Individual Project

Job-Shop Accounting Database Implementation

Database Management Systems (DSA 4513-995)

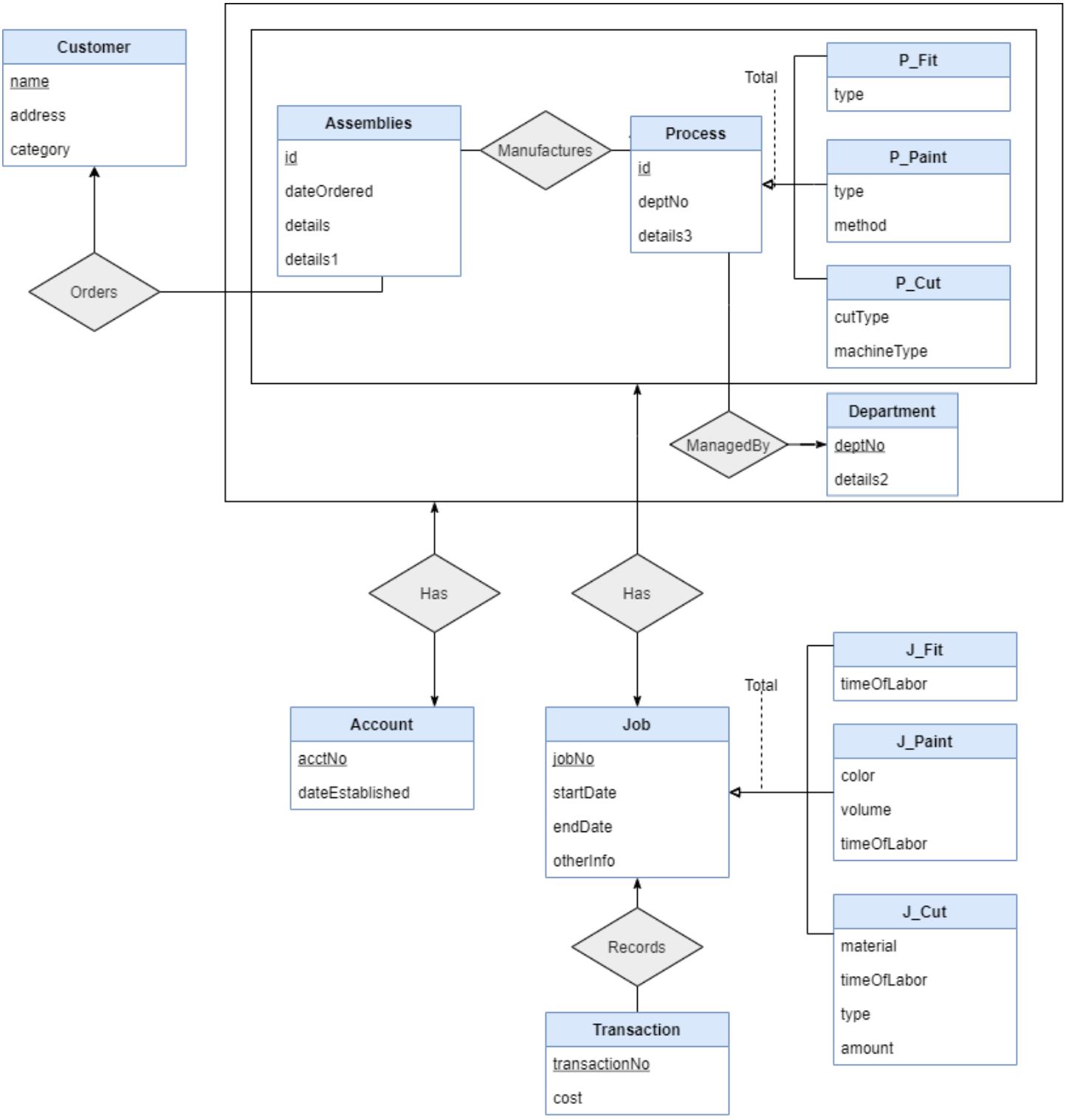
Fall 2021

Instructor Dr. Le Gruenwald

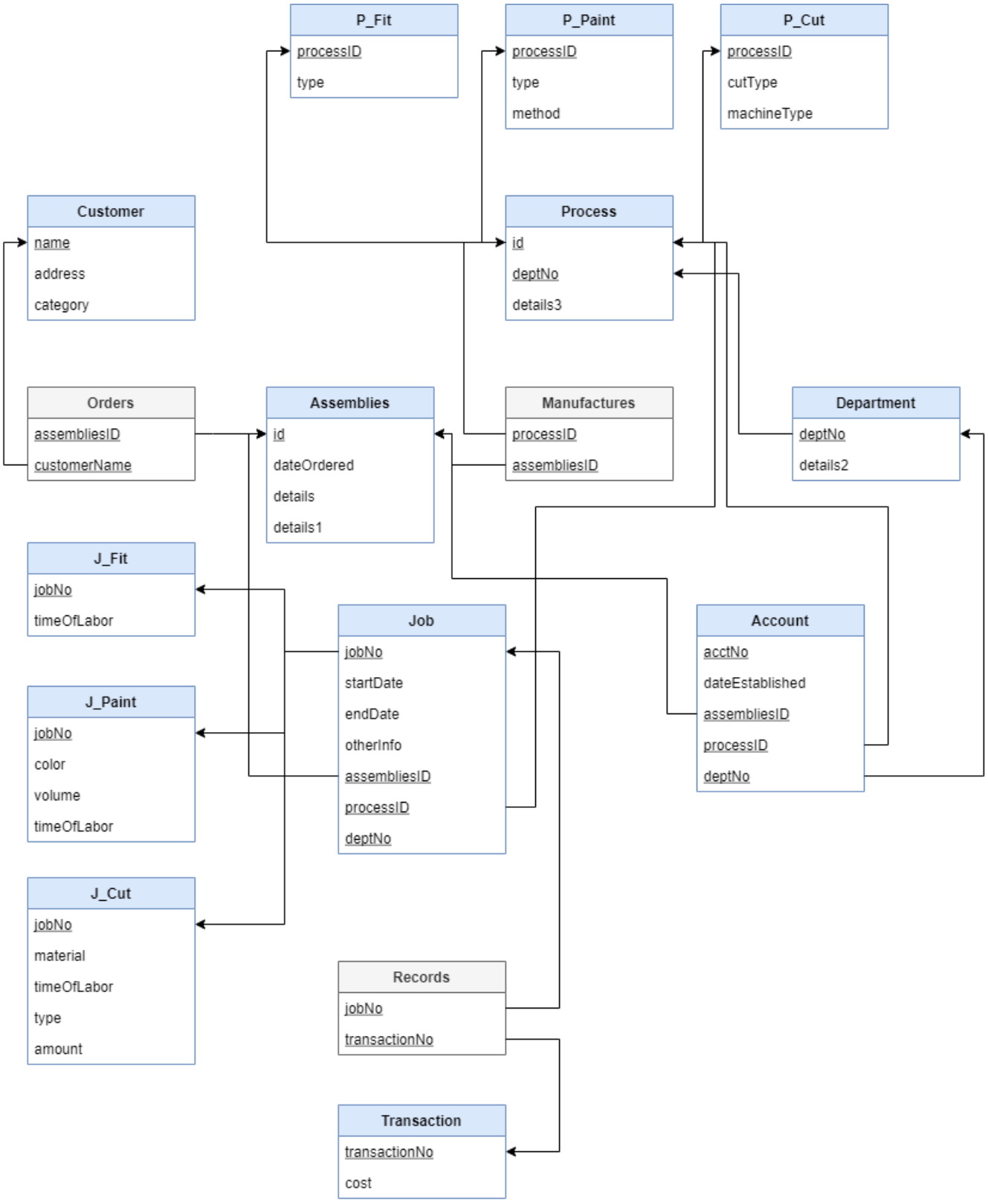
Daniel Carpenter
OU ID: 113009743 | daniel.carpenter@ou.edu

Table of Contents

1.1. Entity Relation Diagram (ERD)



1.2. Relational Database Schema



Task 2. Data Dictionary

Table Name	Attribute Name	Attribute Type	Attribute Sizes (bytes)	Constraint
Customer	name	VARCHAR(255)	1 byte	Primary Key
Customer	address	VARCHAR(255)	1 byte	
Customer	category	INT	4 bytes	from 1 to 10
Orders	customerName	VARCHAR(255)	1 byte	Foreign Key
Orders	assembliesID	INT	4 bytes	Foreign Key
Assemblies	id	INT	4 bytes	Primary Key
Assemblies	dateOrdered	DATE	3 bytes	
Assemblies	details	VARCHAR(255)	1 byte	
Assemblies	details1	REAL	4 - 8 bytes	
Manufactures	assembliesID	INT	4 bytes	Foreign Key
Manufactures	processID	INT	4 bytes	Foreign Key
Process	id	INT	4 bytes	Primary Key
Process	deptNo	INT	4 bytes	Foreign Key
Process	details3	REAL	4 - 8 bytes	
P_Fit	processID	INT	4 bytes	Foreign Key
P_Fit	type	VARCHAR(255)	1 byte	
P_Paint	processID	INT	4 bytes	Foreign Key
P_Paint	type	VARCHAR(255)	1 byte	
P_Paint	method	VARCHAR(255)	1 byte	
P_Cut	processID	INT	4 bytes	Foreign Key
