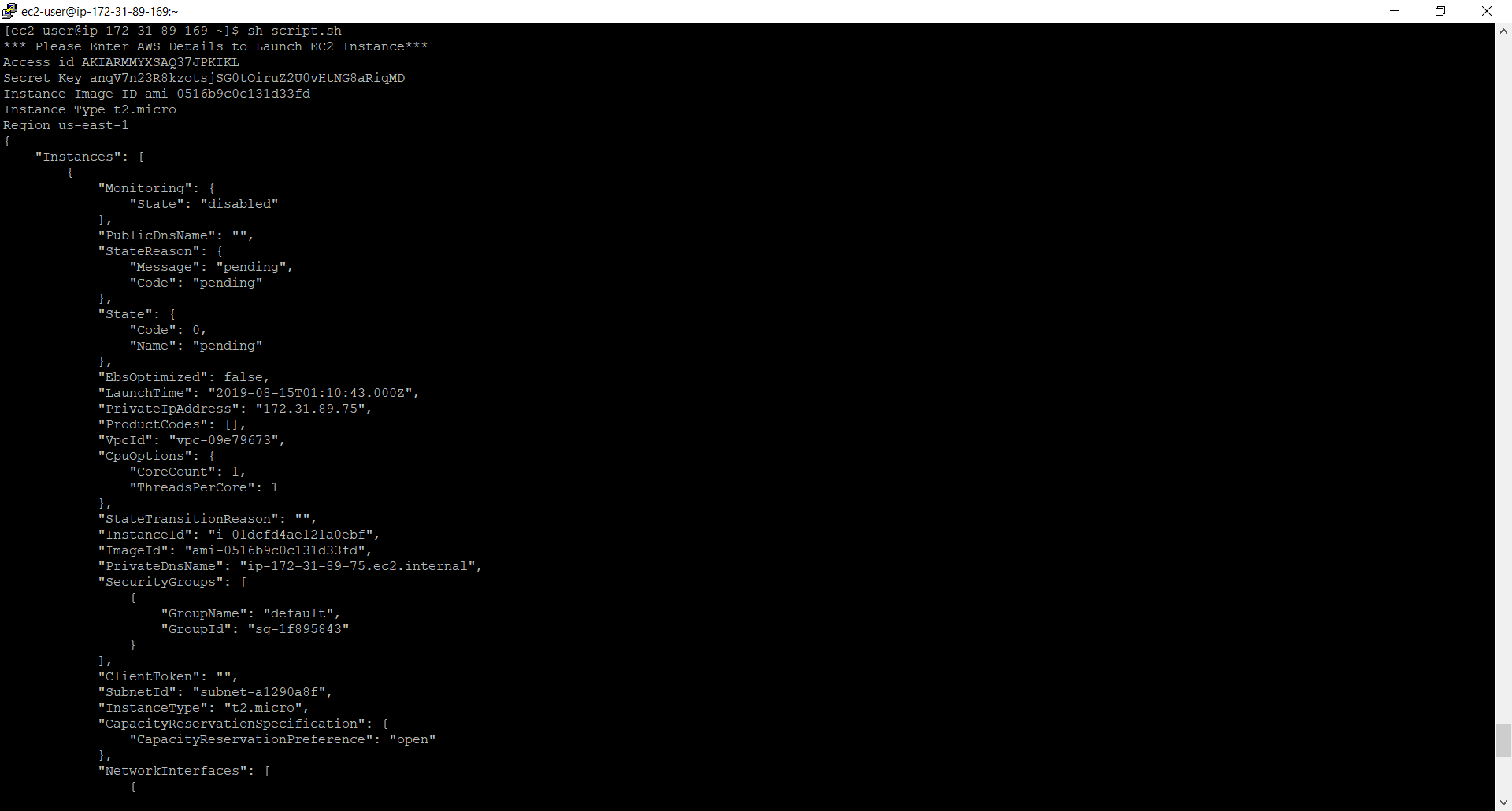
fi

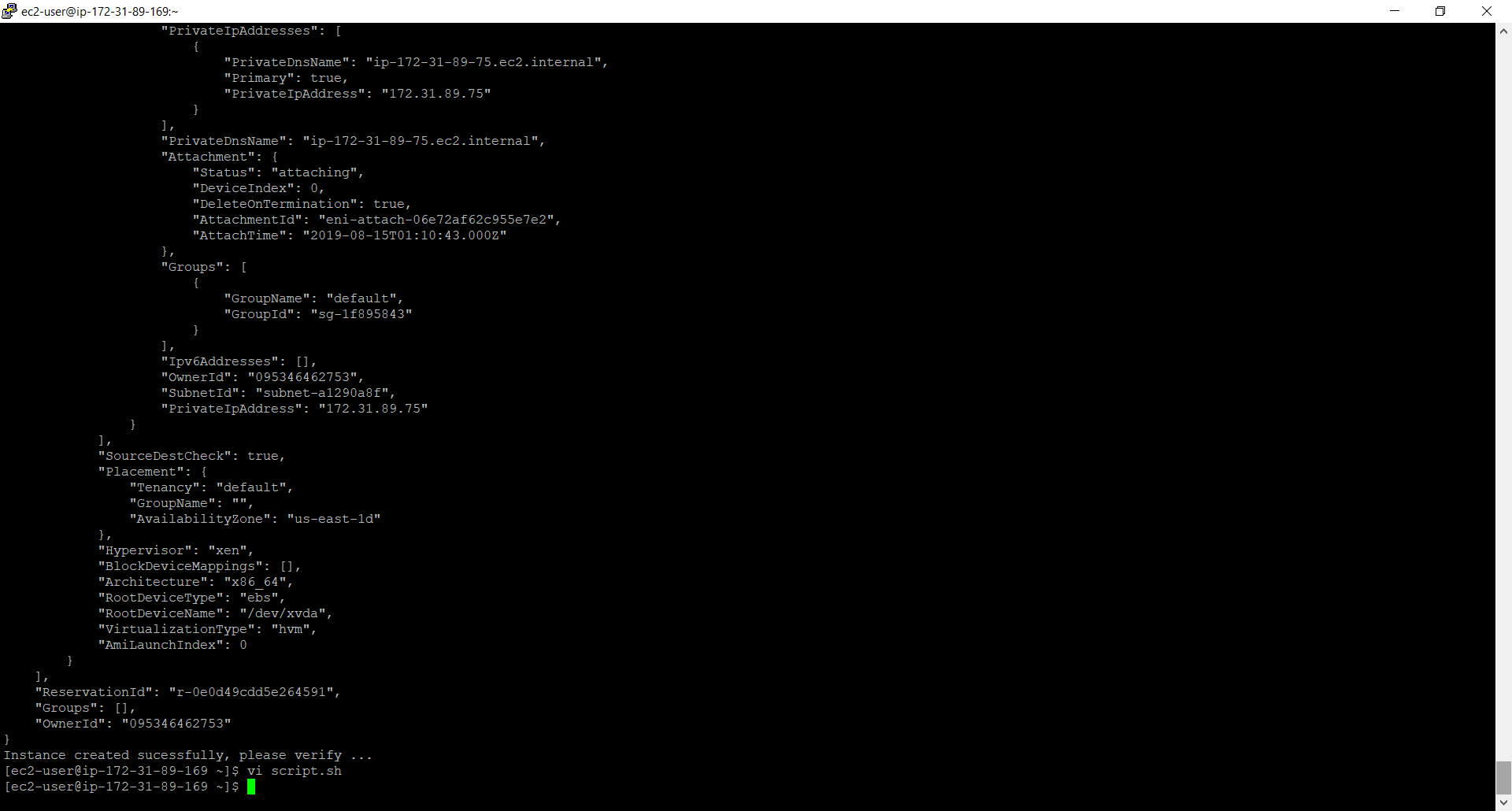
**File Attachment:-**

****

**Output :- ( Highlighted RED instance is created by shell script)**







**s**

**D5. SQL Challenges: -**

1. Create employee and department