

Job-Shop Accounting System

Tony Nguyen (nguy0132)

Student ID: 113418537

Email:

tony.d.nguyen@ou.edu

Database Management
Systems, CS-4513-001

Fall 2019

Dr. Le Gruenwald

Table of Contents

Task 1.

1.1 - ER Diagram.....	pg. 5
1.2 - Relational Database Schema.....	pg. 5-6

Task 2.

2.1 - Data Directory.....	pg. 7-10
---------------------------	----------

Task 3.

3.1 - Discussion of storage structures for tables.....	pg. 11-14
3.2 - Discussion of storage structures for tables (Azure SQL Database).....	pg. 14

Task 4.

4.1 - SQL statements and screenshots showing the creation of tables in Azure SQL Database.....	pg. 16-25
--	-----------

Task 5.

5.1 - The Java source program and screenshots of successful compilation....	pg. 26-59
---	-----------

Task 6.

6.1 - Screenshot showing the testing of query 1.....	pg. 61
6.2 - Screenshot showing the testing of query 2.....	pg. 61
6.3 - Screenshot showing the testing of query 3.....	pg. 62
6.4 - Screenshot showing the testing of query 4.....	pg. 62-63

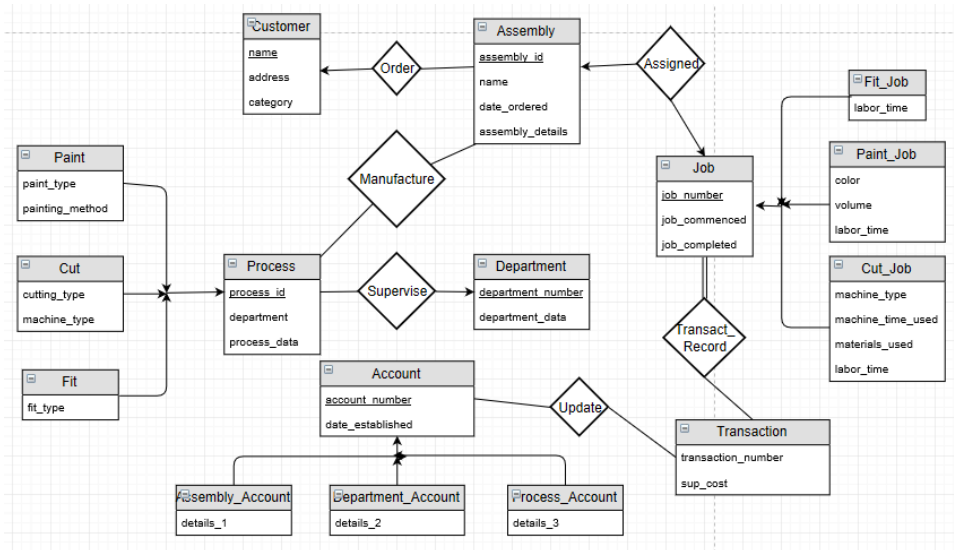
6.5 - Screenshot showing the testing of query 5.....	pg. 64
6.6 - Screenshot showing the testing of query 6.....	pg. 64
6.7 - Screenshot showing the testing of query 7.....	pg. 65
6.8 - Screenshot showing the testing of query 8.....	pg. 65-66
6.9 - Screenshot showing the testing of query 9.....	pg. 66
6.10 - Screenshot showing the testing of query 10.....	pg. 66
6.11 - Screenshot showing the testing of query 11.....	pg. 67
6.12 - Screenshot showing the testing of query 12.....	pg. 67
6.13 - Screenshot showing the testing of query 13.....	pg. 67
6.14 - Screenshot showing the testing of query 14.....	pg. 68
6.15 - Screenshot showing the testing of query 15.....	pg. 68
6.16 - Screenshot showing the testing of query 16.....	pg. 69
6.17 - Screenshot showing the testing of query 17.....	pg. 69
6.18 - Screenshot showing the testing of query 18.....	pg. 70
6.19 - Screenshot showing the testing of query 19.....	pg. 70

Task 7.

7.1 - Web database application source program and screenshots showing its successful compilation.....	pg. 71-79
--	-----------

Task 1

1.1 - ER Diagram of Job-Shop



1.2 - Relational Schema of Job-Shop

- ✓ Customer (name, address, category)
- ✓ Assembly (assembly_id, name, date_ordered, assembly_details)
- ✓ Process (process_id, department, process_data)
 - Paint (process_id, paint_type, painting_method)
 - Fit (process_id, fit_type)
 - Cut (process_id, cutting_type, machine_type)
- ✓ Department (department_number, department_data)
- ✓ Job (job_number, assembly_id, process_id, job_commenced, job_completed)
 - Cut_Job (job_number, machine_type, machine_time_used, materials_used, labor_time)
 - Paint_Job (job_number, color, volume, labor_time)
 - Fit_Job (job_number, labor_time)
- ✓ Account (account_number, date_established)
 - Assembly_Account (account_number, transaction_number, assembly_id, details_1)
 - Process_Account (account_number, process_id, details_3)

- Department_Account (account_number, transaction_number, department_number, details_2)
- ✓ Updates (transaction_number, account_number, sup_cost)

Task 2

2.1 - Data Directory

Name	Type	Size of Attributes (Bytes)	Constraints
Customer	name - (varchar) address - (varchar) category - (int)	name - (64) address - (255) category	Name is primary. That means no duplicates and name is unique.

Name	Type	Size of Attributes (Bytes)	Constraints
Assembly	assembly_id - (int) customer_name - (varchar) date_ordered - (varchar) assembly_details - (varchar)	assembly_id Customer_name - (64) date_ordered - (15) assembly_details - (1024)	assembly_id is unique as it is primary key.

Name	Type	Size of Attributes (Bytes)	Constraints
Process	process_id - (int) department - (int) process_data - (varchar)	process_id Department process_data - (64)	process_id is unique as it is primary key.

Name	Type	Size of Attributes (Bytes)	Constraints
Fit	fit_type - (varchar)	fit_type - (15)	process_id

Name	Type	Size of Attributes (Bytes)	Constraints
------	------	----------------------------	-------------

Paint	paint_type - (varchar) painting_method - (varchar)	paint_type - (20) painting_method - (20)	process_id
-------	---	--	------------

Name	Type	Size of Attributes (Bytes)	Constraints
Cut	cutting_type - (varchar) machine_type - (varchar)	cutting_type - (20) machine_type - (20)	process_id

Name	Type	Size of Attributes (Bytes)	Constraints
Department	department_number - (int) department_data - (varchar)	department_number department_data - (1024)	department_number is unique as it is primary key.

Name	Type	Size of Attributes (Bytes)	Constraints
Job	job_number - (int) job_commenced - (varchar) job_completed - (varchar)	job_number job_commenced - (15) job_completed - (15)	job_number is unique as it is primary key.

Name	Type	Size of Attributes (Bytes)	Constraints
Updates	transaction_number - (int) sup_cost - (int)	transaction_number sup_cost	transaction_number is unique as it is primary key.

Name	Type	Size of Attributes (Bytes)	Constraints
Cut_job	machine_type - (varchar) machine_time_used - (varchar) materials_used - (varchar) labor_time - (int)	machine_type - (20) machine_time_used - (10) materials_used - (1024) labor_time	job_number

Name	Type	Size of Attributes (Bytes)	Constraints
Paint_job	color - (varchar) volume - (int) labor_time - (int)	color - (10) volume labor_time	job_number

Name	Type	Size of Attributes (Bytes)	Constraints
Fit_job	labor_time - (Int)	labor_time	job_number

Name	Type	Size of Attributes (Bytes)	Constraints
Account	account_number - (int) date_established - (varchar)	account_number date_established - (15)	account_number

Name	Type	Size of Attributes (Bytes)	Constraints
Transaction	transaction_number - (int) account_number - (int) sup_cost - (int)	transaction_number account_number sup_cost	transaction_number

Name	Type	Size of Attributes (Bytes)	Constraints
Assembly_Account	account_number - (int) details_1 - (int)	account_number details_1	account_number

Name	Type	Size of Attributes (Bytes)	Constraints
Process_Account	account_number - (int) details_2 - (int)	account_number details_2	account_number

Name	Type	Size of Attributes (Bytes)	Constraints
Department_Account	account_number - (int) details_3 - (int)	account_number details_3	account_number

Task 3

3.1 - Storage Structure for Tables

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	Query 1 and Insertion	name	30/day	Sequential File	Name ordered

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Department	Query 2 and Insertion	department_number	Infrequent	Heap File	Quick insertion and no need for anything fancy as query is infrequent

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Assembly	Query 3 and Insertion	assembly_id	40/day	Indexed-Sequential File	Ordered, has a primary key and a foreign key

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Process	Query 4 and Insertion	process_id	Infrequent	Indexed-Sequential File	Has a primary key and foreign key to access

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Account	Query 5 and Insertion	account_number	10/day	Indexed-Sequential File	Ordered, has a primary key and a foreign key

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Job	Query 6 and Insertion	job_number	50/day	Indexed-Sequential File	Ordered, has a primary key and a foreign key

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Job (Job Completion)	Query 7 and Insertion	job_number	50/day	Indexed-Sequential File	Ordered, has a primary key and a foreign key

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Updates	Query 8 and Insertion	transaction_number	50/day	Indexed-Sequential File	Can use transaction_number as primary index and sup_cost as secondary to update costs for accounts.