P_Cut	cutType	VARCHAR(255)	1 byte	
P_Cut	machineType	VARCHAR(255)	1 byte	
Department	deptNo	INT	4 bytes	Primary Key
Department	details2	REAL	4 - 8 bytes	

Task 2. Data Dictionary *(Continued)*

Table Name	Attribute Name	Attribute Type	Attribute Sizes (bytes)	Constraint
Job	JobNo	INT	4 bytes	Primary Key
Job	startDate	DATE	3 bytes	
Job	endDate	DATE	3 bytes	
Job	otherInfo	VARCHAR(255)	1 byte	
Job	assembliesID	INT	4 bytes	Foreign Key
Job	processID	INT	4 bytes	Foreign Key
Job	deptNo	INT	4 bytes	Foreign Key
J_Fit	JobNo	INT	4 bytes	Foreign Key
J_Fit	timeOfLabor	INT	4 bytes	
J_Paint	JobNo	INT	4 bytes	Foreign Key
J_Paint	color	VARCHAR(255)	1 byte	
J_Paint	volume	REAL	4 - 8 bytes	
J_Paint	timeOfLabor	INT	4 bytes	
J_Cut	JobNo	INT	4 bytes	Foreign Key
J_Cut	material	VARCHAR(255)	1 byte	
J_Cut	timeOfLabor	INT	4 bytes	
J_Cut	type	VARCHAR(255)	1 byte	
J_Cut	amount	REAL	4 - 8 bytes	
Records	jobNo	INT	4 bytes	Foreign Key
Records	transactionNo	INT	4 bytes	Foreign Key
Transaction	transactionNo	INT	4 bytes	Primary Key
Transaction	cost	REAL	4 - 8 bytes	
Account	acctNo	INT	4 bytes	Primary Key
Account	dateEstablished	DATE	3 bytes	
Account	assembliesID	INT	4 bytes	Foreign Key
Account	processID	INT	4 bytes	Foreign Key
Account	deptNo	INT	4 bytes	Foreign Key