tables, calculate average employee salary

**Script :-**

create table department (

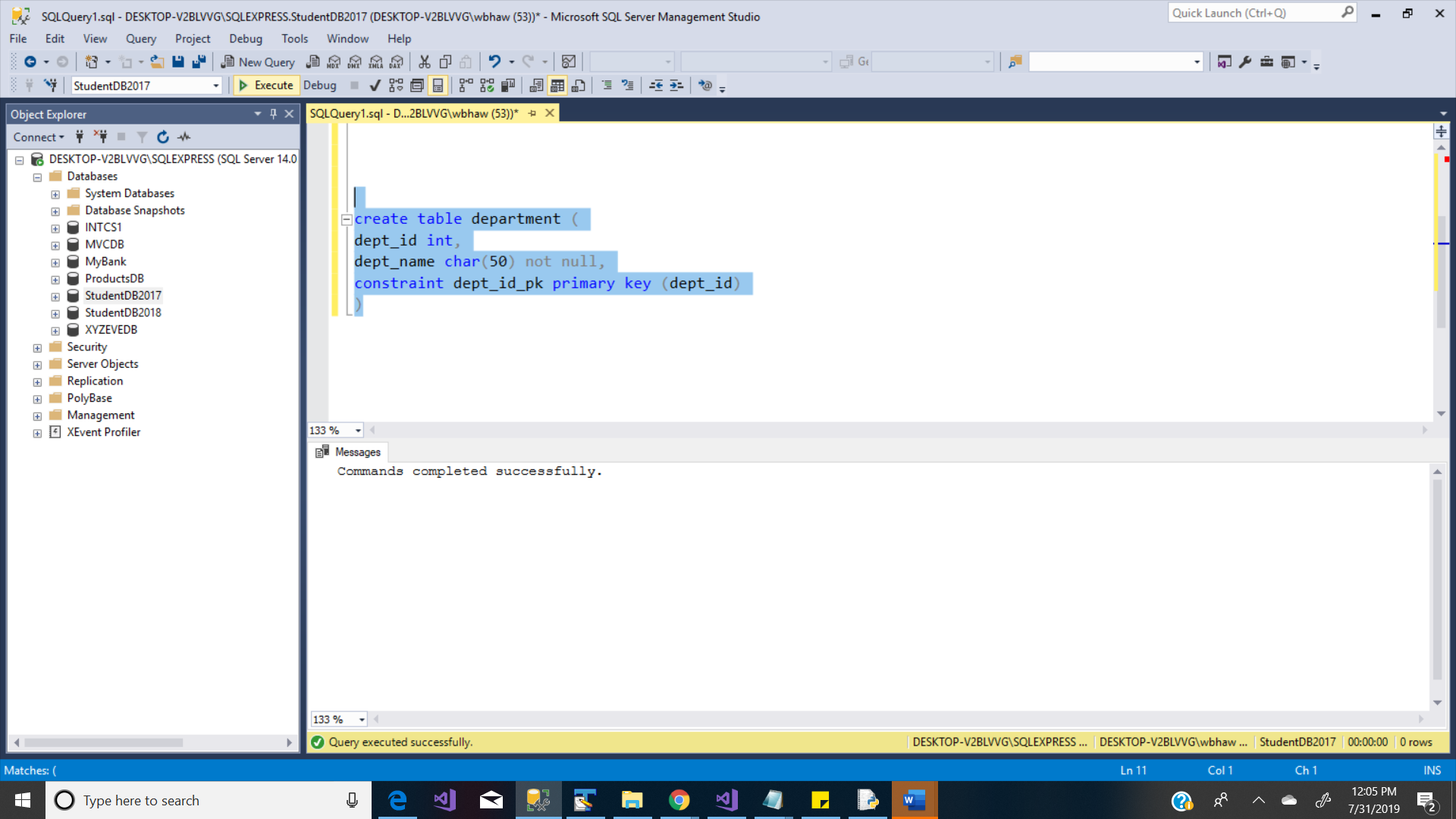
dept\_id int,

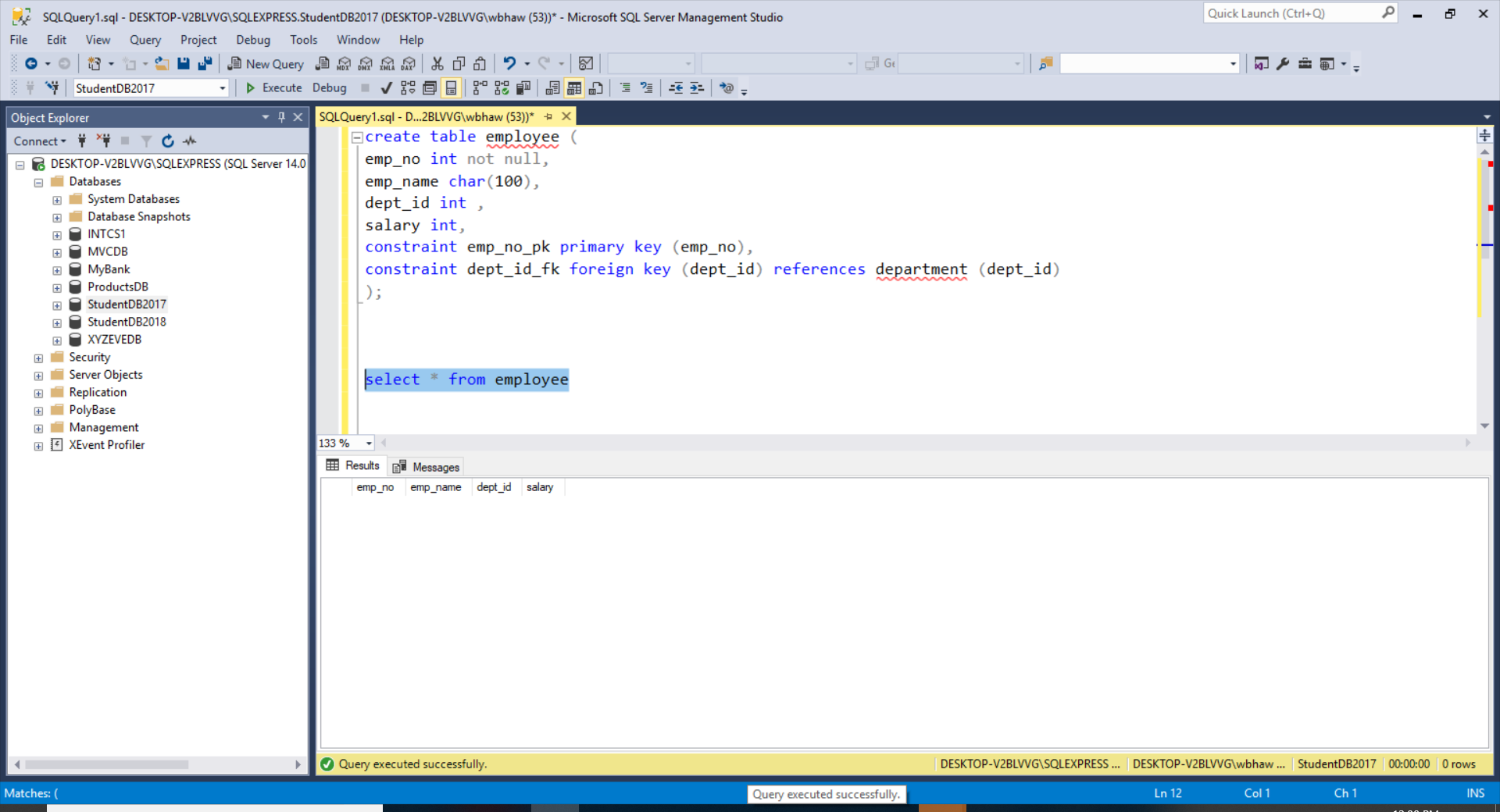
dept\_name char(50) not null,

constraint dept\_id\_pk primary key (dept\_id)

)