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
-------------	------------------------	-------------------	------------------------	-----------------------------------	-----------------------

Assembly_Account	Query 9 and Random Search	assembly_id	200/day	B+-Tree	For range and random searches
------------------	------------------------------------	-------------	---------	---------	-------------------------------------

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Job	Query 10 and Random Search	department_number and date	20/day	Indexed- Sequential File	For range and random searches

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Process	Query 11 and Random Search	assembly_id and department_number	100/day	B+-Tree	For random searches and can handle large data sizes as this query is very frequent

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Jobs	Query 12 and Random Search	assembly_id and department_number	20/day	Heap File	Fast insertion

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
------	-----------------------	------------	--------------------	-------------------------------	----------------

Process	Query 13 and Random Search	assembly_id and department_number	100/day	B+-Tree	For random searches and can handle large data sizes as this query is very frequent
---------	----------------------------	-----------------------------------	---------	---------	--

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Cut_job	Query 14 and Deletion, Range Search	job_number	1/month	Indexed Sequential File	Efficient for range search

Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Paint_job	Query 15 and Random Search	job_number	1/week	Static hashing	Efficient for random search

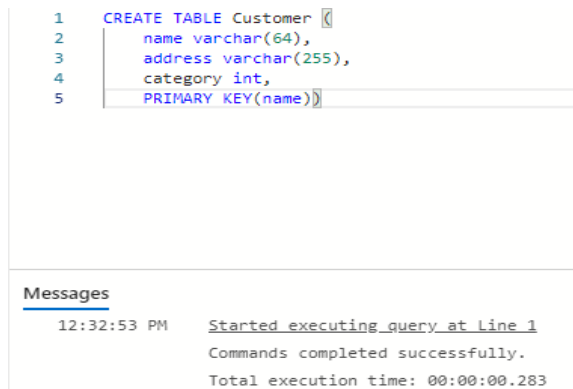
3.2 - Storage Options for each Relational Table

Data in Azure is stored in a transactional manner with advance querying. Data is stored on the cloud. Data can be stored using a table with rows and columns, user-defined functions, stored procedures, T-sql language specs, and more. Data can be hidden too such as sensitive info such as social security number, credit card number, etc. It can be hidden through dynamic data masking. After using Azure, I noticed the primary key is stored in name order as well, which is a sequential file organization. Azure includes variety types of file organization options here.

Task 4

Customer Query:

```
CREATE TABLE Customer (  
    name varchar(64),  
    address varchar(255),  
    category int,  
    PRIMARY KEY(name))
```



The screenshot displays a SQL query execution window. The query is a CREATE TABLE statement for a table named 'Customer'. The query is shown in a multi-line editor with line numbers 1 through 5. Below the query, a 'Messages' section shows the execution status: 'Started executing query at Line 1', 'Commands completed successfully.', and 'Total execution time: 00:00:00.283'.

```
1 CREATE TABLE Customer (  
2     name varchar(64),  
3     address varchar(255),  
4     category int,  
5     PRIMARY KEY(name))
```

Messages

12:32:53 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.283

Assembly Query:

```
CREATE TABLE Assembly (  
    assembly_id int,  
    customer_name varchar(64),  
    date_ordered varchar(15),  
    assembly_details varchar(1024),  
    PRIMARY KEY(assembly_id),  
    FOREIGN KEY (customer_name) REFERENCES Customer(name))
```

```

1 CREATE TABLE Assembly (
2     assembly_id int,
3     customer_name varchar(64),
4     date_ordered varchar(15),
5     assembly_details varchar(1024),
6     PRIMARY KEY(assembly_id),
7     FOREIGN KEY (customer_name) REFERENCES Customer(name))

```

Messages

12:35:28 PM Started executing query at Line 1
 Commands completed successfully.
 Total execution time: 00:00:00.278

Process Query:

```

CREATE TABLE Process (
    process_id int,
    department int,
    process_data varchar(64),
    PRIMARY KEY(process_id),
    FOREIGN KEY (department) REFERENCES Department(department_number))

```

```

1 CREATE TABLE Process (
2     process_id int,
3     department int,
4     process_data varchar(64),
5     PRIMARY KEY(process_id),
6     FOREIGN KEY (department) REFERENCES Department(department_number))

```

Messages

1:28:58 PM Started executing query at Line 1
 Commands completed successfully.
 Total execution time: 00:00:00.569

Fit (Type of Process) Query:

```

CREATE TABLE Fit (
    process_id int REFERENCES Process(process_id),
    fit_type varchar(15))

```



```

1 CREATE TABLE Fit (
2     process_id int REFERENCES Process(process_id),
3     fit_type varchar(15))

```

Messages

```

1:37:29 PM      Started executing query at Line 1
                Commands completed successfully.
                Total execution time: 00:00:00.069

```

Paint (Type of Process) Query:

```

CREATE TABLE Paint (
    process_id int REFERENCES Process(process_id),
    paint_type varchar(20),
    painting_method varchar(20))

```

```

1 CREATE TABLE Paint (
2     process_id int REFERENCES Process(process_id),
3     paint_type varchar(20),
4     painting_method varchar(20))

```

Messages

```

2:17:21 PM      Started executing query at Line 1
                Commands completed successfully.
                Total execution time: 00:00:00.170

```

Cut (Type of Process) Query:

```

CREATE TABLE Cut (
    process_id int REFERENCES Process(process_id),
    cutting_type varchar(20),
    machine_type varchar(20))

```

```

1 CREATE TABLE Cut ([
2     process_id int REFERENCES Process(process_id),
3     cutting_type varchar(20),
4     machine_type varchar(20)])

```

Messages

2:19:07 PM Started executing query at Line 1
 Commands completed successfully.
 Total execution time: 00:00:00.061

Department Query:

```

CREATE TABLE Department (
    department_number int,
    department_data varchar(1024),
    PRIMARY KEY(department_number))

```

```

1 CREATE TABLE Department ([
2     department_number int,
3     department_data varchar(1024),
4     PRIMARY KEY(department_number)])

```

Messages

12:38:51 PM Started executing query at Line 1
 Commands completed successfully.
 Total execution time: 00:00:00.338

Account Query:

```

CREATE TABLE Account (
    account_number int,
    date_established varchar(10),

```

PRIMARY KEY (account_number))

```
1 CREATE TABLE Account ([
2     account_number int,
3     date_established varchar(10),
4     PRIMARY KEY (account_number)])
```

Messages

7:02:23 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.218

Assembly_Account Query:

```
CREATE TABLE Assembly_Account (
    account_number int REFERENCES Account(account_number),
    assembly_id int,
    details_1 int,
    FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id))
```

```
1 CREATE TABLE Assembly_Account ([
2     account_number int REFERENCES Account(account_number),
3     details_1 varchar(1024)])
```

Messages

7:33:50 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.161

Department_Account Query:

```
CREATE TABLE Department_Account (
    account_number int REFERENCES Account(account_number),
```

department_number int,
details_2 int,
FOREIGN KEY (department_number) REFERENCES
Department(department_number))

```
1 CREATE TABLE Department_Account ([  
2   account_number int REFERENCES Account(account_number),  
3   details_2 varchar(1024)])
```

Messages

7:36:05 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.217

Process_Account Query:

CREATE TABLE Process_Account (
account_number int REFERENCES Account(account_number),
process_id int,
details_3 int,
FOREIGN KEY (process_id) REFERENCES Process(process_id))

```
1 CREATE TABLE Process_Account ([  
2   account_number int REFERENCES Account(account_number),  
3   details_3 varchar(1024)])
```

Messages

7:37:36 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.195

Job Query:

```

CREATE TABLE Job (

    job_number int,

    assembly_id int,

    process_id int,

    job_commenced varchar(15),

    job_completed varchar(15),

    PRIMARY KEY (job_number),

    FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id),

    FOREIGN KEY (process_id) REFERENCES Process(process_id))

```

```

1 CREATE TABLE Job ([
2     job_number int,
3     assembly_id int,
4     process_id int,
5     job_commenced varchar(15),
6     job_completed varchar(15),
7     PRIMARY KEY (job_number),
8     FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id),
9     FOREIGN KEY (process_id) REFERENCES Process(process_id)])

```

Messages

```

11:17:13 AM Started executing query at line 1
Commands completed successfully.
Total execution time: 00:00:00.384

```

Fit_Job Query:

```

CREATE TABLE Fit_Job (

    job_number int REFERENCES Job(job_number),

    labor_time int)

```

```

1 CREATE TABLE Fit_Job (
2     job_number int REFERENCES Job(job_number),
3     labor_time int)

```

Messages

```

11:53:16 AM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.073

```

Paint_Job Query:

```

CREATE TABLE Paint_Job (
    job_number int REFERENCES Job(job_number),
    color varchar(10),
    volume int,
    labor_time int)

```

```

1 CREATE TABLE Paint_Job (
2     job_number int REFERENCES Job(job_number),
3     color varchar(10),
4     volume int,
5     labor_time int)

```

Messages

```

11:55:08 AM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.074

```

Cut_Job Query:

```

CREATE TABLE Cut_Job (
    job_number int REFERENCES Job(job_number),

```

machine_type varchar(20),
machine_time_used int,
materials_used varchar(1024),
labor_time int)

```
1 CREATE TABLE Cut_Job (  
2     job_number int REFERENCES Job(job_number),  
3     machine_type varchar(20),  
4     machine_time_used int,  
5     materials_used varchar(1024),  
6     labor_time int)
```

Messages

12:34:19 PM Started executing query at line 1
Commands completed successfully.
Total execution time: 00:00:00.059

Updates Query:

```
CREATE TABLE Updates (  
  
    transaction_number int,  
  
    account_number int,  
  
    sup_cost int,  
  
    PRIMARY KEY (transaction_number),
```

FOREIGN KEY (account_number) REFERENCES Account(account_number))

