/\*\*

\* -----Name: Mario Muscarella

\* -----ID: 2478702

\* -----Description: Lab 2: Simulating a Simple Query Processor that evaluates

\* a SQL Query in Select-From-Where

\*\*/

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Processor

{

class Program

{

static void Main(string[] args)

{

// Adding table's data into table.

string path =

Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;

string filepath\_EMPLOYEE = path + @"\CSV\Employee.csv";

DataTable EMPLOYEE = CSVtoDataTable(filepath\_EMPLOYEE);

string filepath\_DEPARTMENT = path + @"\CSV\Department.csv";

DataTable DEPARTMENT = CSVtoDataTable(filepath\_DEPARTMENT);

// variables for Selection Step

string selection\_tablename;

string selection\_column\_name;

string selection\_final\_table;

DataTable step1table = new DataTable();

//string selection\_oprator;

string selection\_number;

// variable for Join Step

string join\_table2, join\_col1, join\_col2, join\_final\_table;

DataTable step2table = new DataTable();

// variable for Projection Step

string[] projection = new string[10];

int j = 0;

//Reading Input file.

string line;

string[] lines = new string[3];

StreamReader sr = new StreamReader(path + @"\Input.txt");

while ((line = sr.ReadLine()) != null)

{

string[] words = line.Split(' ');

for (int i = 0; i < words.Length; i++)

{

//Reading file based on input and specific keyword

//Step1: Selection In which at end we have a result

if (words[i] == "Selection")

{

i++;

selection\_tablename = words[i];

i++;

selection\_column\_name = words[i];

i++;

i++;

selection\_number = words[i];

i++;

selection\_final\_table = words[i];

DataTable table = new

DataTable(selection\_final\_table.ToString());

string emp = "EMPLOYEE";

string dep = "DEPARTMENT";

if (selection\_tablename == emp)

table = selection(EMPLOYEE, selection\_column\_name,

selection\_number);

else if (selection\_tablename == dep)

table = selection(DEPARTMENT, selection\_column\_name, selection\_number);

step1table = table.Copy();

Console.WriteLine("//After execution of step 1");

ShowTable(step1table); //

break;

}

if (words[i] == "Join")

{

i++;

i++;

join\_table2 = words[i];

i++;

join\_col1 = words[i];

i++;

i++;

join\_col2 = words[i];

i++;

join\_final\_table = words[i];

step2table = new DataTable(join\_final\_table.ToString());

string dep = "DEPARTMENT";

if (join\_table2 == dep)

step2table = MergeTables(step1table, DEPARTMENT, "DNO");

Console.WriteLine("//After execution of step 2");

ShowTable(step2table);

break;

}

if (words[i] == "Projection")

{

i++; i++;

projection[j] = words[i];

j++; i++;

projection[j] = words[i];

j++; i++;

projection[j] = words[i];

j++; i++;

projection[j] = words[i];

j++; i++;

projection[j] = words[i];

j++; i++;

DataTable table = new DataTable(words[i]);

table = projection\_final(step2table, projection);

Console.WriteLine("//After execution of step 3");

ShowTable(table);

}

}

}

}

/\*\* MethodName : projection\_final

Description : Return table depending on projection

Input : Table name

Output : Table with specific column names

\*/

private static DataTable projection\_final(DataTable tb, string[] selected\_column)

{

DataTable temp\_table = new DataTable();

temp\_table = tb.Copy();

int flag = 1;

foreach (DataColumn c in tb.Columns)

{

for (int i = 0; i < selected\_column.Length; i++)

{

if (c.ColumnName == selected\_column[i])

{

flag = 1;

break;

}

else

flag = 0;

}

if (flag == 0)

{

temp\_table.Columns.Remove(c.ColumnName);

}

}

return temp\_table;

}

/\*\* MethodName : MergeTables

Description : Return table depending two tabe join

Input : Table names and column name

Output : Whole table with all data

\*/

private static DataTable MergeTables(DataTable dtFirst, DataTable dtSecond, string CommonColumn)

{

DataTable dtResults = dtFirst.Clone();

int count = 0;

for (int i = 0; i < dtSecond.Columns.Count; i++)

{

if (!dtFirst.Columns.Contains(dtSecond.Columns[i].ColumnName))

{

dtResults.Columns.Add(dtSecond.Columns[i].ColumnName, dtSecond.Columns[i].DataType);

count++;