3.1. Storage Structures for Tables

Table Nam	Query #	Query Type	Search Key	Query Frequency Per Day	Query Frequency Per Mont	Selected File Organization	Justification
Account	5	Insertion	acctNo	10	310	B+ Tree on search key: acctNo	Since both range and random searches total to be the most frequent query, ordered indices are to be preferred. B+ Trees will efficiently handle both of these search types. Additionally, since insertion makes up the remaining queries, B+ Trees fortunately will be preferred over B Trees.
Account	8	Random Search	acctNo	50	1,550		
Account	9	Random Search	acctNo	200	6,200		
Account	10	Random Search	acctNo	20	620		
Account	11	Range Search	acctNo	100	3,100		
Account	12	Range Search	acctNo	20	620		
Assemblies	4	Insertion	id	40	1,240	B+ Tree on search key: id	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Assemblies	5	Random Search	id	10	310		
Assemblies	11	Range Search	id	100	3,100		
Assemblies	8	Insertion	details1	50	1,550	Heap on search key details1	Since heaps handle insertion very well, use heap index on search key
Customer	1	Insertion	name	30	930	B+ Tree on search key: name	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Customer	4	Insertion	name	40	1,240		
Customer	13	Range Search	name	100	3,100		
Department	2	Insertion	deptNo	Infrequent	Infrequent	B+ Tree on search key: deptNo	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Department	3	Insertion	deptNo	Infrequent	Infrequent		
Department	5	Random Search	deptNo	10	310		
Department	10	Random Search	deptNo	20	620		
Department	11	Range Search	deptNo	100	3,100		
Department	12	Range Search	deptNo	20	620		
Department	8	Insertion	details2	50	1,550	Heap on search key details2	Since heaps handle insertion very well, use heap index on search key

3.1. Storage Structures for Tables *(Continued)*

Table Nam	Query #	Query Type	Search Key	Query Frequency Per Day	Query Frequency Per Mont	Selected File Organization	Justification
J_Cut	10	Random Search	jobNo	20	620	Static Hash Table on search key: JobNo	Since only random search, and the table stores attributes of the main table, a hash table would work well for retrieving the search key. Since no insertion anticipated, a static hash function will work.
J_Fit	10	Random Search	jobNo	20	620	Static Hash Table on search key: JobNo	Since only random search, and the table stores attributes of the main table, a hash table would work well for retrieving the search key. Since no insertion anticipated, a static hash function will work.
J_Paint	10	Random Search	jobNo	20	620	Dynamic Hash Table on search key: JobNo	Since only random search, and the table stores attributes of the main table, a hash table would work well for retrieving the search key. Since no insertion anticipated, a static hash function will work.
J_Paint	15	Insertion/Deletion	color	1/7	4	B+ Tree on search key: color	Since B+ Trees can handle insertion and deletion well, we will index using a B+ Tree on the effected attributed, color.
Job	7	Insertion	dateCompleted, otherInfo	50	1,550	Heap on search key: dateCompleted, otherInfo	Since heaps handle insertion very well, use heap index on search key
Job	6	Insertion	JobNo	50	1,550	B+ Tree on search key: JobNo	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Job	8	Random Search	JobNo	50	1,550		
Job	9	Random Search	JobNo	200	6,200		
Job	10	Random Search	JobNo	20	620		
Job	11	Range Search	JobNo	100	3,100		
Job	12	Range Search	JobNo	20	620		
Job	14	Deletion	JobNo	1/31	1		

3.1. Storage Structures for Tables *(Continued)*

Table Nam	Query #	Query Type	Search Key	Query Frequency Per Day	Query Frequency Per Monti	Selected File Organization	Justification
Manufactures	4	Random Search	assembliesID	40	1,240	B+ Tree on search key: assembliesID	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Manufactures	11	Range Search	assembliesID	100	3,100		
Orders	4	Insertion	customerName	40	1,240	Heap on search key: customerName	Since insertion is the only query type, Heaps offer fast processing since no searching happens before inserting.
Process	3	Insertion	id	Infrequent	Infrequent	B+ Tree on search key: id	Since range searches are the most common query, ordered indices are to be preferred. Also, B+ Trees efficiently handle range searches.
Process	4	Random Search	id	40	1,240		
Process	5	Random Search	id	10	310		
Process	11	Range Search	id	100	3,100		
Process	8	Insertion	details3	50	1,550	Heap on search key details3	Since heaps handle insertion very well, use heap index on search key
P_Cut	3	Insertion	processID	Infrequent	Infrequent	Heap on search key: processID	Since insertion is the only query type, Heaps offer fast processing since no searching happens before inserting.
P_Fit	3	Insertion	processID	Infrequent	Infrequent	Heap on search key: processID	Since insertion is the only query type, Heaps offer fast processing since no searching happens before inserting.
P_Paint	3	Insertion	processID	Infrequent	Infrequent	Heap on search key: processID	Since insertion is the only query type, Heaps offer fast processing since no searching happens before inserting.
Records	8	Random Search	jobNo	50	1,550	Hash Table on search key: jobNo	Since random search occurs 250 times per day, a hash table would work well for retrieving the search key.
Records	9	Random Search	jobNo	200	6,200		