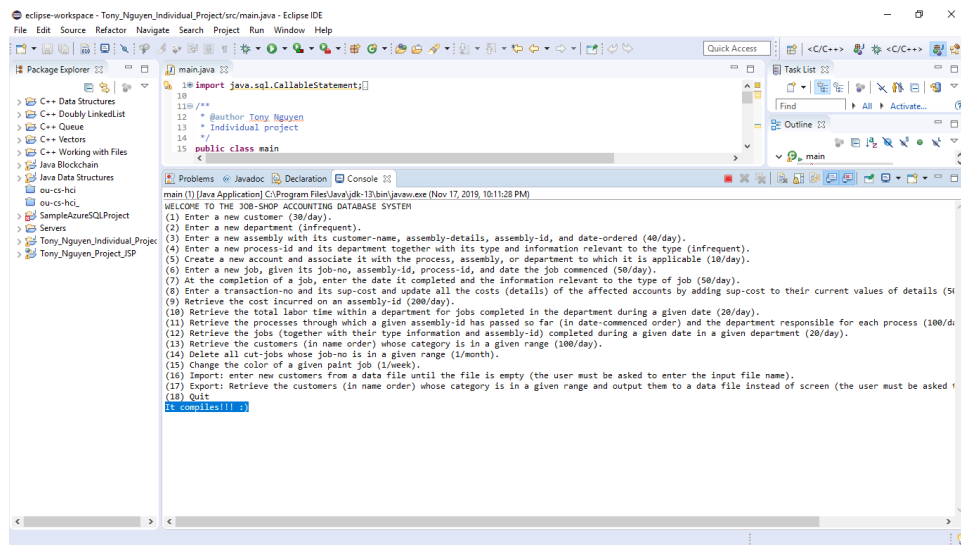
```
1 CREATE TABLE Updates (  
2     transaction_number int,  
3     account_number int,  
4     sup_cost int,  
5     PRIMARY KEY (transaction_number),  
6     FOREIGN KEY (account_number) REFERENCES Account(account_number))
```

Messages

8:48:41 AM Started executing query at line 1
Commands completed successfully.
Total execution time: 00:00:00.180

Task 5

5.1 - Source Code with Compilation Screenshot



```
import java.io.File;
```

```
import java.io.FileOutputStream;
```

```
import java.io.PrintWriter;
```

```
import java.sql.Connection;
```

```
import java.sql.Statement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.ResultSetMetaData;
```

```
import java.sql.SQLException;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```

import java.util.Scanner;

/**
 * @author Tony Nguyen
 * Individual project
 */

public class main
{
    /**
     * Main method to run the Java-SQL program.
     * @param args
     */

    public static void main(String[] args) throws SQLException
    {
        // Setup to connect with the SQL database.

        final String hostName = "nguy0132-sql-server.database.windows.net";

        final String dbName = "cs-dsa-4513-sql-db";

        final String user = "nguy0132";

        final String password = "sN8!CVD$NBvib*7J9L23";

        final String url =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;"

```

```
        + "hostNameInCertificate=*.database.windows.net;loginTimeout=30;", hostName,  
dbName, user, password);
```

```
// Display query options for user.
```

```
System.out.println("WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM");
```

```
displayOption();
```

```
// Making changes to SQL database.
```

```
try (final Connection connection = DriverManager.getConnection(url)) {
```

```
    final String schema = connection.getSchema();
```

```
// Created a scanner object for strings.
```

```
Scanner optionScanner = new Scanner(System.in);
```

```
// Created a scanner object for ints.
```

```
Scanner intScanner = new Scanner(System.in);
```

```
String option = optionScanner.nextLine();
```

```
while (!option.equals("18"))
```

```
{
```

```
    // Option 18. Exit the program.
```

```

        if (option.equals("18"))
        {
            optionScanner.close();

            intScanner.close();

            System.out.println("THANK YOU FOR USING THE JOB-SHOP ACCOUNTING DATABASE
SYSTEM");

            System.out.println("Closing system...");

            System.exit(0);
        }

        // Option Query 1. Enter a new customer.

        if (option.equals("1"))
        {
            System.out.println("Please enter customer name:");

            String customerName = optionScanner.nextLine();

            System.out.println("Please enter customer address:");

            String customerAddress = optionScanner.nextLine();

            System.out.println("Please enter a category from 1-10:");

            int category = intScanner.nextInt();

```

```

PreparedStatement newCustomer = connection.prepareStatement(

    "INSERT INTO Customer (name, address, category) "

    + "VALUES ('"+customerName+"', '"+customerAddress+"', '"+category+"')");

newCustomer.execute();

}

// Option Query 2. Enter a new department.

else if (option.equals("2"))

{

    System.out.println("Please enter a department number:");

    int departmentNumber = intScanner.nextInt();

    System.out.println("Please enter department data:");

    String departmentData = optionScanner.nextLine();

    PreparedStatement newDepartment = connection.prepareStatement(

        "INSERT INTO Department (department_number, department_data) "

        + "VALUES ('"+departmentNumber+"', '"+departmentData+"')");

    newDepartment.execute();

```

```

    }

    // Option Query 3. Enter a new assembly connected with customer name.

    else if (option.equals("3"))

    {

        System.out.println("Please enter assembly ID:");

        int assemblyID = intScanner.nextInt();

        System.out.println("Please enter an existing customer name to add the assembly to:");

        String customerName = optionScanner.nextLine();

        System.out.println("Please enter date ordered:");

        String dateOrdered = optionScanner.nextLine();

        System.out.println("Please enter assembly details:");

        String assemblyDetails = optionScanner.nextLine();

        PreparedStatement newAssembly = connection.prepareStatement(

            "INSERT INTO Assembly (assembly_id, customer_name, date_ordered,

assembly_details) "

            + "VALUES ('"+assemblyID+"', '"+customerName+"', '"+dateOrdered+"',

 '"+assemblyDetails+"')");

```

```

        newAssembly.execute();
    }

    // Option Query 4. Enter a new process.

    else if (option.equals("4"))
    {

        System.out.println("Please enter a new process ID:");

        int processID = intScanner.nextInt();


        System.out.println("Please enter an existing department number to add process to:");

        int departmentNumber = intScanner.nextInt();


        System.out.println("Please enter process data:");

        String processData = optionScanner.nextLine();


        // Insert process into database.

        PreparedStatement newProcess = connection.prepareStatement(

            "INSERT INTO Process (process_id, department, process_data) "

            + "VALUES ('"+processID+"', '"+departmentNumber+"', '"+processData+"')");

```

```

// Get what type of process is it: Fit, Paint, Cut.

System.out.println("What type of process is it? Enter:");

System.out.println("1 for Fit");

System.out.println("2 for Paint");

System.out.println("3 for Cut");

int processType = intScanner.nextInt();


// Starting inserting the type into database.

if (processType == 1)

{

    System.out.println("Please enter a fit type:");

    String fitType = optionScanner.nextLine();


    PreparedStatement newFit = connection.prepareStatement(

        "INSERT INTO Fit (process_id, fit_type) "

        + "VALUES ('"+processID+"', '"+fitType+"')");


    newProcess.execute();

    newFit.execute();

}

```



```

else if (processType == 2)

{

    System.out.println("Please enter a paint type:");

    String paintType = optionScanner.nextLine();

    System.out.println("Please enter painting method:");

    String paintingMethod = optionScanner.nextLine();


    PreparedStatement newPaint = connection.prepareStatement(

        "INSERT INTO Paint (process_id, paint_type, painting_method) "

        + "VALUES ('"+processID+"', '"+paintType+"', '"+paintingMethod+"')");


    newProcess.execute();

    newPaint.execute();

}

else if (processType == 3)

{

    System.out.println("Please enter a cut type:");

    String cutType = optionScanner.nextLine();

    System.out.println("Please enter machine type:");

    String machineType = optionScanner.nextLine();

```

```

        PreparedStatement newCut = connection.prepareStatement(

            "INSERT INTO Cut (process_id, cutting_type, machine_type) "

            + "VALUES ('"+processID+"', '"+cutType+"', '"+machineType+"')");

        newProcess.execute();

        newCut.execute();

    }

}

// Option Query 5. Enter a new account.

else if (option.equals("5"))

{

    System.out.println("Enter a new account number:");

    int accountNumber = intScanner.nextInt();

    System.out.println("Date account established:");

    String dateEstablished = optionScanner.nextLine();

    PreparedStatement newAccount = connection.prepareStatement(

        "INSERT INTO Account (account_number, date_established) "

```

```

        + "VALUES ('"+accountNumber+"', '"+dateEstablished+"')");

newAccount.execute();

// Get what account is it for: assembly, department, process.

System.out.println("What type of account is it? Enter:");

System.out.println("1 for Assembly");

System.out.println("2 for Department");

System.out.println("3 for Process");

int accountType = intScanner.nextInt();

// Starting inserting the type into database.

if (accountType == 1)

{

    System.out.println("Please enter assembly account details:");

    String details_1 = optionScanner.nextLine();

    PreparedStatement assemblyDetails = connection.prepareStatement(

        "INSERT INTO Assembly_Account (account_number, details_1) "

        + "VALUES ('"+accountNumber+"', '"+details_1+"')");

```

```

        assemblyDetails.execute();

    }

    else if (accountType == 2)

    {

        System.out.println("Please enter department account details:");

        String details_2 = optionScanner.nextLine();

        PreparedStatement departmentDetails = connection.prepareStatement(

            "INSERT INTO Department_Account (account_number, details_2) "

            + "VALUES ('"+accountNumber+"', '"+details_2+"')");

        departmentDetails.execute();

    }

    else if (accountType == 3)

    {

        System.out.println("Please enter process account details:");

        String details_3 = optionScanner.nextLine();

        PreparedStatement processDetails = connection.prepareStatement(

```

```
"INSERT INTO Process_Account (account_number, details_3) "  
+ "VALUES ('"+accountNumber+"', '"+details_3+"')");
```

```
processDetails.execute();
```

```
}
```

```
}
```