**Output:-**





**Script :-**

create table employee (

emp\_no int not null,

emp\_name char(100),

dept\_id int ,

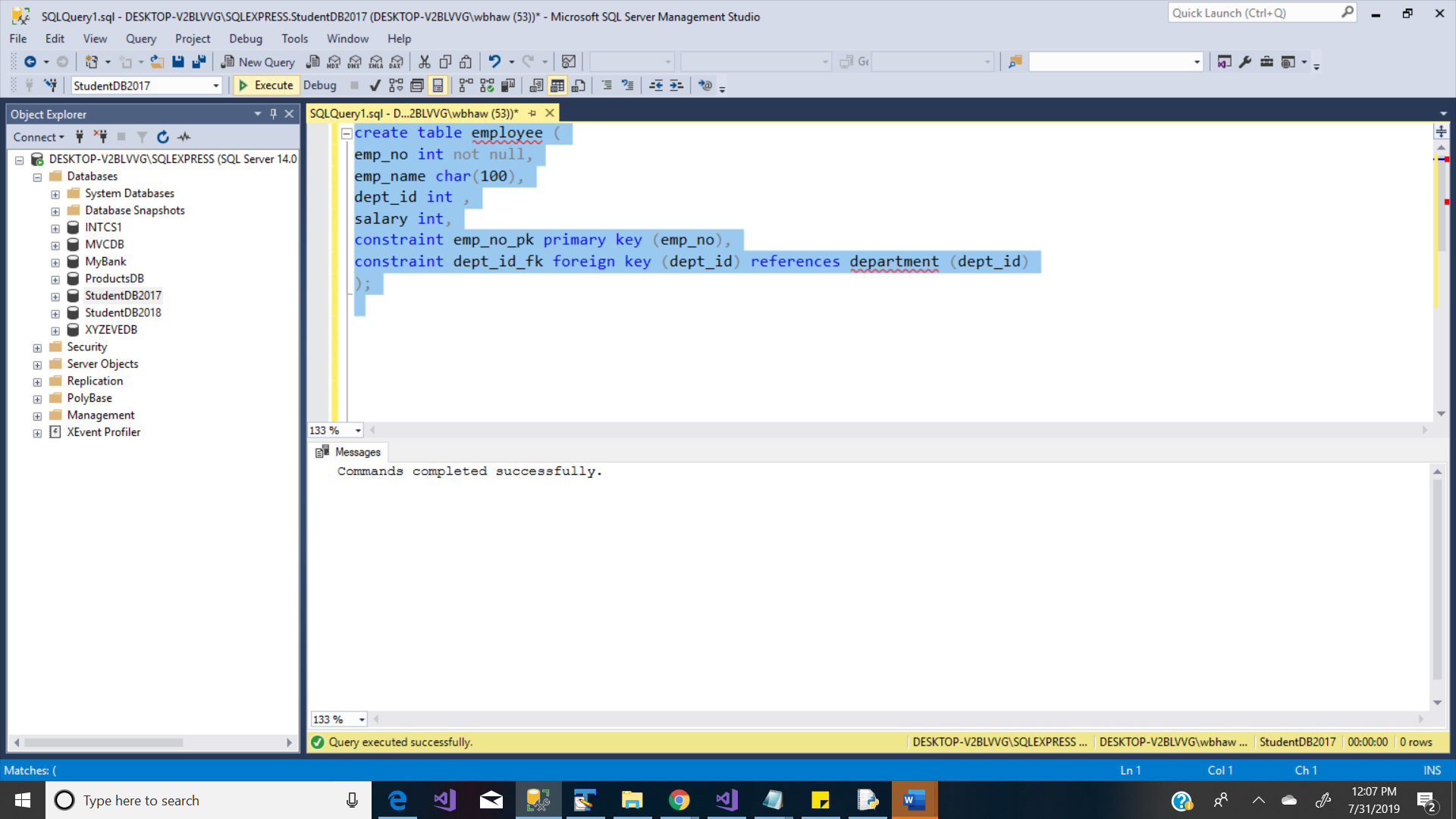
salary int,

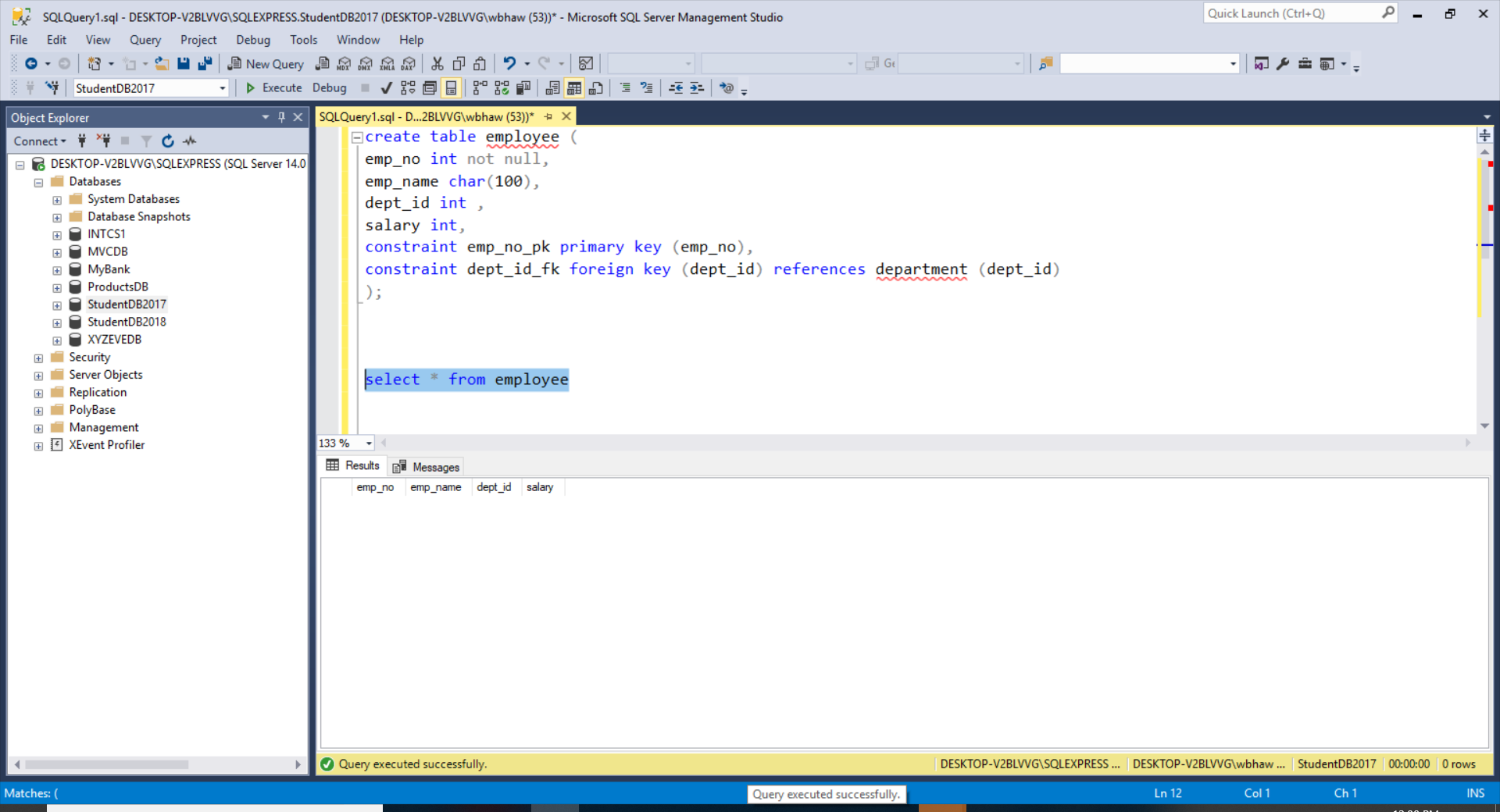
constraint emp\_no\_pk primary key (emp\_no),

constraint dept\_id\_fk foreign key (dept\_id) references department (dept\_id)

);

**Output:-**





**Data Insertion :-**

insert into department

values(10, 'ACCOUNTING');

insert into department

values(20, 'RESEARCH');