}

}

DataColumn[] columns = new DataColumn[count];

int j = 0;

for (int i = 0; i < dtSecond.Columns.Count; i++)

{

if (!dtFirst.Columns.Contains(dtSecond.Columns[i].ColumnName))

{

columns[j++] = new DataColumn(dtSecond.Columns[i].ColumnName, dtSecond.Columns[i].DataType);

}

}

dtResults.BeginLoadData();

foreach (DataRow dr in dtFirst.Rows)

{

dtResults.Rows.Add(dr.ItemArray);

}

foreach (DataRow dr in dtSecond.Rows)

{

foreach (DataRow dr1 in dtResults.Rows)

{

if (dr1[CommonColumn].ToString().Equals(dr[CommonColumn].ToString()))

{

foreach (DataColumn c in columns)

{

dr1[c.ColumnName] = dr[c.ColumnName];

}

}

}

}

dtResults.EndLoadData();

return dtResults;

}

/\*\* MethodName : selection

Description : Return table depending on table name, column name and number

Input : Table names, column name and number

Output : Table which are on specific condition

\*/

private static DataTable selection(DataTable tablename, string column\_name, string number)

{

Console.WriteLine("//Displaying Data from table at beginning");

int x\_row = 0;

string[] temp\_row = new string[10];

DataTable temp\_table = new DataTable();

//DataColumn column = new DataColumn;

//DataRow row;

foreach (DataColumn c in tablename.Columns)

{

temp\_table.Columns.Add(c.ToString());

Console.Write("{0,-14}", c.ColumnName);

}

Console.WriteLine();

foreach (DataRow r in tablename.Rows)

{

int flag = 0;

x\_row = 0;

foreach (DataColumn c in tablename.Columns)

{

temp\_row[x\_row] = r[c].ToString();

Console.Write("{0,-14}", r[c]);

x\_row++;

if (c.ColumnName == column\_name)

{

if (r[c].ToString() == number)

{

flag++;

}

}

}

if (flag >= 1)

{

temp\_table.ImportRow(r);

}

Console.WriteLine();

}

return temp\_table;

}

/\*\* MethodName : CSVtoDataTable

Description : Return table and converts csv file to C#

Input : CSV file path

Output : Returns C# Datateble

\*/

private static DataTable CSVtoDataTable(string filepath)

{

StreamReader sr = new StreamReader(filepath);

string[] headers = sr.ReadLine().Split(',');

DataTable dt = new DataTable();

foreach (string header in headers)

{

dt.Columns.Add(header);

}

while (!sr.EndOfStream)

{

string[] rows = sr.ReadLine().Split(',');

DataRow dr = dt.NewRow();

for (int i = 0; i < headers.Length; i++)

{

dr[i] = rows[i];

}

dt.Rows.Add(dr);

}

return dt;

}

/\*\* MethodName : ShowTable

Description : Print table in output

Input : Data Table

Output : Shows table

\*/

private static void ShowTable(DataTable table)

{

//Console.WriteLine();

Console.WriteLine("-----------------------------------------------------------------------------------------------------");

foreach (DataColumn col in table.Columns)

{

Console.Write("{0,-14}", col.ColumnName);

}

Console.WriteLine();

foreach (DataRow row in table.Rows)

{

foreach (DataColumn col in table.Columns)

{

Console.Write("{0,-14}", row[col]);

}

Console.WriteLine();

}

Console.WriteLine();

}

private static void ToPrintConsole(DataTable dataTable)

{

// Print top line

Console.WriteLine(new string('-', 75));

// Print col headers

var colHeaders = dataTable.Columns.Cast<DataColumn>().Select(arg => arg.ColumnName);

foreach (String s in colHeaders)

{

Console.Write("| {0,-20}", s);

}

Console.WriteLine();

// Print line below col headers

Console.WriteLine(new string('-', 75));

// Print rows

foreach (DataRow row in dataTable.Rows)

{

foreach (Object o in row.ItemArray)

{

Console.Write("| {0,-20}", o.ToString());

}

Console.WriteLine();

}

// Print bottom line

Console.WriteLine(new string('-', 75));