// Option Query 6. Enter a new job with process and assembly id as well as when it started.

```
else if (option.equals("6"))
```

```
{
```

```
System.out.println("Please enter a new job number:");
```

```
String jobNumber = optionScanner.nextLine();
```

```
System.out.println("Please enter date of job commenced:");
```

```
String jobStart = optionScanner.nextLine();
```

```
System.out.println("Please enter an existing assembly ID to add new job to:");
```

```
String assemblyID = optionScanner.nextLine();
```

```
System.out.println("Please enter an existing process ID to add new job to:");
```

```
String processID = optionScanner.nextLine();
```

```
PreparedStatement newJob = connection.prepareStatement(
```

```

        "INSERT INTO Job (job_number, assembly_id, process_id, job_commenced) "
        + "VALUES ('"+jobNumber+"', '"+assemblyID+"', '"+processID+"', '"+jobStart+"')");

newJob.execute();

}

// Option Query 7. Enter date of job completion.

else if (option.equals("7"))

{

    // Update table.

    System.out.println("Please enter job number that was completed:");

    String jobNumber = optionScanner.nextLine();

    System.out.println("Please enter the date of job completion:");

    String dateCompleted = optionScanner.nextLine();

    PreparedStatement jobCompletion = connection.prepareStatement(

        "UPDATE Job "

        + "SET job_completed = '"+dateCompleted+"'"

        + "WHERE job_number = '"+jobNumber+"'");

    jobCompletion.execute();

```

```

// Get what job type it is: Fit_Job, Paint_Job, Cut_Job.

System.out.println("What type of job is it? Enter:");

System.out.println("1 for Fit job");

System.out.println("2 for Paint job");

System.out.println("3 for Cut job");

int jobType = intScanner.nextInt();

if (jobType == 1)
{
    System.out.println("Please enter the labor time (minutes) of that job:");

    int laborTime = intScanner.nextInt();

    PreparedStatement jobFit = connection.prepareStatement(

        "INSERT INTO Fit_Job (job_number, labor_time) "

        + "VALUES ('"+jobNumber+"', '"+laborTime+"')");

    jobFit.execute();
}

else if (jobType == 2)

```

```

{

    System.out.println("Please enter the paint color used on that job:");

    String color = optionScanner.nextLine();

    System.out.println("Please enter the volume (liters) of paint on that job:");

    int volume = intScanner.nextInt();

    System.out.println("Please enter the labor time (minutes) of that job:");

    int laborTime = intScanner.nextInt();


    PreparedStatement jobPaint = connection.prepareStatement(

        "INSERT INTO Paint_Job (job_number, color, volume, labor_time) "

        + "VALUES ('"+jobNumber+"', '"+color+"', '"+volume+"', '"+laborTime+"')");


    jobPaint.execute();

}

else if (jobType == 3)

{

    System.out.println("Please enter the machine type used on that job:");

    String machineType = optionScanner.nextLine();

    System.out.println("Please enter the time spent (minutes) on that machine on that

job:");

    int machineTimeUsed = intScanner.nextInt();

```



```

        System.out.println("Please enter materials used on that job:");

        String materialsUsed = optionScanner.nextLine();

        System.out.println("Please enter the labor time (minutes) of that job:");

        int laborTime = intScanner.nextInt();

        PreparedStatement jobCut = connection.prepareStatement(

            "INSERT INTO Cut_Job (job_number, machine_type, machine_time_used,
materials_used, labor_time) "

            + "VALUES (" + jobNumber + ", " + machineType + ", " + machineTimeUsed + ",
" + materialsUsed + ", " + laborTime + ")");

        jobCut.execute();

    }

}

// Option Query 8. Transactions and cost entering.

else if (option.equals("8"))

{

    System.out.println("Please enter a new transaction number:");

    int transactionNumber = intScanner.nextInt();

    System.out.println("Please enter the sup cost in that transaction:");

    int supCost = intScanner.nextInt();

```

```
System.out.println("Please enter the account number this transaction is a part of:");
```

```
int accountNumber = intScanner.nextInt();
```

```
PreparedStatement transaction = connection.prepareStatement(
```

```
    "INSERT INTO Updates (transaction_number, account_number, sup_cost) "
```

```
    + "VALUES (" + transactionNumber + ", " + accountNumber + ", " + supCost + ")");
```

```
transaction.execute();
```

```
// Updating accounts.
```

```
// Get what account type it is: Assembly, Process, Department.
```

```
System.out.println("What type of job is it? Enter:");
```

```
System.out.println("1 for Assembly");
```

```
System.out.println("2 for Process");
```

```
System.out.println("3 for Department");
```

```
int accountType = intScanner.nextInt();
```

```
if (accountType == 1)
```

```
{
```

```
    System.out.println("Please enter the assembly ID associated with the transaction:");
```

```

int assemblyID = intScanner.nextInt();

// Inserting account number and assembly ID and leaving details_1 as null.

PreparedStatement updateAssemblyAccount = connection.prepareStatement(

    "INSERT INTO Assembly_Account (account_number, assembly_id) "

    + "VALUES ('"+accountNumber+"', '"+assemblyID+"')");

updateAssemblyAccount.execute();

// Update the sup cost in the specified assembly account.

PreparedStatement assemblyCost = connection.prepareStatement(

    "Update Assembly_Account "

    + "SET details_1 += '"+supCost+"'"

    + "WHERE assembly_id = '"+assemblyID+"' and account_number =

 '"+accountNumber+'";

assemblyCost.execute();

}

else if (accountType == 2)

{

    System.out.println("Please enter the process ID associated with the transaction:");

```

```

int processID = intScanner.nextInt();

// Inserting account number and process ID and leaving details_3 as null.

PreparedStatement updateProcessAccount = connection.prepareStatement(

    "INSERT INTO Process_Account (account_number, process_id) "

    + "VALUES ('"+accountNumber+"', '"+processID+"')");

updateProcessAccount.execute();

// Update the sup cost in the specified process account.

PreparedStatement processCost = connection.prepareStatement(

    "Update Process_Account "

    + "SET details_3 += '"+supCost+"' "

    + "WHERE process_id = '"+processID+"' and account_number =

"+accountNumber+"");

processCost.execute();

}

else if (accountType == 3)

{

    System.out.println("Please enter the department associated with the transaction:");

```

```

int departmentNumber = intScanner.nextInt();

// Inserting account number and department ID and leaving details_3 as null.

PreparedStatement updateDepartmentAccount = connection.prepareStatement(

    "INSERT INTO Department_Account (account_number, department_number) "

    + "VALUES ('"+accountNumber+"', '"+departmentNumber+"')");

updateDepartmentAccount.execute();

// Update the sup cost in the specified department account.

PreparedStatement departmentCost = connection.prepareStatement(

    "Update Department_Account "

    + "SET details_2 += '"+supCost+"'"

    + "WHERE department_number = '"+departmentNumber+"' and

account_number = '"+accountNumber+"'");

departmentCost.execute();

}

}

// Option Query 9. Get the cost incurred on an assembly ID.

```

```

else if (option.equals("9"))

{

    System.out.println("Enter an assembly_id to retrieve the cost of:");

    int assemblyID = intScanner.nextInt();

    // Define retrieve cost to create and query to get cost of a given assembly ID.

    ResultSet retrieveCost;

    Statement statement = connection.createStatement();

    // Execute the query.

    retrieveCost = statement.executeQuery("SELECT details_1 from Assembly_Account
WHERE assembly_id = '"+assemblyID+"'");

    // Get the cost in an assembly ID if it exists. Else try again.

    if (retrieveCost.next())

    {

        Object object = retrieveCost.getObject(1);

        System.out.printf("%s", object.toString());

        System.out.printf("%n");

    }

    else

```

```

    {

        System.out.println("No such assembly ID exists. Try again.");

    }

}

// Option Query 10. Retrieve total labor time within a department for jobs completed
during a given date.

else if (option.equals("10"))

{

    System.out.println("Please enter a department to get total labor time of jobs:");

    int departmentNumber = intScanner.nextInt();

    System.out.println("Please enter job date of completion in MM/DD/YY format:");

    String dateCompleted = optionScanner.nextLine();


    //select process_id from process where department = departmenNumber

}

// Option Query 11. Enter an assembly_id and get the processes it has.

else if (option.equals("11"))

{

    System.out.println("Please enter an assembly ID to get the processes it has passed:");

    String assemblyID = optionScanner.nextLine();

}

```

// Option Query 12. Retrieve the jobs completed during a given date in a given department.

```
else if (option.equals("12"))  
{  
    System.out.println("Please enter a date of job completion:");  
  
    String jobCompletionDate = optionScanner.nextLine();  
  
    System.out.println("Please enter the department of that job:");  
  
    String jobDepartment = optionScanner.nextLine();  
}
```

// Option Query 13. Retrieve customers (in name order) whose category is in a given range.

```
// Also Query 17 but export customers to data file and not to the screen.  
  
else if (option.equals("13") || option.equals("17"))  
{  
  
    System.out.println("Please enter lower bound category number (1-10):");  
  
    int lowerCategory = intScanner.nextInt();  
  
    System.out.println("Please enter upper bound category number (1-10):");  
  
    int upperCategory = intScanner.nextInt();  
  
    /**
```

* Code inside if statement inspired by this [StackOverflow link](#).