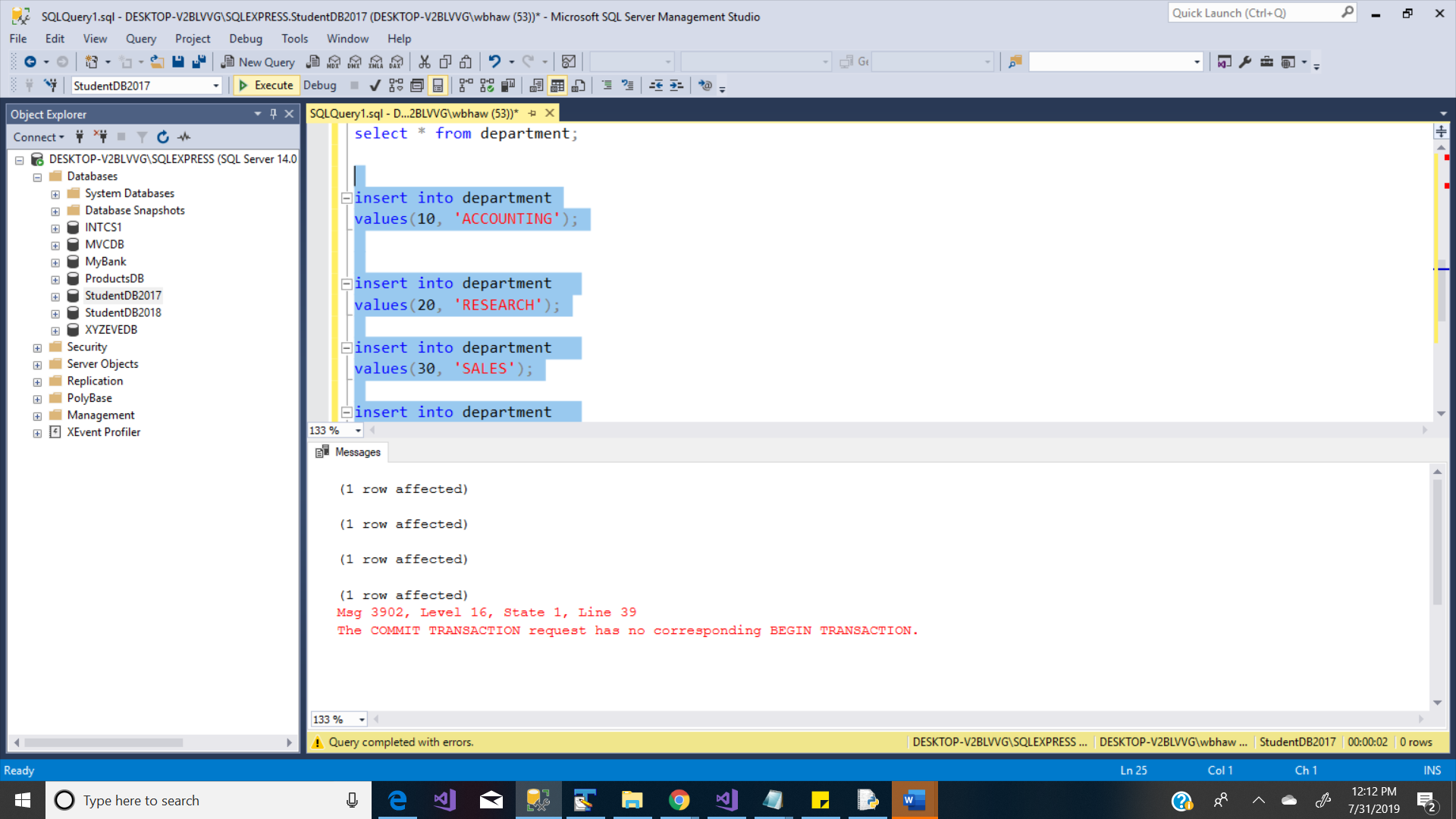
insert into department

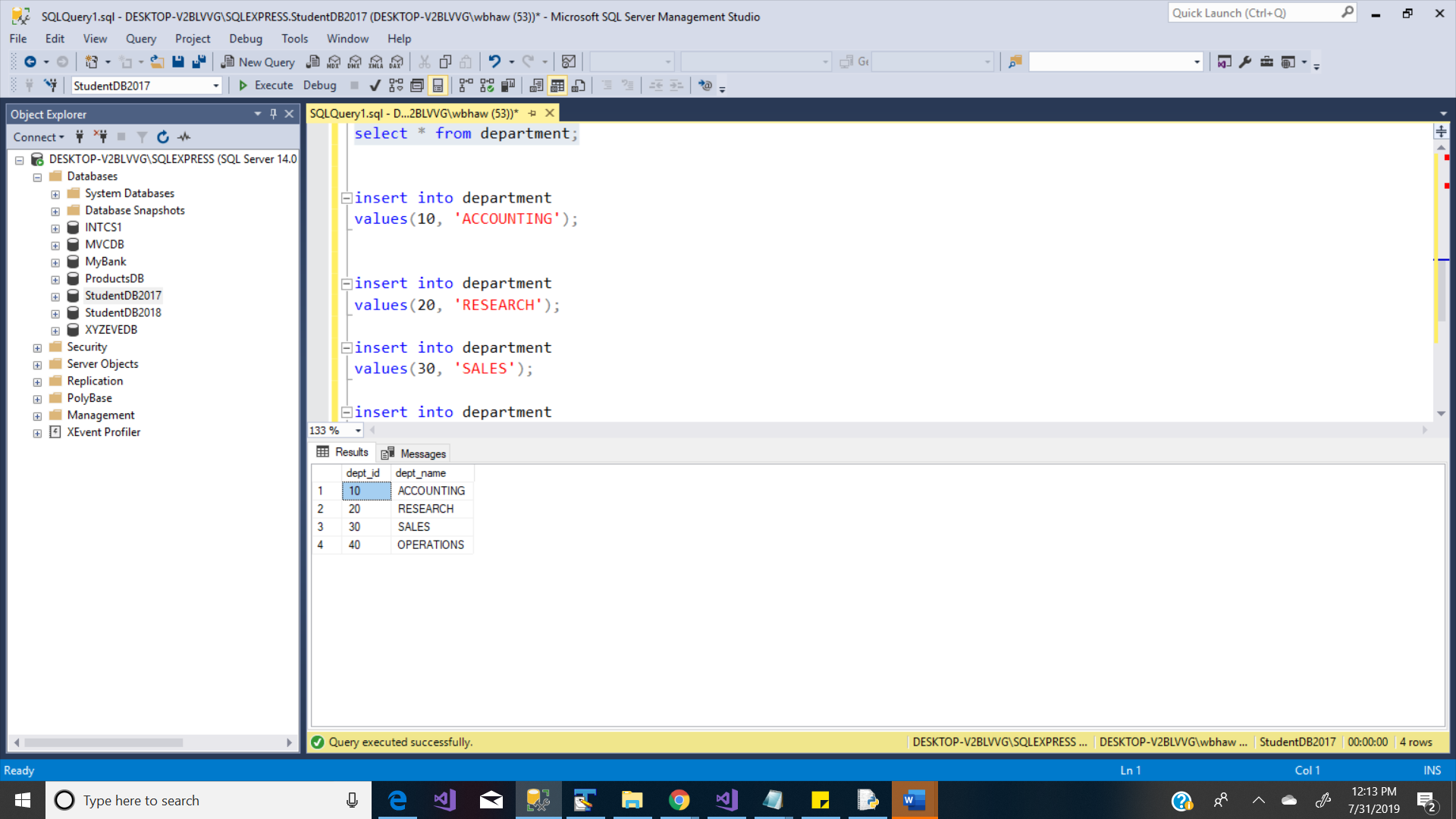
values(30, 'SALES');

insert into department

values(40, 'OPERATIONS');

commit;





insert into employee

values(

100, 'Alex White', 10, 60000

);

insert into employee

values(

101, 'James Hunter', 10, 65000

);

insert into employee

values(

102, 'Olivia Ditcher', 10, 70000

);

insert into employee

values(

103, 'Lisa Ray', 10, 40000

);

insert into employee

values(

104, 'Naveen Mishra', 20, 50000

);

insert into employee

valuses(

105, 'Kelly Evans', 30, 90000

);