3.1. Storage Structures for Tables *(Continued)*

Table Nam	Query #	Query Type	Search Key	Query Frequency Per Day	Query Frequency Per Mont	Selected File Organization	Justification
Transaction	8	Insertion	transactionNo	50	1,550	Dynamic Hash Table	Since random search occurs 200 times per day (versus insertion in query number 8 of 50 times per day), a hash table would work well for retrieving the search key. Since insertions occur often though, a dynamic hash function is likely necessary.
Transaction	9	Random Search	transactionNo	200	6,200	on search key: transactionNo	

3.2. Storage Structures for Table (Azure SQL Database)

B+ Tree Data Structure

In order to implement a B+ Tree, basic index creation in SQL Server are B+ Trees. Therefore, for all recommended indices with B+ Tree implementations use a simple index on the attribute.

Source: https://sqlity.net/en/2445/b-plus-tree/

Heap Data Structure

By default, a table is a Heap when created.

Source: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes?view=sql-server-ver15

Static Hash Table Data Structure

It is possible to create Hash tables for memory optimized tables. To implement this hash index, you would alter (either immediately after creation of the table, or at some future point) the table and specify the number of buckets for each bin. However, I am unaware of how to create this memory optimized table; per recommendations of Microsoft for an alternative, I have instead created a non-clustered index on the specified attribute(s).

Source: https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15#hash_index

Dynamic Hash Table Data Structure

The process is the same as creating a static hash index, but you would dynamically set the number of buckets (or bins). Note that you would need to somehow schedule some update of these bins, which I theorize could be possible with a scheduled stored procedure. If I were to implement it, I would use some hash function based on the number of bins (e.g. using a modulo function) for the table at that point in time. Due to the issues stated in the Static Hash section, I have chosen to use a non-clustered index for each specified attribute.