* <https://stackoverflow.com/questions/23291346/printing-all-columns-from-select-query-in-java>

```
*/

if (lowerCategory < upperCategory)

{

    // Define retrieveCustomer to add results to and create a statement to execute query.

    ResultSet retrieveCustomer;

    Statement statement = connection.createStatement();

    retrieveCustomer = statement.executeQuery("SELECT name from Customer "

        + "WHERE category between '"+lowerCategory+"' AND '"+upperCategory+"'");

    // Getting column data.

    ResultSetMetaData metaData = retrieveCustomer.getMetaData();

    int columnCount = metaData.getColumnCount();

    // If option is 13 then print to the screen.

    if (option.equals("13"))

    {

        // While there is another data in column.

        while (retrieveCustomer.next())
```

```

{

    // Print out all columns as long as there is one.

    for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++)

    {

        Object object = retrieveCustomer.getObject(columnIndex);

        System.out.printf("%s", object == null ? "NULL" : object.toString());

    }

    System.out.printf("%n");

}

}

// If option is 17 then send output to a file.

if (option.equals("17"))

{

    System.out.println("Please enter a text file name to export customer data to:");

    String fileName = optionScanner.nextLine();

    String customerData = "";

    try {

        File file = new File(fileName);

```

```

FileOutputStream fileStream = new FileOutputStream(file);

PrintWriter writer = new PrintWriter(fileStream);


// While there is another data in column.

while (retrieveCustomer.next())

{

    // Print out all columns as long as there is one.

    for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++)

    {

        Object object = retrieveCustomer.getObject(columnIndex);

        customerData = customerData + object.toString() + ",";

    }

}


// Write to a specified file and then close and flush everything.

writer.write(customerData);

writer.flush();

writer.close();

fileStream.close();

}

```

```

        catch (Exception e)

        {

            System.out.println("Can't create file.");

        }

    }

}

else

{

    System.out.println("Upper bound category number must be greater. Try again.");

}

}

// Option Query 14. Delete all cut jobs whose job numbers fall in a certain range.

else if (option.equals("14"))

{

    System.out.println("Please enter lower bound job number:");

    int lowerJobNumber = intScanner.nextInt();

    System.out.println("Please enter upper bound job number:");

    int upperJobNumber = intScanner.nextInt();

```

```

if (lowerJobNumber < upperJobNumber)

{

    PreparedStatement deleteCutJob = connection.prepareStatement(

        "DELETE from Cut_Job "

        + "WHERE job_number between '" + lowerJobNumber + "' AND

        '" + upperJobNumber + "'");

    deleteCutJob.execute();

}

else

{

    System.out.println("Upper bound job number must be greater. Try again.");

}

}

// Option Query 15. Change color of a paint job.

else if (option.equals("15"))

{

    System.out.println("Please enter the paint job number to change that color of:");

    int jobNumber = intScanner.nextInt();

    System.out.println("Please enter the color to change to:");

    String newColor = optionScanner.nextLine();

```

```
PreparedStatement updateColor = connection.prepareStatement(
```

```
    "UPDATE Paint_Job "
```

```
    + "SET color = '"+newColor+"'" "
```

```
    + "WHERE job_number = '"+jobNumber+"'");
```

```
updateColor.execute();
```

```
}
```

// Option Query 16. Importing customer info from a text file and entering it in the database.

```
else if (option.equals("16"))
```

```
{
```

```
    System.out.println("Please enter an input file name:");
```

```
    String fileName = optionScanner.nextLine();
```

```
    Scanner readFile;
```

```
    try
```

```
    {
```

```
        // Open the file.
```

```
        readFile = new Scanner(new File(fileName));
```

```
        // Read the file.
```

// Assuming everything is in order with spaces and no commas and one word name
and addresses.

```
while (readFile.hasNext())  
{  
  
    String customerName = readFile.next();  
  
    String customerAddress = readFile.next();  
  
    int category = readFile.nextInt();  
  
    PreparedStatement newCustomer = connection.prepareStatement(  
  
        "INSERT INTO Customer (name, address, category) "  
  
        + "VALUES (" + customerName + ", " + customerAddress + ", " + category + ")");  
  
    newCustomer.execute();  
  
}  
  
// Close the file.  
  
readFile.close();  
  
}  
  
catch (Exception e)  
{  
  
    System.out.println("File was not found. Try again.");
```

```

        }

    }

    else

    {

        System.out.println("Not a valid option. Please enter a number 1-18.");

    }


    // Get another user input as they didn't enter a value from 1-18.

    System.out.println("Please pick another option below:");

    displayOption();

    option = optionScanner.nextLine();

    }

}

```

```

    System.out.println("THANK YOU FOR USING THE JOB-SHOP ACCOUNTING DATABASE
SYSTEM");

```

```

    System.out.println("Closing system...");

}

```

```

/**

```

```

 * Method when called will display query options 1 to 18.

```



```

*/

public static void displayOption()

{

    // Let them choose options (1-18) on what they want to do to the database.

    System.out.println("(1) Enter a new customer (30/day).");

    System.out.println("(2) Enter a new department (infrequent).");

    System.out.println("(3) Enter a new assembly with its customer-name, assembly-details,
assembly-id, "

        + "and date-ordered (40/day).");

    System.out.println("(4) Enter a new process-id and its department together with its type "

        + "and information relevant to the type (infrequent).");

    System.out.println("(5) Create a new account and associate it with the process, assembly, "

        + "or department to which it is applicable (10/day).");

    System.out.println("(6) Enter a new job, given its job-no, assembly-id, process-id, "

        + "and date the job commenced (50/day).");

    System.out.println("(7) At the completion of a job, enter the date it completed "

        + "and the information relevant to the type of job (50/day).");

    System.out.println("(8) Enter a transaction-no and its sup-cost and update all the costs (details)
of the affected accounts by "

        + "adding sup-cost to their current values of details (50/day).");

    System.out.println("(9) Retrieve the cost incurred on an assembly-id (200/day).");

```

```

        System.out.println("(10) Retrieve the total labor time within a department for jobs completed "
            + "in the department during a given date (20/day).");

        System.out.println("(11) Retrieve the processes through which a given assembly-id has passed
so far "
            + "(in date-commenced order) and the department responsible for each process
(100/day).");

        System.out.println("(12) Retrieve the jobs (together with their type information and assembly-
id) "
            + "completed during a given date in a given department (20/day).");

        System.out.println("(13) Retrieve the customers (in name order) whose category is in a given
range (100/day).");

        System.out.println("(14) Delete all cut-jobs whose job-no is in a given range (1/month).");

        System.out.println("(15) Change the color of a given paint job (1/week).");

        System.out.println("(16) Import: enter new customers from a data file until the file is empty "
            + "(the user must be asked to enter the input file name).");

        System.out.println("(17) Export: Retrieve the customers (in name order) "
            + "whose category is in a given range and output them to a data file instead of screen "
            + "(the user must be asked to enter the output file name).");

        System.out.println("(18) Quit");

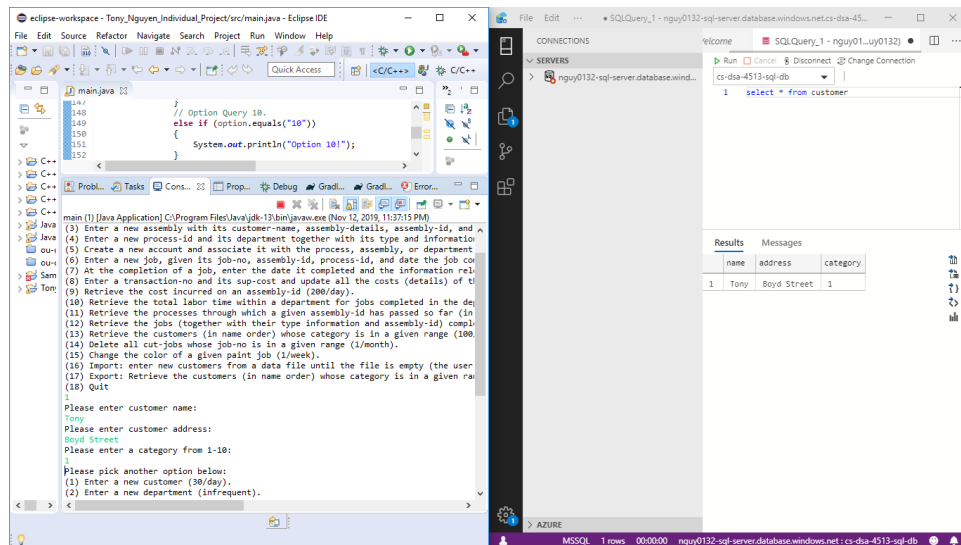
    }

}

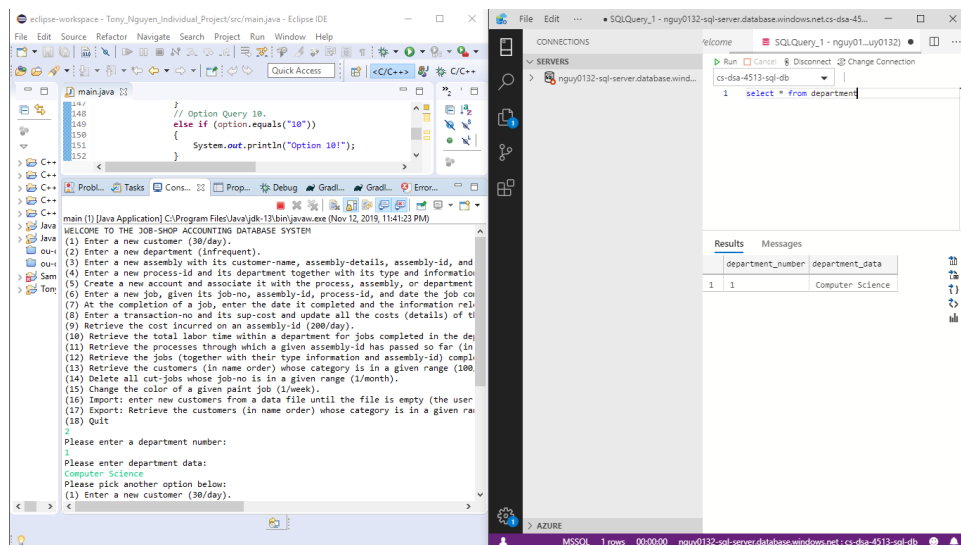
```