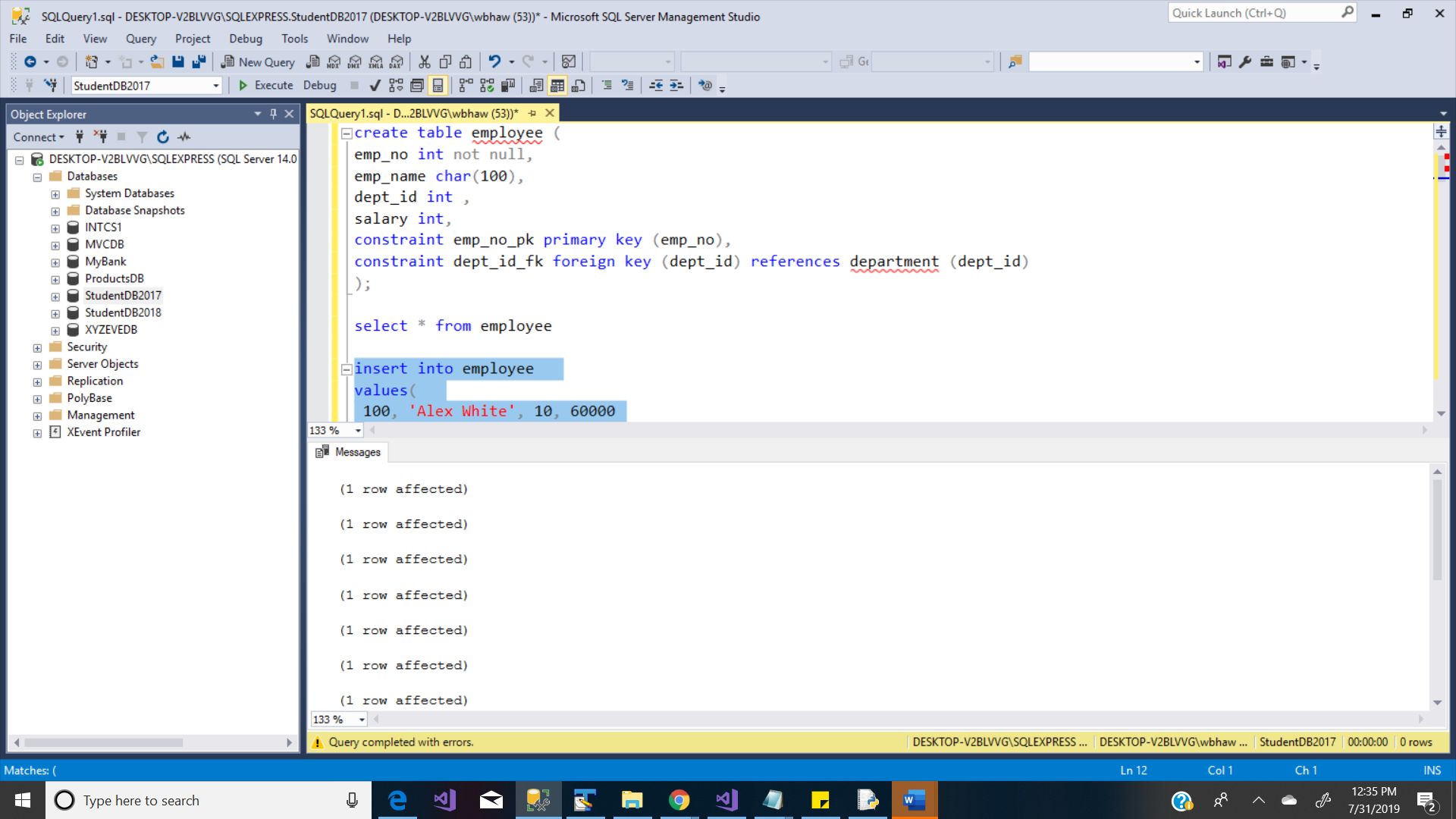
insert into employee

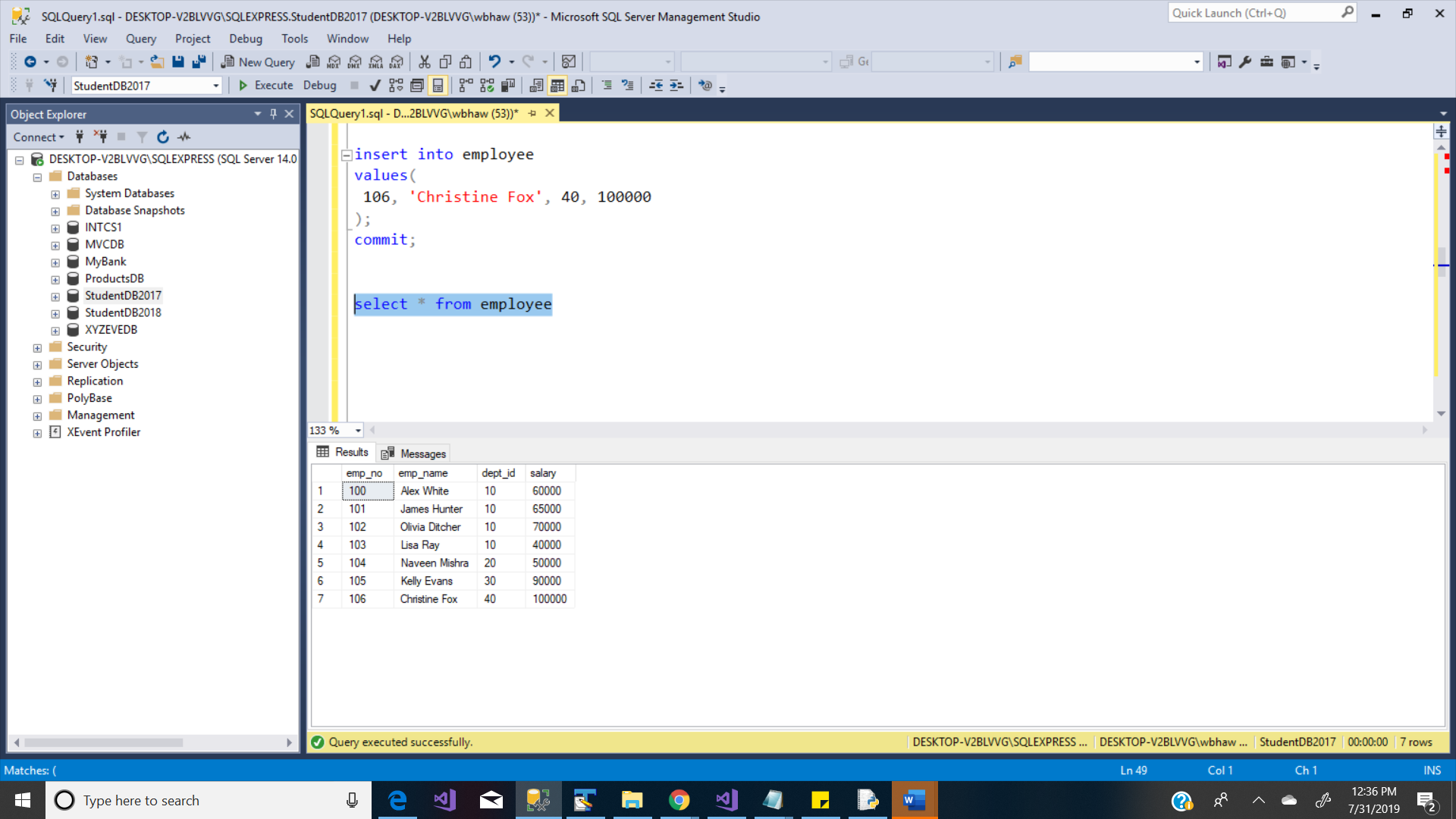
values(

106, 'Christine Fox', 40, 100000

);