Task 4. SQL Statements to Create Tables in Azure SQL Database

```
-- ================================================================================
-- @class:  DSA 4513
-- @asnmt:  Class Project
-- @task:   4
-- @author: Daniel Carpenter, ID: 113009743
-- @description:
--     Queries to create tables, contraints, and indiceds of the job-shop
--     accounting system database
-- ================================================================================

-- Use Daniel Carpenter's Azure Database for all queries
USE [cs-dsa-4513-sql-db]

-- ================================================================================
-- DROP TABLES IF ALREADY EXISTING
-- Note: Using schema named 'Project' for this Project's Tables
-- ================================================================================

    -- Tables ------------------------------------------------------------------
    DROP TABLE IF EXISTS [Project].[P_Fit];
    DROP TABLE IF EXISTS [Project].[P_Paint];
    DROP TABLE IF EXISTS [Project].[P_Cut];
    DROP TABLE IF EXISTS [Project].[J_Fit];
    DROP TABLE IF EXISTS [Project].[J_Paint];
    DROP TABLE IF EXISTS [Project].[J_Cut];
    DROP TABLE IF EXISTS [Project].[Orders];
    DROP TABLE IF EXISTS [Project].[Manufactures];
    DROP TABLE IF EXISTS [Project].[ManagedBy];
    DROP TABLE IF EXISTS [Project].[Records];
    DROP TABLE IF EXISTS [Project].[Customer];
    DROP TABLE IF EXISTS [Project].[Job];
    DROP TABLE IF EXISTS [Project].[Transaction];
    DROP TABLE IF EXISTS [Project].[Account];
```

```
    DROP TABLE IF EXISTS [Project].[Assemblies];
    DROP TABLE IF EXISTS [Project].[Process];
    DROP TABLE IF EXISTS [Project].[Department];


-- ================================================================================
-- CREATE TABLES, INDICES, AND CONSTRAINTS FOR JOB-SHOP DATABASE
-- ================================================================================

-- Create the Customer Table ------------------------------------------------
CREATE TABLE [Project].Customer (
    [name]          VARCHAR(255) PRIMARY KEY,
    [address]       VARCHAR(255) NOT NULL,
    [category]      INT NOT NULL,
    CONSTRAINT      CTGRY_RANGE CHECK(category >= 1 AND category <= 10)
)
-- Create B+ Tree Index on name of Customer table
CREATE INDEX [IDX_Customer_ON_name] ON [Project].Customer([name]);


-- Create the Assemblies Table ----------------------------------------------
CREATE TABLE [Project].Assemblies (
    [id]            INT PRIMARY KEY,
    [dateOrdered]   DATE NOT NULL,
    [details]       VARCHAR(255) DEFAULT NULL,
    [details1]      REAL DEFAULT 0,
    CONSTRAINT      [NON_NEG_id_Assemblies] CHECK(id > 0)
)
-- Create B+ Tree Index on id of Assemblies table
CREATE INDEX [IDX_Assemblies_ON_id] ON [Project].Assemblies(id);


-- Create the Orders Table --------------------------------------------------
CREATE TABLE [Project].Orders (
    [customerName]  VARCHAR(255) FOREIGN KEY REFERENCES [Project].Customer,
```

```
    [assembliesID]  INT NOT NULL FOREIGN KEY REFERENCES [Project].Assemblies
)

-- Create the Fit Table for Processes ------------------------------------
CREATE TABLE [Project].P_Fit (
    [processID]             INT PRIMARY KEY,
    [type]          VARCHAR(255) DEFAULT NULL,
    CONSTRAINT      [NON_NEG_id_P_Fit] CHECK(processID > 0)
)


-- Create the Paint Table for Processes ----------------------------------
CREATE TABLE [Project].P_Paint (
    [processID]             INT PRIMARY KEY,
    [type]          VARCHAR(255) DEFAULT NULL,
    [method]        VARCHAR(255) DEFAULT NULL,
    CONSTRAINT      [NON_NEG_id_P_Paint] CHECK(processID > 0)
)


-- Create the Cut Table for Processes ------------------------------------
CREATE TABLE [Project].P_Cut (
    [processID]             INT PRIMARY KEY,
    [cutType]       VARCHAR(255) DEFAULT NULL,
    [machineType]   VARCHAR(255) DEFAULT NULL,
    CONSTRAINT      [NON_NEG_id_P_Cut] CHECK(processID > 0)
)


-- Create the Table for Departments---------------------------------------
CREATE TABLE [Project].Department (
    [deptNo]        INT PRIMARY KEY,
    [details2]      REAL DEFAULT 0,
    CONSTRAINT      [NON_NEG_deptNo_Department] CHECK(deptNo > 0)
)
```

```sql
-- Create B+ Tree Index on deptNo of the Department table
CREATE INDEX [IDX_Department_ON_deptNo] ON [Project].Department(deptNo);


-- Create the Table for Processes ------------------------------------------
CREATE TABLE [Project].Process (
    [id]            INT PRIMARY KEY,
    [deptNo]        INT FOREIGN KEY REFERENCES [Project].Department,
    [details3]      REAL DEFAULT 0,
    CONSTRAINT      [NON_NEG_id_Process] CHECK(id > 0)
)


-- Create the Relation Table between the Process and Assemblies tables ------
CREATE TABLE [Project].Manufactures (
    [assembliesID]  INT NOT NULL FOREIGN KEY REFERENCES [Project].Assemblies,
    [processID]     INT NOT NULL FOREIGN KEY REFERENCES [Project].Process
)
--Create B+ Tree Index on deptNo of the Manufactures table
CREATE INDEX [IDX_Manufactures_ON_assembliesID] ON [Project].Manufactures(assembliesID);


-- Create the Table for Jobs -----------------------------------------------
CREATE TABLE [Project].Job (
    [jobNo]         INT PRIMARY KEY,
    [startDate]     DATE NOT NULL,
    [endDate]       DATE DEFAULT NULL,
    [otherInfo]     VARCHAR(255) DEFAULT NULL,
    [assembliesID]  INT NOT NULL FOREIGN KEY REFERENCES [Project].Assemblies,
    [processID]     INT NOT NULL FOREIGN KEY REFERENCES [Project].Process,
    [deptNo]        INT NOT NULL FOREIGN KEY REFERENCES [Project].Department,
    CONSTRAINT      [NON_NEG_JobNo_Job] CHECK(JobNo > 0),
    CONSTRAINT      [POS_VAR_DATES_Job] CHECK(endDate >= startDate)
)
-- Create B+ Tree Index on id of Job table
```

```
CREATE INDEX [IDX_Job_ON_JobNo] ON [Project].Job(JobNo);


-- Create the Fit Table for Jobs ------------------------------------------
CREATE TABLE [Project].J_Fit (
    [jobNo]         INT FOREIGN KEY REFERENCES [Project].Job,
    [timeOfLabor]   INT DEFAULT 0,
    CONSTRAINT      [NON_NEG_id_J_Fit] CHECK(jobNo > 0),
    CONSTRAINT      [NON_NEG_timeOfLabor_J_Fit] CHECK([timeOfLabor] >= 0)
)
-- Create non-clustered index in place of ideal static hash table
CREATE NONCLUSTERED INDEX [IDX_J_Fit_ON_id] ON [Project].J_Fit(jobNo);


-- Create the Paint Table for Jobs ----------------------------------------
CREATE TABLE [Project].J_Paint (
    [jobNo]         INT FOREIGN KEY REFERENCES [Project].Job,
    [color]         VARCHAR(255) DEFAULT NULL,
    [volume]        REAL DEFAULT 0 NOT NULL,
    [timeOfLabor]   INT DEFAULT 0 NOT NULL,
    CONSTRAINT      [NON_NEG_id_J_Paint] CHECK(jobNo > 0),
    CONSTRAINT      [NON_NEG_timeOfLabor_J_Paint] CHECK([timeOfLabor] >= 0)
)
-- Create non-clustered index in place of dunamic hash table on id
CREATE NONCLUSTERED INDEX [IDX_J_Paint_ON_id] ON [Project].J_Paint(jobNo);

-- Create B+ tree index for color
CREATE INDEX [IDX_J_Paint_ON_color] ON [Project].J_Paint(color);

-- Create the Cut Table for Jobs ------------------------------------------
CREATE TABLE [Project].J_Cut (
    [jobNo]         INT FOREIGN KEY REFERENCES [Project].Job,
    [material]      VARCHAR(255) DEFAULT NULL,
    [timeOfLabor]   INT DEFAULT 0 NOT NULL,
    [type]          VARCHAR(255) DEFAULT NULL,
```

```
    [amount]        REAL DEFAULT 0 NOT NULL,
    CONSTRAINT      [NON_NEG_id_J_Cut]          CHECK(jobNo > 0),
    CONSTRAINT      [NON_NEG_timeOfLabor_J_Cut] CHECK([timeOfLabor] >= 0)
)
-- Create non-clustered index in place of ideal static hash table
CREATE NONCLUSTERED INDEX [IDX_J_Cut_ON_id] ON [Project].J_Cut(jobNo);

-- Create the Table for Transactions ---------------------------------------
CREATE TABLE [Project].[Transaction] (
    [transactionNo] INT PRIMARY KEY,
    [cost]          REAL DEFAULT 0 NOT NULL,
    CONSTRAINT      [NON_NEG_transactionNo_Transaction] CHECK(transactionNo > 0),
    CONSTRAINT      [NON_NEG_cost_Transaction]          CHECK(cost >= 0)
)
-- Create non-clustered index in place of ideal dynamic hash table
CREATE NONCLUSTERED INDEX [IDX_Transaction_ON_transactionNo] ON [Project].[Transaction](transactionNo);


-- Create the Relation Table for the Job and Transactions tables -------------
CREATE TABLE [Project].Records (
    [jobNo]         INT NOT NULL FOREIGN KEY REFERENCES [Project].Job,
    [transactionNo] INT NOT NULL FOREIGN KEY REFERENCES [Project].[Transaction]
)
-- Create non-clustered index in place of ideal dynamic hash table
CREATE NONCLUSTERED INDEX [IDX_Records_ON_jobNo] ON [Project].Records(jobNo);


-- Create the Table for Accounts -------------------------------------------
CREATE TABLE [Project].Account (
    [acctNo]            INT PRIMARY KEY,
    [dateEstablished]   DATE NOT NULL,
    [assembliesID]      INT NOT NULL FOREIGN KEY REFERENCES [Project].Assemblies,
    [processID]         INT NOT NULL FOREIGN KEY REFERENCES [Project].Process,
    [deptNo]            INT NOT NULL FOREIGN KEY REFERENCES [Project].Department,
    CONSTRAINT          [NON_NEG_acctNo_Account] CHECK(acctNo > 0)
```

```
)
-- Create B+ Tree Index on id of Accounts table
CREATE INDEX [IDX_Account_ON_acctNo] ON [Project].Account(acctNo);
```