Task 6

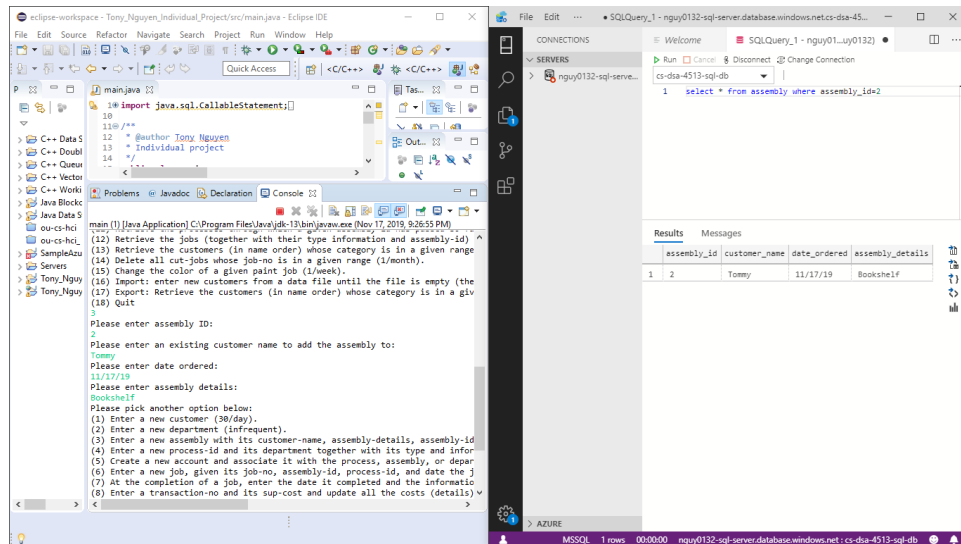
6.1 - Query 1 Screenshot



6.2 - Query 2 Screenshot

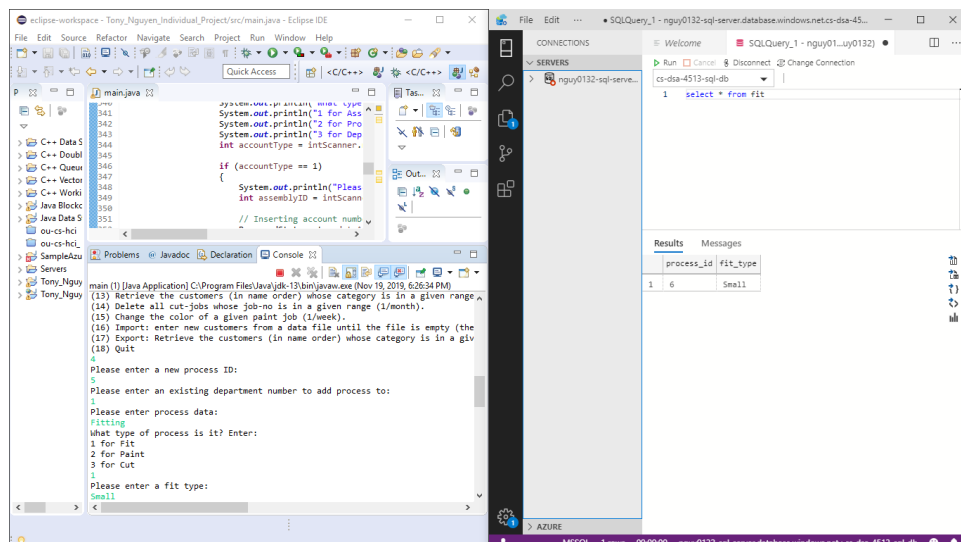


6.3 - Query 3 Screenshot

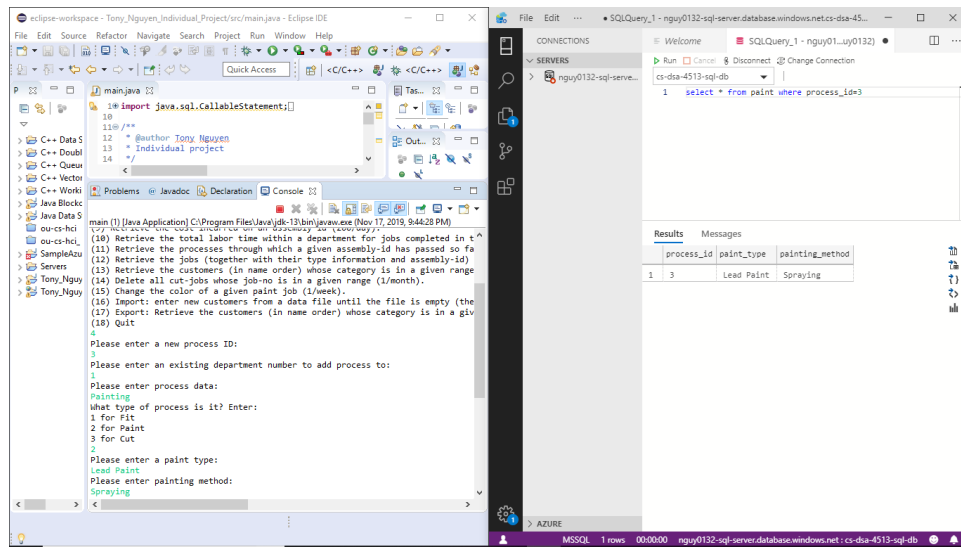


6.4 - Query 4 Screenshot

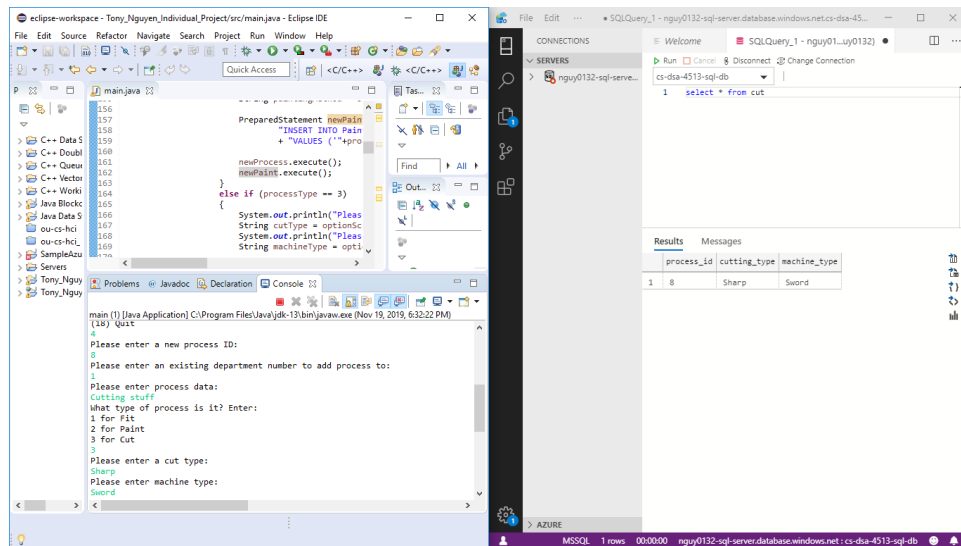
Fit Type Insertion:



Paint Type Insertion:



Cut Type Insertion:



6.5- Query 5 Screenshot

The screenshot shows the Eclipse IDE with a Java application running in the console. The application prompts the user to enter account details. The console output shows the user has entered account number 5 and date 11/17/19. The application then prompts for process account details, and the user enters 100. The application then displays the results of a SQL query in the SQLQuery_1 window.

SQLQuery_1 - nguy0132-sql-server.database.windows.net:cs-dsa-45...

1 select * from process_account

Results Messages

account_number	details_3
5	100

6.6 - Query 6 Screenshot

The screenshot shows the Eclipse IDE with a Java application running in the console. The application prompts the user to enter job details. The console output shows the user has entered job number 6, date 11/17/19, assembly ID 2, and process ID 3. The application then displays the results of a SQL query in the SQLQuery_1 window.

SQLQuery_1 - nguy0132-sql-server.database.windows.net:cs-dsa-45...

1 select * from job where job_number=6

Results Messages

job_number	assembly_id	process_id	job_commenced	job_completed
6	2	3	11/17/19	NULL

6.7 - Query 7 Screenshot

The screenshot shows the Eclipse IDE with a Java application running. The console output is as follows:

```
main [1] [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Nov 17, 2019, 9:51:55 PM)
(10) Retrieve the total labor time within a department for jobs completed in t
(11) Retrieve the processes through which a given assembly-id has passed so fa
(12) Retrieve the jobs (together with their type information and assembly-id)
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range (1/month).
(15) Change the color of a given paint job (1/week).
(16) Import: enter new customers from a data file until the file is empty (the
(17) Export: Retrieve the customers (in name order) whose category is in a giv
(18) Quit

Please enter job number that was completed:
6
Please enter the date of job completion:
12/01/19
What type of job is it? Enter:
1 for Fit job
2 for Paint job
3 for Cut job
Please enter the paint color used on that job:
Red
Please enter the volume (liters) of paint on that job:
12
Please enter the labor time (minutes) of that job:
1500

SQLQuery_1 - nguy0132-sql-server.database.windows.net:cs-dsa-45...
Welcome
SQLQuery_1 - nguy01...uy0132)
Run Cancel Disconnect Change Connection
cs-dsa-4513-sql-db
1 select * from paint_job where job_number=6

Results Messages
job_number color volume labor_time
1 6 Red 12 1500
```