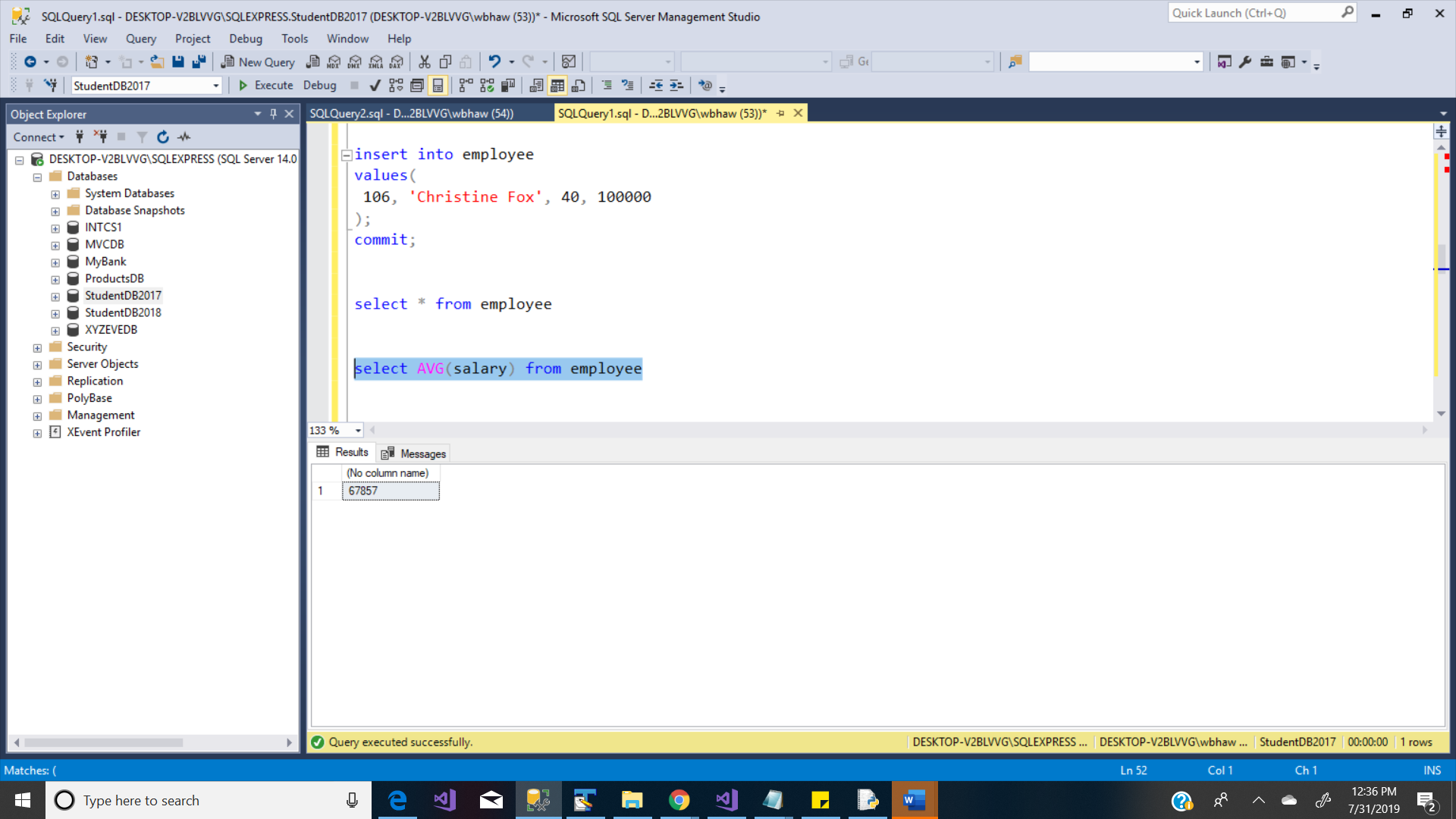
commit;