}

}

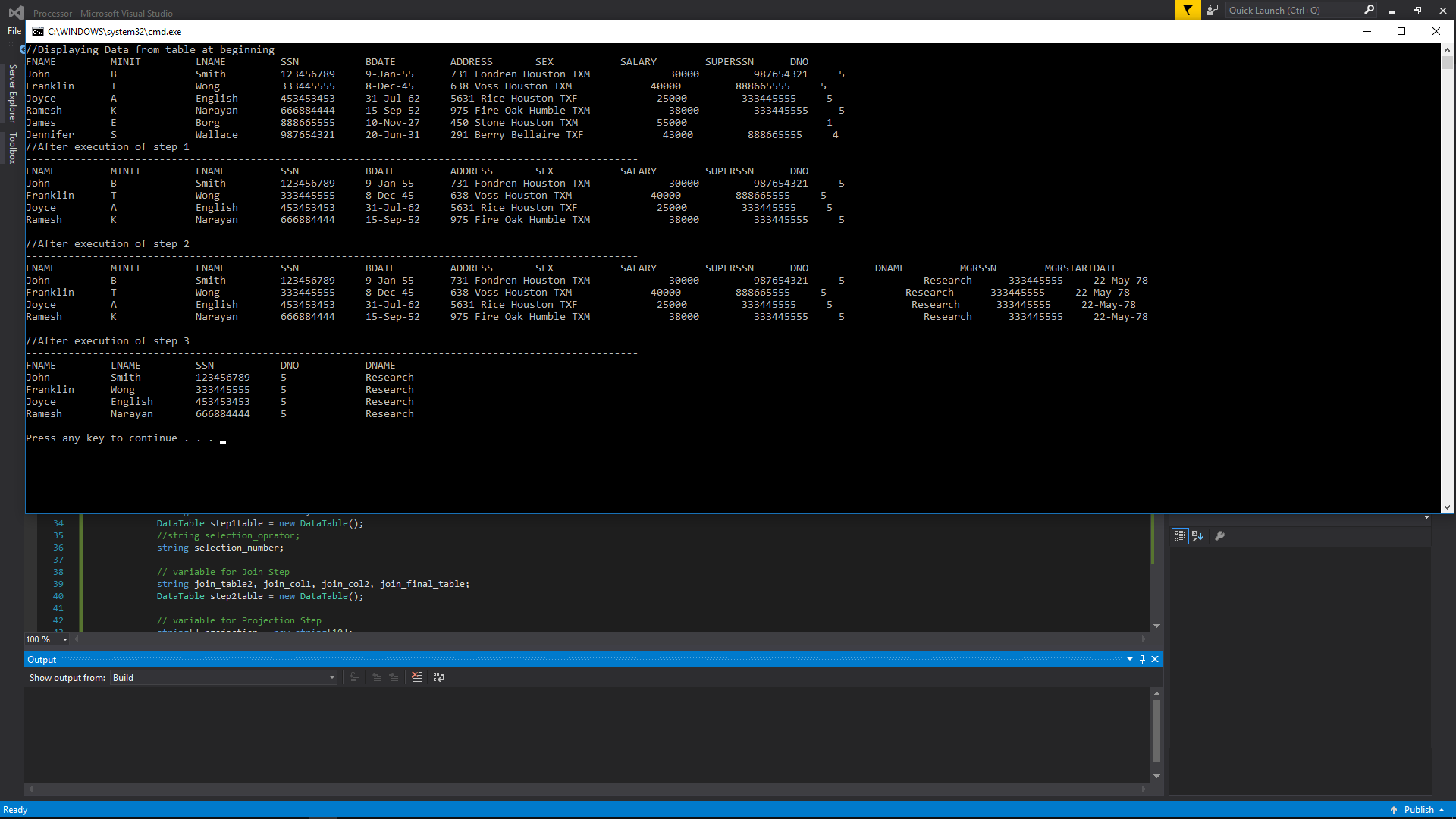
}

Output:

Selection EMPLOYEE DNO = 5 EMPS\_DNO5

Join EMPS\_DNO5 DEPARTMENT DNO = DNUMBER EMP\_DEPT\_DNO5

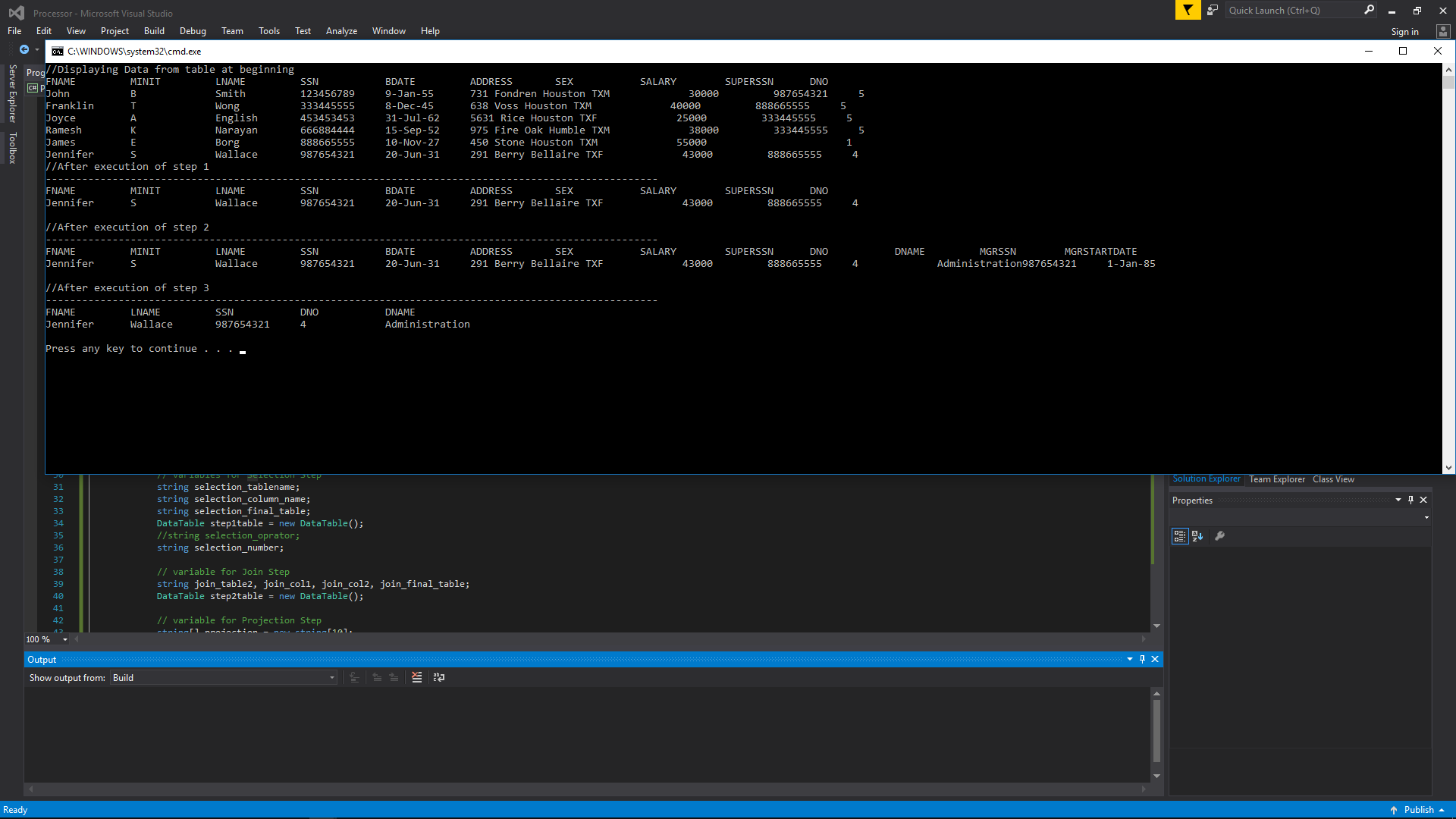
Projection EMP\_DEPT\_DNO5 FNAME LNAME SSN DNO DNAME EMP\_DEPT\_MGR\_DEPENDENT