6.8 - Query 8 Screenshot

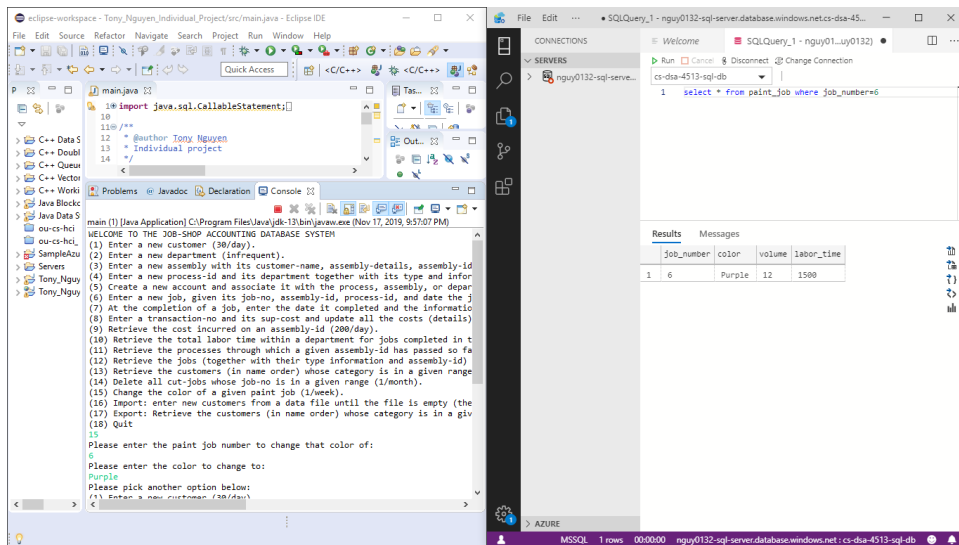
The screenshot shows the Eclipse IDE with a Java application running. The console output is as follows:

```
main [1] [Java Application] C:\Program Files\Java\jdk-13\bin\javaw.exe (Nov 17, 2019, 9:51:55 PM)
(10) Retrieve the total labor time within a department for jobs completed in t
(11) Retrieve the processes through which a given assembly-id has passed so fa
(12) Retrieve the jobs (together with their type information and assembly-id)
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range (1/month).
(15) Change the color of a given paint job (1/week).
(16) Import: enter new customers from a data file until the file is empty (the
(17) Export: Retrieve the customers (in name order) whose category is in a giv
(18) Quit

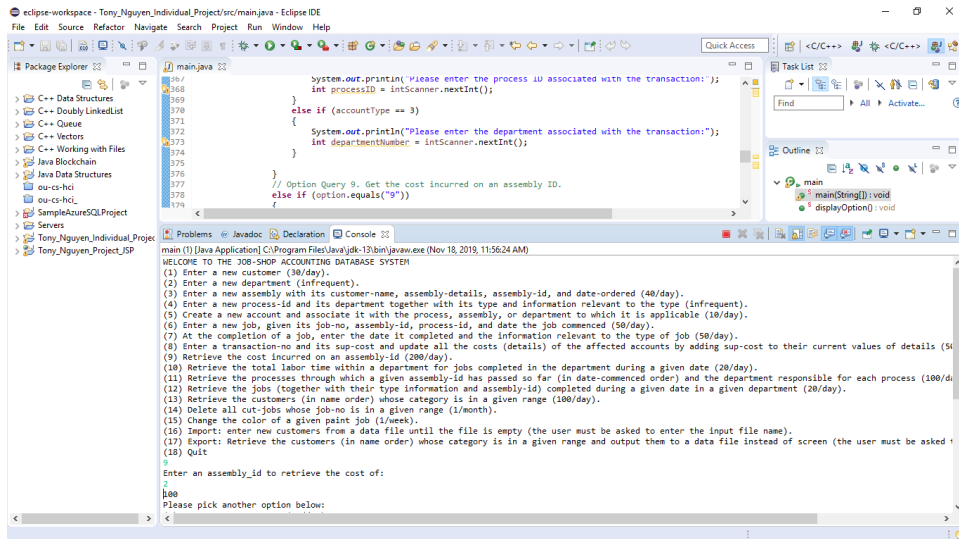
Please enter job number that was completed:
6
Please enter the date of job completion:
12/01/19
What type of job is it? Enter:
1 for Fit job
2 for Paint job
3 for Cut job
Please enter the paint color used on that job:
Red
Please enter the volume (liters) of paint on that job:
12
Please enter the labor time (minutes) of that job:
1500

SQLQuery_1 - nguy0132-sql-server.database.windows.net:cs-dsa-45...
Welcome
SQLQuery_1 - nguy01...uy0132)
Run Cancel Disconnect Change Connection
cs-dsa-4513-sql-db
1 select * from paint_job where job_number=6

Results Messages
job_number color volume labor_time
1 6 Red 12 1500
```

6.9 - Query 9 Screenshot

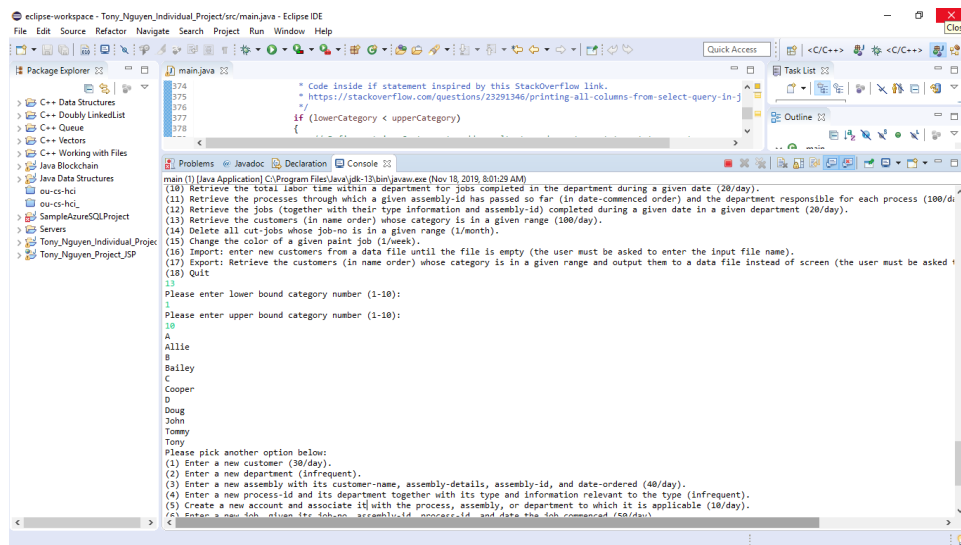


6.10 - Query 10 Screenshot

6.11 - Query 11 Screenshot

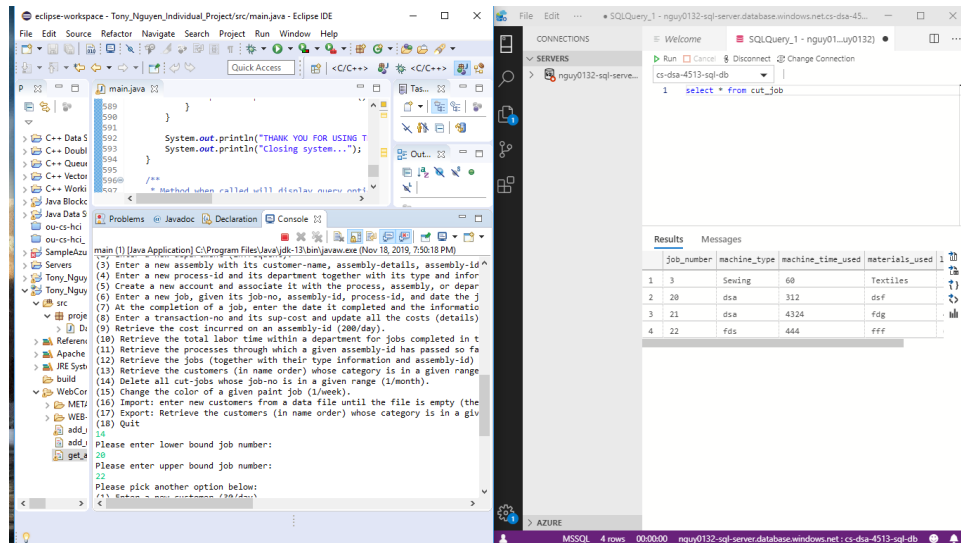
6.12 - Query 12 Screenshot

6.13 - Query 13 Screenshot



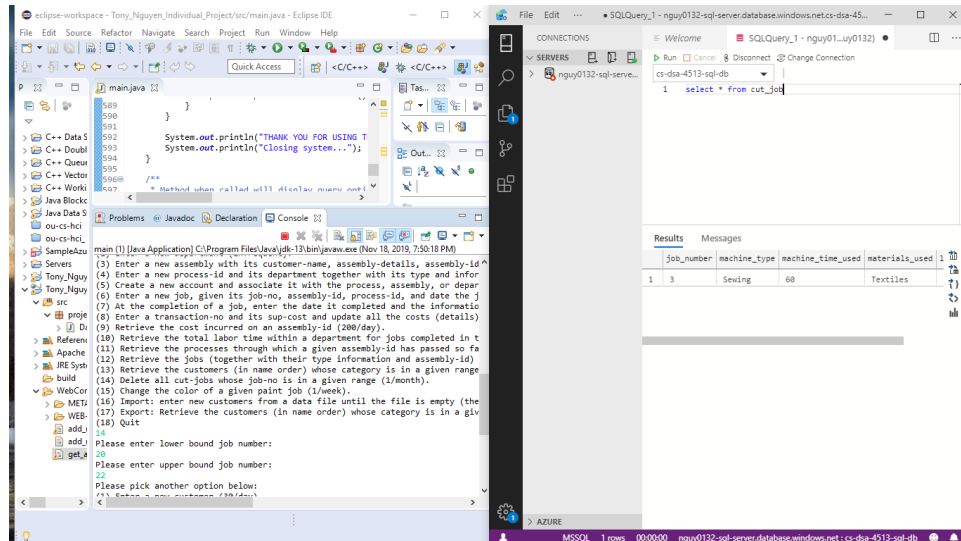
```
main [7] [Java Application] C:\Program Files\Java\jdk-19\bin\java.exe (Nov 18, 2019 8:01:29 AM)
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day).
(11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100/day).
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department (20/day).
(13) Retrieve the customers (in name order) whose category is in a given range (100/day).
(14) Delete all cut-jobs whose job-no is in a given range (1/month).
(15) Change the color of a given paint job (1/week).
(16) Import: enter new customers from a data file until the file is empty (the user must be asked to enter the input file name).
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen (the user must be asked to enter the output file name).
(18) Quit
1
Please enter lower bound category number (1-10):
10
A
Allie
B
Bailey
C
Cooper
D
Doug
John
Tommy
Tony
Please pick another option below:
(1) Enter a new customer (30/day).
(2) Enter a new department (infrequent).
(3) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered (40/day).
(4) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day).
(6) Enter a new job: given its job-no, assembly-id, process-id, and date the job commenced (50/day).
```

6.14 - Query 14 Screenshot



The screenshot shows the Eclipse IDE with a Java application running. The main window displays a menu of options. The console shows the application running and displaying a menu. The SQL editor on the right shows a query: `select * from cut_job`. The results window shows a table with 4 rows and 4 columns: `job_number`, `machine_type`, `machine_time_used`, and `materials_used`.

job_number	machine_type	machine_time_used	materials_used
1	3	Sewing	60
2	20	dsa	312
3	21	dsa	4324
4	22	fds	444



The screenshot shows the Eclipse IDE with a Java application running. The main window displays a menu of options. The console shows the application running and displaying a menu. The SQL editor on the right shows a query: `select * from cut_job`. The results window shows a table with 4 rows and 4 columns: `job_number`, `machine_type`, `machine_time_used`, and `materials_used`.

job_number	machine_type	machine_time_used	materials_used
1	3	Sewing	60
2	20	dsa	312
3	21	dsa	4324
4	22	fds	444

6.15 - Query 15 Screenshot

6.16 - Query 16 Import Screenshot

The screenshot shows the Azure Data Studio interface with a query window titled 'SQLQuery_1 - nguy0132-sql-server.database.windows.net.master (nguy0132)'. The query is `select * from customer;`. The results are displayed in a table with columns 'name', 'address', and 'category'. A Notepad window titled 'reading.txt - Notepad' shows the imported data as a single line: 'Nancy Norman 5 Jamie OKC 3'.