To Calculate avg salary of all employees

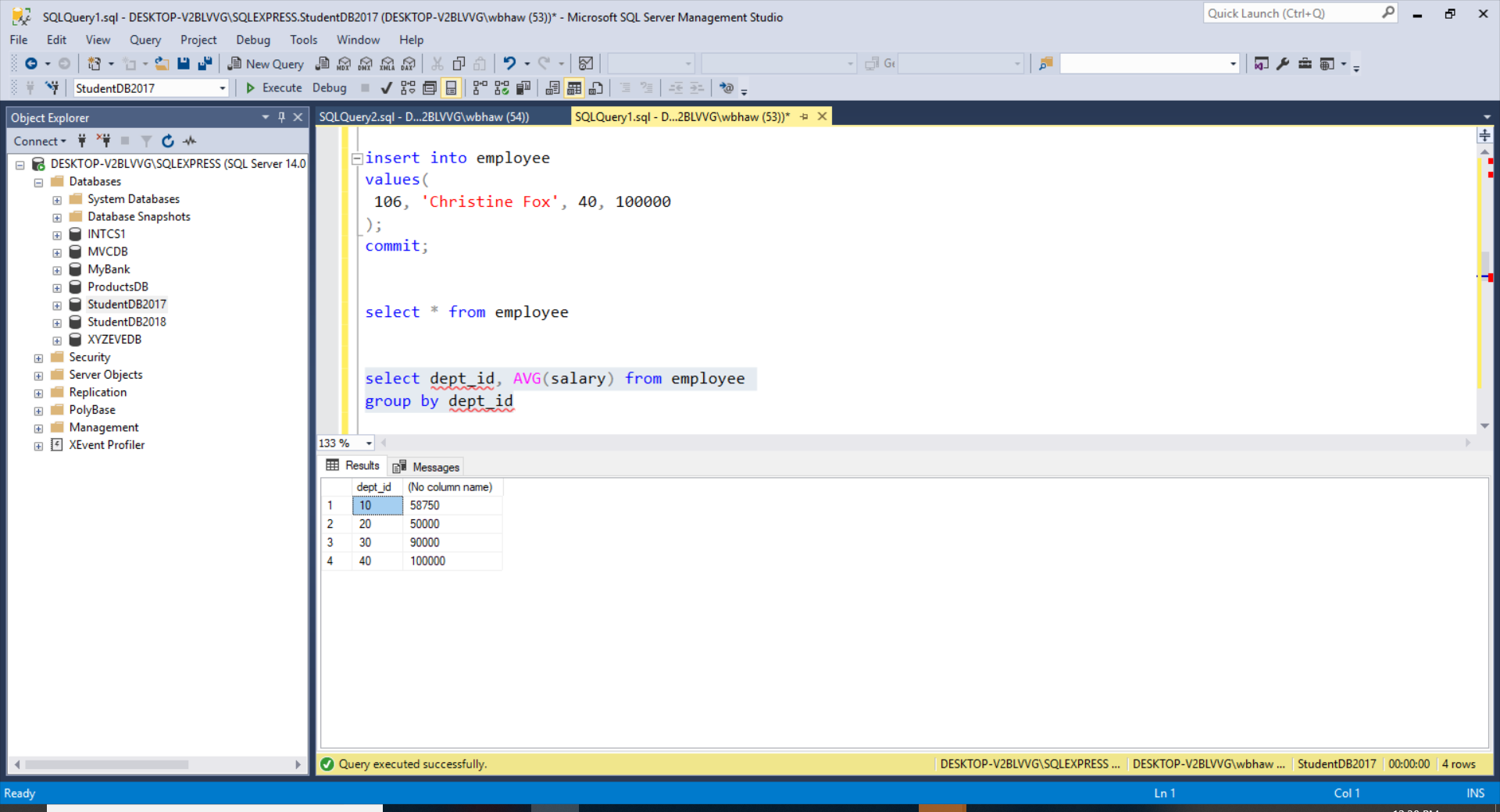
select AVG(salary) from employee



To calculate avg salary department wise

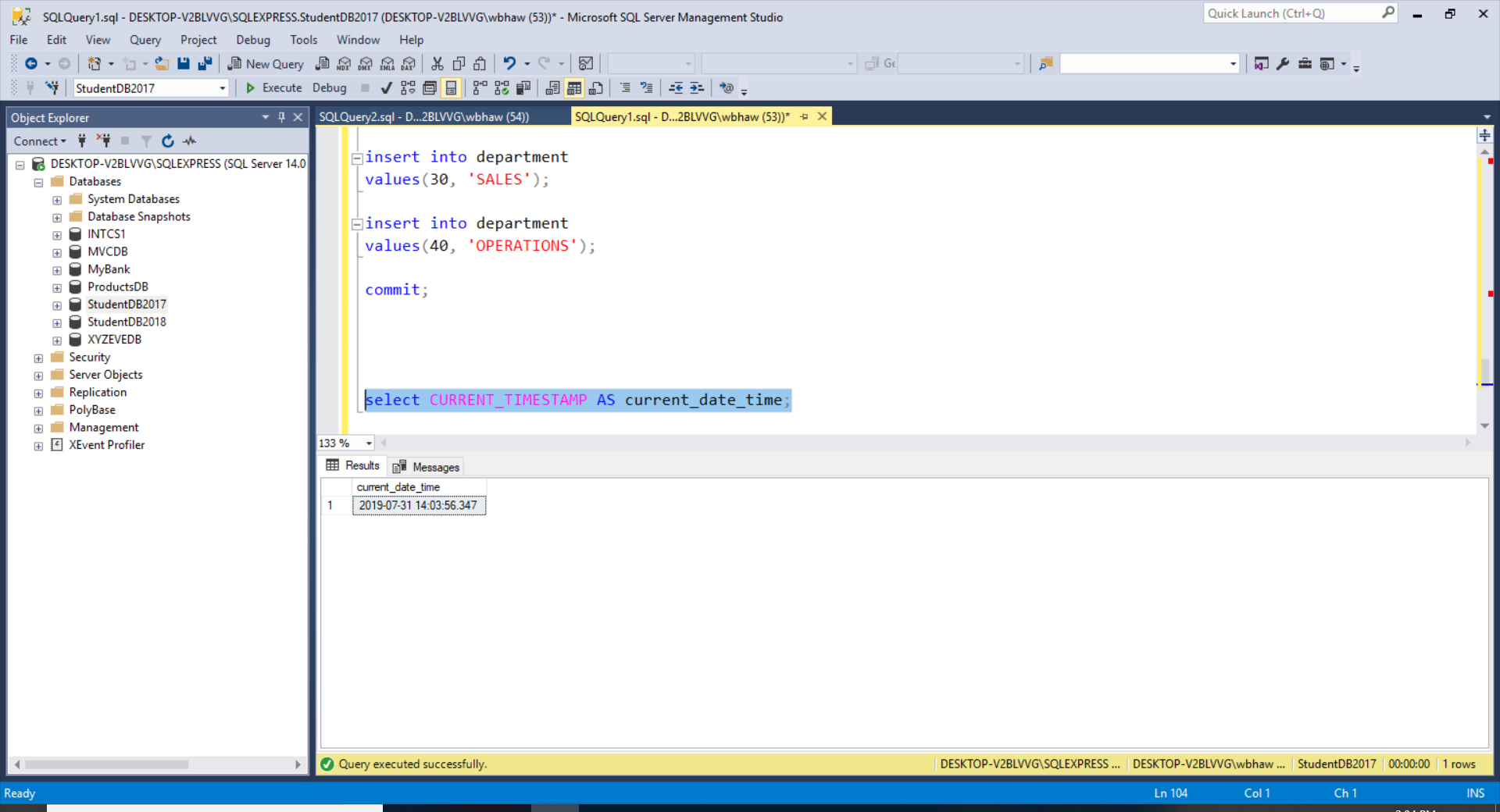
select dept\_id, AVG(salary) from employee

group by dept\_id;