5.1. SQL Statements and Stored Procedures Implementing Queries 1 – 15

```
-- ========================================================================
-- @class:  DSA 4513
-- @asnmt:  Class Project
-- @task:   5 (a)
-- @author: Daniel Carpenter, ID: 113009743
-- @description:
--     Queries to create procedures for queries 1 - 15 and 17 of the
--     job-shop accounting system database
-- ========================================================================

-- Use Daniel Carpenter's Azure Database for all queries
USE [cs-dsa-4513-sql-db]

-- ========================================================================
-- DROP PROCEDURES IF ALREADY EXISTING
-- Note: Using schema named 'Project' for this Project's Tables
-- ========================================================================

    -- Drop Procedures  ----------------------------------------------------
    DROP PROCEDURE IF EXISTS [Project].addCustomer;         -- 1
    DROP PROCEDURE IF EXISTS [Project].addDepartment;       -- 2
    DROP PROCEDURE IF EXISTS [Project].addProcess;          -- 3
    DROP PROCEDURE IF EXISTS [Project].addAssembly;         -- 4
    DROP PROCEDURE IF EXISTS [Project].addAccount;          -- 5
    DROP PROCEDURE IF EXISTS [Project].addJob;              -- 6
    DROP PROCEDURE IF EXISTS [Project].setJobAsCompleted;   -- 7
    DROP PROCEDURE IF EXISTS [Project].addTransaction;      -- 8
    DROP PROCEDURE IF EXISTS [Project].getTotalCosts;       -- 9
    DROP PROCEDURE IF EXISTS [Project].getTotalLaborTime;   -- 10
    DROP PROCEDURE IF EXISTS [Project].getProcessUpdate;    -- 11
    DROP PROCEDURE IF EXISTS [Project].getJobs;             -- 12
    DROP PROCEDURE IF EXISTS [Project].getCustomers;        -- 13
    DROP PROCEDURE IF EXISTS [Project].deleteJobs;          -- 14
```

```
    DROP PROCEDURE IF EXISTS [Project].setPaintJob;          -- 15
    DROP PROCEDURE IF EXISTS [Project].getCustomersInRange; -- 17


-- ================================================================================
-- CREATE PROCDURES USED IN JAVA
-- ================================================================================


--------------------------------------------------------------------------------
-- 1. Enter a new customer
--------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addCustomer
        @name       VARCHAR(255),
        @address    VARCHAR(255),
        @category   INT

        AS
        BEGIN
            INSERT INTO [Project].Customer VALUES (@name, @address, @category)
        END

    GO


--------------------------------------------------------------------------------
-- 2. Enter a new department
--------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addDepartment
        @deptNo INT
```

```
        AS
        BEGIN
            INSERT INTO [Project].Department VALUES (@deptNo, 0)
        END
    GO


-------------------------------------------------------------------------------
-- 3. Enter a new process-id and its department together with its type and information
-- relevant to the type
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addProcess
        @id         INT,
        @deptNo     INT

        AS
        BEGIN
            -- Update the process table
            INSERT INTO [Project].Process VALUES (@id, @deptNo, 0) -- assumes that the department already exists

            -- Insert the Process into Job Cut, Fit, and Paint table
            -- with other attributes assumed null or 0 where applicable
            INSERT INTO [Project].[P_Cut]   (processID) VALUES (@id)
            INSERT INTO [Project].[P_Fit]   (processID) VALUES (@id)
            INSERT INTO [Project].[P_Paint] (processID) VALUES (@id)
        END
    GO


-------------------------------------------------------------------------------
-- 4. Enter a new assembly with its customer-name, assembly-details, assembly-id,
-- and dateordered and associate it with one or more processes
-------------------------------------------------------------------------------
```

```
    GO
    CREATE PROCEDURE [Project].addAssembly
        @assID          INT,
        @dateOrdered    DATE,
        @details        VARCHAR(255),
        @customerName   VARCHAR(255),
        @processID      INT

        AS
        BEGIN
            -- Add the assembly
            INSERT INTO [Project].Assemblies VALUES (@assID, @dateOrdered, @details, 0)

            -- Add assembly id and customer name to the relation table 'Orders'
            INSERT INTO [Project].Orders     VALUES (@customerName, @assID)

            -- Update into the Relation table manufactures
            INSERT INTO [Project].Manufactures VALUES (@assID, @processID)
        END
    GO


-------------------------------------------------------------------------------
-- 5. Create a new account and associate it with the process, assembly, or department
-- to which it is applicable
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addAccount
        @acctNo             INT,
        @dateEstablished    DATE,
        @assembliesID       INT,
        @processID          INT,
        @deptNo             INT
```

```
        AS
        BEGIN
            -- Add an account to the 'Account' table
            INSERT INTO [Project].Account VALUES (@acctNo, @dateEstablished, @assembliesID, @processID, @deptNo)

        END
    GO


-------------------------------------------------------------------------------
-- 6. Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addJob
        @jobNo          INT,
        @startDate      DATE,
        @assembliesID   INT,
        @processID      INT,
        @deptNo        INT

        AS
        BEGIN
            -- Insert only releveant values into Job table,
            -- intentionally omitting end date and other info attributes
            INSERT INTO [Project].Job (jobNo,   startDate,  assembliesID,  processID,  deptNo)
            VALUES                    (@jobNo, @startDate, @assembliesID, @processID, @deptNo)

            -- Insert the job into Job Cut, Fit, and Paint table
            -- with other attributes assumed null or 0 where applicable
            INSERT INTO [Project].[J_Cut]   (jobNo) VALUES (@jobNo)
            INSERT INTO [Project].[J_Fit]   (jobNo) VALUES (@jobNo)
            INSERT INTO [Project].[J_Paint] (jobNo) VALUES (@jobNo)
        END
```

```
    GO


-------------------------------------------------------------------------------
-- 7. At the completion of a job, enter the date it completed and the information
-- relevant to the type of job
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].setJobAsCompleted
        @jobNo      INT,
        @endDate    DATE,
        @otherInfo  VARCHAR(255)

        AS
        BEGIN
            -- Update existing job's end date and other info
            UPDATE  [Project].Job
            SET [Project].Job.endDate   = @endDate,
                [Project].Job.otherInfo = @otherInfo

            -- only show selected values for a given job number