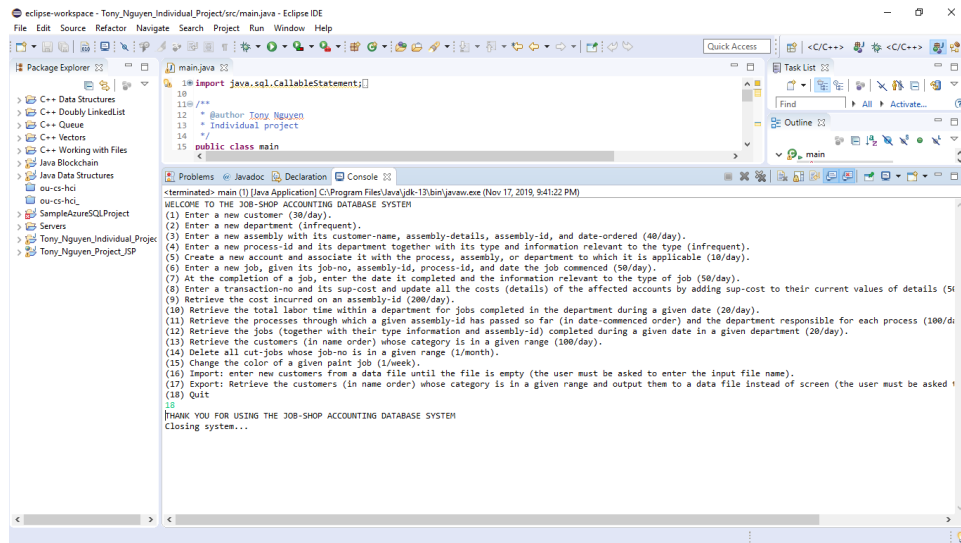
	name	address	category
1	A	Norman OK	10
2	Allie	Norman	9
3	B	Norman OK	9
4	Bailey	Norman	10
5	C	Norman OK	8
6	Copper	Norman	4
7	D	Norman OK	7
8	Doug	Norman	1
9	Jamie	OKC	3
10	John	9713 Norman OK	3
11	Nancy	Norman	5
12	Tommy	2443 Street Ave, Norman, OK	2
13	Tony	Norman, Oklahoma	1

6.17 - Query 17 Export Screenshot

The screenshot shows the Eclipse IDE interface with a Java application running. The console window displays the following output:

```
main([1] [Java Application] C:\Program Files\Java\jdk-12\bin\java.exe (Nov 18 2019 12:54:44 PM)
(5) Create a new account and associate it with the process, assembly,
(6) Enter a new job, given its job-no, assembly-id, process-id, and da
(7) At the completion of a job, enter the date it completed and the in
(8) Enter a transaction-no and its sup-cost and update all the costs (
(9) Retrieve the cost incurred on an assembly-id (200/day).
(10) Retrieve the total labor time within a department for jobs comple
(11) Retrieve the processes through which a given assembly-id has pass
(12) Retrieve the jobs (together with their type information and assem
(13) Retrieve the customers (in name order) whose category is in a giv
(14) Delete all cut-jobs whose job-no is in a given range (1/month).
(15) Change the color of a given paint job (1/week).
(16) Import: enter new customers from a data file until the file is em
(17) Export: Retrieve the customers (in name order) whose category is
(18) Quit
Please enter lower bound category number (1-10):
5
Please enter upper bound category number (1-10):
10
Please enter a text file name to export customer data to:
testing.txt
Please pick another option below:
(1) Enter a new customer (30/day),
```

6.18 - Query 18 Quit Screenshot



Task 7

7.1 Web Application Source Program

DataHandler.java

```
package project_jsp;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class DataHandler {
    private Connection conn;

    // Azure SQL connection credentials.
    private String server = "nguy0132-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "nguy0132";
    private String password = "sN8!CVD$NBvib*7J9L23";

    // Resulting connection string.
    final private String url =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt
=true;"
        + "hostNameInCertificate=*.database.windows.net;loginTimeout=30;", server,
database, username, password);
```

```

// Initialize and save the database connection.

private void getDBConnection() throws SQLException {
    if (conn != null) {
        return;
    }

    this.conn = DriverManager.getConnection(url);
}

// Return the result of selecting everything from the table.
public ResultSet getAllCustomers() throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM Customer";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    return stmt.executeQuery();
}

// Inserts a record into the Customer table with the given attribute values.
public boolean addCustomer(String customerName, String customerAddress, int
category) throws SQLException {

    // Prepare the database connection.
    getDBConnection();

    // Prepare the SQL statement to insert into Customer table.

```

```

        final String sqlQuery =

            "INSERT INTO Customer (name, address, category) "

            + "VALUES ('"+customerName+"', '"+customerAddress+"',

"+category+"')";

        // Execute query.

        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        return stmt.executeUpdate() == 1;

    }

}

```

add_customer_form.jsp

```

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Add Customer</title>

</head>

<body>

<h2>Add Customer</h2>

<!--

Form for collecting user input for the new customer record.

Upon form submission, add_customer.jsp file will be invoked.

-->

<form action="add_customer.jsp">

```


<!-- The form organized in an HTML table for better clarity. -->

```
<table border=1>
```

```
<tr>
```

```
<th colspan="2">Enter the Customer Data:</th>
```

```
</tr>
```

```
<tr>
```

```
<td>Name:</td>
```

```
<td><div style="text-align: center;">
```

```
<input type=text name=name>
```

```
</div></td>
```

```
</tr>
```

```
<tr>
```

```
<td>Address:</td>
```

```
<td><div style="text-align: center;">
```

```
<input type=text name=address>
```

```
</div></td>
```

```
</tr>
```

```
<tr>
```

```
<td>Category:</td>
```

```
<td><div style="text-align: center;">
```

```
<input type=text name=category>
```

```
</div></td>
```

```
</tr>
```

```
<tr>
```

```
<td><div style="text-align: center;">
```

```

<input type=reset value=Clear>
</div> </td>

<td> <div style="text-align: center;">
<input type=submit value=Insert>
</div> </td>

</tr>

</table>

</form>

</body>

</html>

```

add_customer.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
<body>
<%@page import="project_jsp.DataHandler"%>
<%@page import="java.sql.ResultSet"%>

```

```

<%@page import="java.sql.Array"%>

<%

// The handler is the one in charge of establishing the connection.

DataHandler handler = new DataHandler();

// Get the attribute values passed from the input form.

String name = request.getParameter("name");

String address = request.getParameter("address");

String category = request.getParameter("category");

/*

* If the user hasn't filled out all the name, address, and category. This is very simple
checking.

*/

if (name.equals("") || address.equals("") || category.equals("")) {

response.sendRedirect("add_customer_form.jsp");

} else {

int customerCategory = Integer.parseInt(category);

// Now perform the query with the data from the form.

boolean success = handler.addCustomer(name, address, customerCategory);

if (!success) { // Something went wrong

%>

<h2>There was a problem inserting the course</h2>

<%

} else { // Confirm success to the user

%>

<h2>The Customer Data:</h2>

```

```
<ul>

<li>Name: <%=name%> </li>

<li>Address: <%=address%> </li>

<li>Category: <%=category%> </li>

</ul>

<h2>Was successfully inserted.</h2>

<a href="get_all_customers.jsp">See all customers.</a>

<%

}

}

%>

</body>

</html>
```

get_all_customers.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Customers</title>

</head>

<body>

<%@page import="project_jsp.DataHandler"%>
```

```

<%@page import="java.sql.ResultSet"%>

<%
// We instantiate the data handler here, and get all the Customers from the database.
final DataHandler handler = new DataHandler();
final ResultSet customers = handler.getAllCustomers();
%>

<!-- The table for displaying all the customer records -->
<table cellspacing="2" cellpadding="2" border="1">
<tr> <!-- The table headers row -->
<td align="center">
<h4>Name</h4>
</td>
<td align="center">
<h4>Address</h4>
</td>
<td align="center">
<h4>Category</h4>
</td>
</tr>
<%
while(customers.next()) { // For each customer record returned...
// Extract the attribute values for every row returned
final String name = customers.getString("name");
final String address = customers.getString("address");
final String category = customers.getString("category");
out.println("<tr>"); // Start printing out the new table row

```

```
out.println( // Print each attribute value
"<td align=\"center\">" + name +
"</td><td align=\"center\"> " + address +
"</td><td align=\"center\"> " + category + "</td>");
out.println("</tr>");
}
%>

</table>

</body>

</html>
```