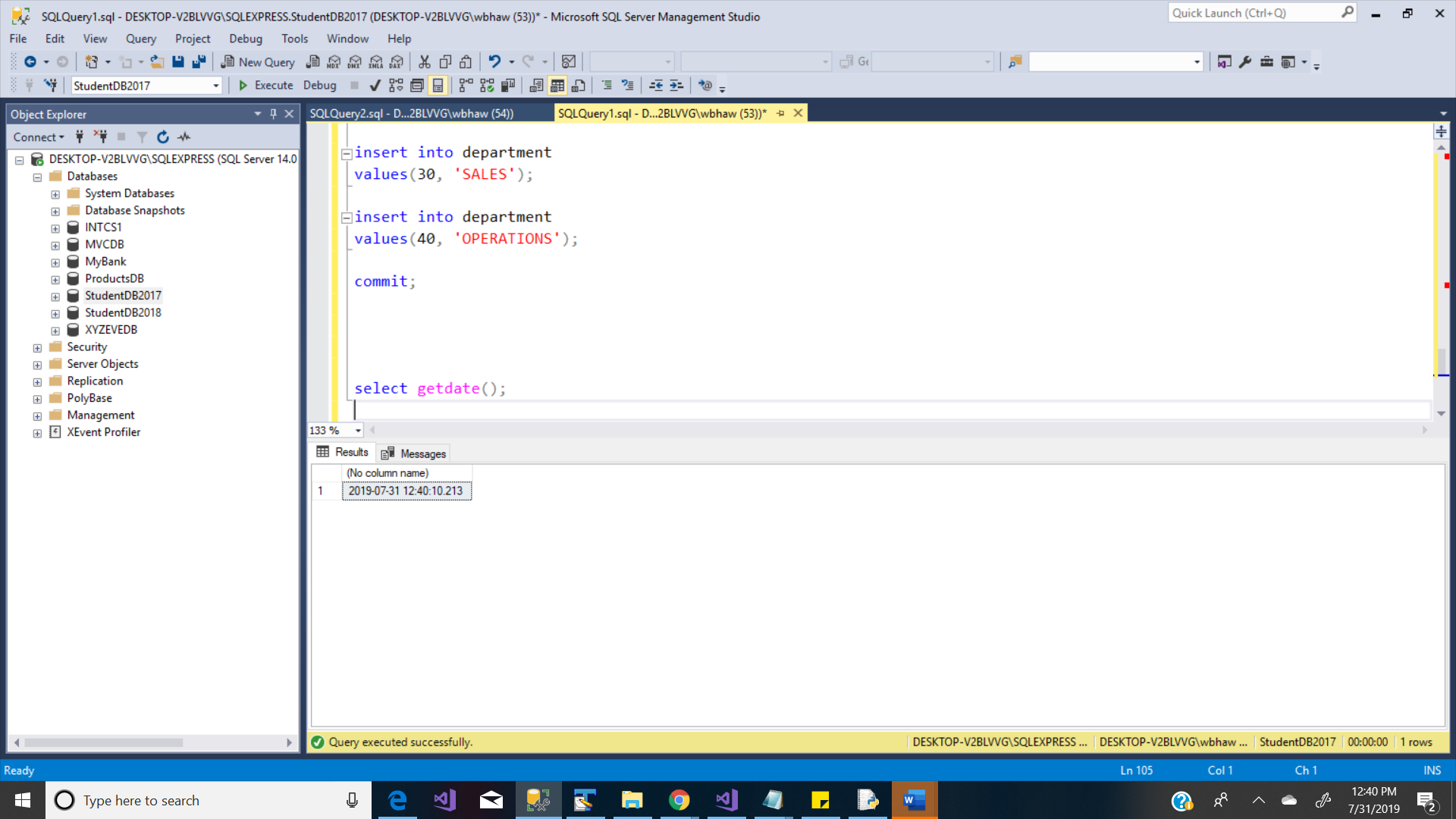
1. sCurrent date and time

select CURRENT\_TIMESTAMP AS current\_date\_time;



To check system date :

select getdate();



**File Attachment: -**

****

**Note :- Please refer PYTHON&JAVA reference code folder for more details .s**