            WHERE [Project].Job.jobNo   = @jobNo
        END
    GO

-------------------------------------------------------------------------------
-- 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
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].addTransaction
        @transactionNo  INT,
        @cost           REAL,
```

```
        @deptNo         INT,
        @assembliesID   INT,
        @processID      INT,
        @jobNo          INT

        AS
        BEGIN
            -- Update the transaction table
            INSERT INTO [Project].[Transaction] VALUES (@transactionNo, @cost)

            -- Update the Relation table Records to associate with a job
            INSERT INTO [Project].[Records] VALUES (@jobNo, @transactionNo)

            UPDATE  [Project].Assemblies
            SET     [Project].Assemblies.details1 += @cost -- Adds to the existing values in field
            WHERE   [Project].Assemblies.id = @assembliesID

            -- Update the Department Table for the given department
            UPDATE  [Project].Department
            SET     [Project].Department.details2 += @cost -- Adds to the existing values in field
            WHERE   [Project].Department.deptNo = @deptNo

            -- Update the Process table for the given Process
            UPDATE  [Project].Process
            SET     [Project].Process.details3 += @cost -- Adds to the existing values in field
            WHERE   [Project].Process.id = @processID
        END
    GO

-------------------------------------------------------------------------------
-- 9. Retrieve the total cost incurred on an assembly-id
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getTotalCosts
```

```
        @id INT

    AS
        BEGIN
            SELECT details1 AS TOTAL_COST
            FROM [Project].Assemblies

            -- Only show the selected assemblies ID
            WHERE id = @id
        END
    GO

-------------------------------------------------------------------------------
-- 10. Retrieve the total labor time within a department for jobs completed in the
-- department during a given date
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getTotalLaborTime
        @deptNo  INT,
        @endDate DATE

    AS
        BEGIN
            SELECT
                deptNo,
                fit.timeOfLabor + cut.timeOfLabor + paint.timeOfLabor AS totalTimeOfLabor

            FROM [Project].Job job
                -- Join job cut table
                LEFT JOIN [Project].J_Cut cut
                    ON job.jobNo = cut.jobNo

                -- Join job fit table
                LEFT JOIN [Project].J_Fit fit
```

```
                    ON job.jobNo = fit.jobNo

                -- Join job paint table
                LEFT JOIN [Project].J_Paint paint
                    ON job.jobNo = paint.jobNo

            -- Only show the selected deptNo and end date
            WHERE
                    deptNo  = @deptNo
                AND endDate = @endDate
        END
    GO

-------------------------------------------------------------------------------
-- 11. Retrieve the processes through which a given assembly-id has passed so far
-- (in datecommenced order) and the department responsible for each process
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getProcessUpdate
        @assID INT

    AS
    BEGIN
        SELECT
            dateOrdered,
            processID,
            deptNo
        FROM [Project].Process [p]

            -- Relation table between manufactures and assemblies
            LEFT JOIN [Project].Manufactures [m]
                ON p.id = m.processID

            -- Join assemblies table
```

```
            LEFT JOIN [Project].[Assemblies] [a]
                ON m.assembliesID = a.id

        -- Only show the given assemblies ID
        WHERE a.id = @assID

        ORDER BY dateOrdered
    END

    GO

-------------------------------------------------------------------------------
-- 12. Retrieve the jobs (together with their type information and assembly-id)
-- completed during a given date in a given department
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getJobs
            @deptNo  INT,
            @endDate DATE

    AS
        BEGIN
            SELECT
                job.jobNo,
                otherInfo,
                assembliesID

            FROM [Project].Job job
                -- Join job cut table
                LEFT JOIN [Project].J_Cut cut
                    ON job.jobNo = cut.jobNo

                -- Join job fit table
                LEFT JOIN [Project].J_Fit fit
```

```
                    ON job.jobNo = fit.jobNo

                -- Join job paint table
                LEFT JOIN [Project].J_Paint paint
                    ON job.jobNo = paint.jobNo
            WHERE

                -- Conditional upon selected dept and date
                    deptNo  = @deptNo
                AND endDate = @endDate
        END
    GO

-------------------------------------------------------------------------------
-- 13. Retrieve the customers (in name order) whose category is in a given range
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getCustomers
        @min INT,
        @max INT

    AS
        BEGIN
            SELECT [name]
            FROM [Project].Customer

            -- Get the customers with category between min and max range
            WHERE category BETWEEN @min AND @max
            ORDER BY [name]
        END
    GO

-------------------------------------------------------------------------------
-- 14. Delete all cut-jobs whose job-no is in a given range
```

```
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].deleteJobs
        @min INT,
        @max INT

    AS
        BEGIN
            DELETE FROM [Project].J_Cut

            -- Delete cuts between min and max job number range
            -- (with data related to a cut job)
            WHERE
                    jobNo BETWEEN @min AND @max
                AND material IS NOT NULL
        END
    GO

-------------------------------------------------------------------------------
-- 15. Change the color of a given paint job
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].setPaintJob
        @newColor   VARCHAR(255),
        @jobNo      INT

    AS
        BEGIN
            UPDATE  [Project].J_Paint
            SET     [Project].J_Paint.color = @newColor
            WHERE   [Project].J_Paint.jobNo = @jobNo
        END
    GO
```

```
-------------------------------------------------------------------------------
-- 17. Retrieve the customers (in name order) whose category is in a given range
-------------------------------------------------------------------------------

    GO
    CREATE PROCEDURE [Project].getCustomersInRange
        @min INT,
        @max INT

    AS
        BEGIN
            SELECT *
            FROM [Project].Customer

            -- Only within the min and max range
            WHERE category BETWEEN @min AND @max
            ORDER BY [name]
        END
    GO
```