Selection EMPLOYEE DNO = 4 EMPS\_DNO5

Join EMPS\_DNO5 DEPARTMENT DNO = DNUMBER EMP\_DEPT\_DNO5

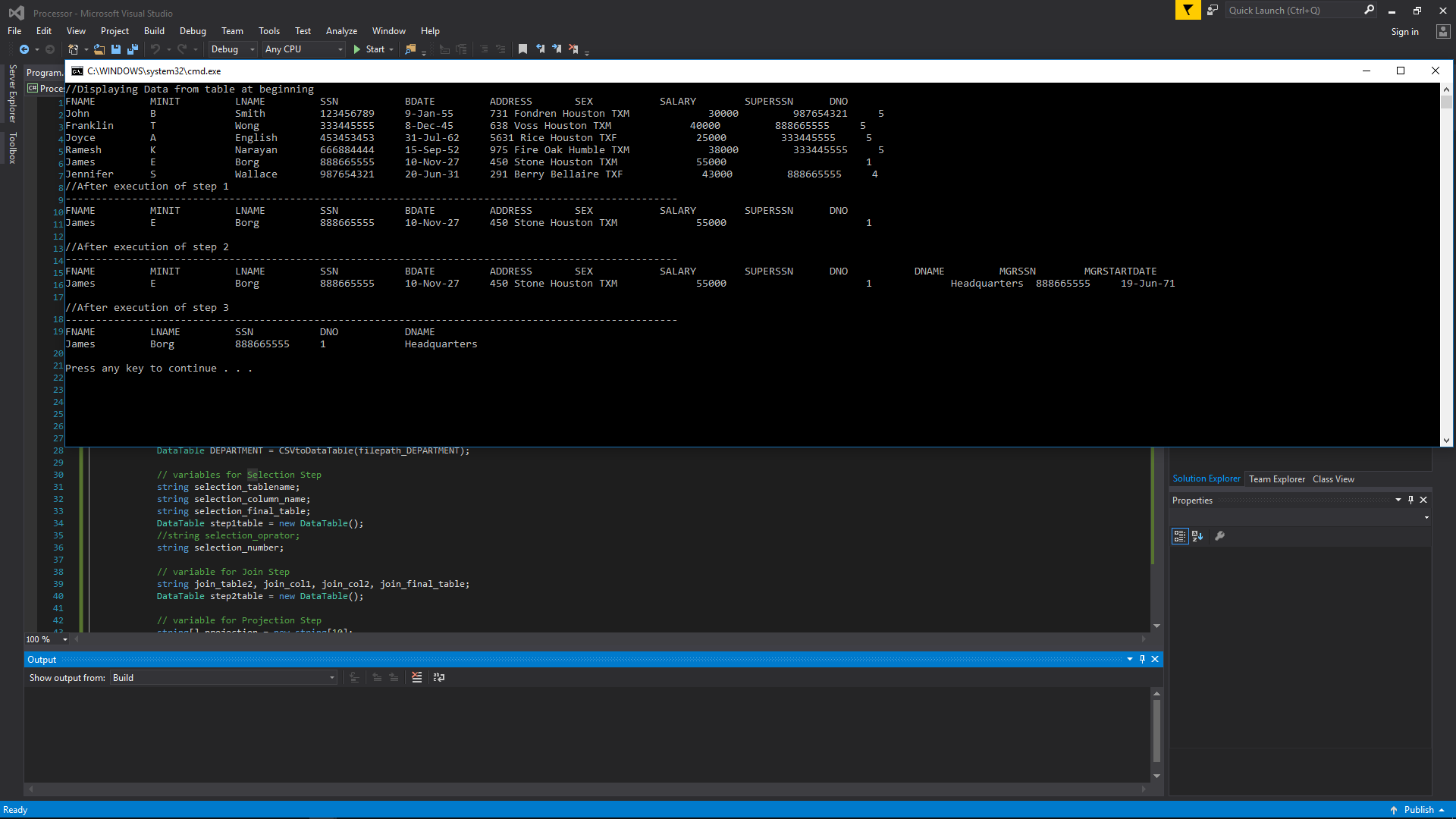
Projection EMP\_DEPT\_DNO5 FNAME LNAME SSN DNO DNAME EMP\_DEPT\_MGR\_DEPENDENT



Selection EMPLOYEE DNO = 1 EMPS\_DNO5

Join EMPS\_DNO5 DEPARTMENT DNO = DNUMBER EMP\_DEPT\_DNO5

Projection EMP\_DEPT\_DNO5 FNAME LNAME SSN DNO DNAME EMP\_DEPT\_MGR\_DEPENDENT



I was not able to get the spacing to line up later in the query. I chose C# because of the Data Science curriculum I am working my way through, and figured I could use some C# practice. All files in zip folder.

\*\*\*\* To run, be sure to select ‘Start Without Debugging’ from the Debug menu in Visual Studio \*\*\*\*