5.2. Java Source Code

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.*;
import java.io.FileWriter;

// ========================================================================
// @class:  DSA 4513
// @asnmt:  Class Project
// @task:   5 (b)
// @author: Daniel Carpenter, ID: 113009743
// @description:
//     Program to implement job-shop accounting system queries
// ========================================================================

public class Carpenter_Daniel_IP_Task5b {

    // Database credentials
    final static String HOSTNAME = "carp9743.database.windows.net";
    final static String DBNAME   = "cs-dsa-4513-sql-db";
    final static String USERNAME = "carp9743";
    final static String PASSWORD = "tacoBout$97315!";

    // Database connection string
    final static String URL =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*
            HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Selected integer that quits application
    final static int QUIT_APPLICATION = 18;

    // Create Template Queries to Execute stored procedures
    final static String QUERY_1  = "EXEC [Project].addCustomer @name = ?,  @address = ?, @category = ?";
    final static String QUERY_2  = "EXEC [Project].addDepartment @deptNo = ?";
    final static String QUERY_3  = "EXEC [Project].addProcess @id = ?, @deptNo = ?";
    final static String QUERY_4  = "EXEC [Project].addAssembly @assID = ?, @dateOrdered = ?, @details = ?, @customerName = ?, @processID
```

```
    final static String QUERY_5  = "EXEC [Project].addAccount @acctNo = ?, @dateEstablished = ?, @assembliesID = ?, @processID = ?, @dept
    final static String QUERY_6  = "EXEC [Project].addJob @jobNo = ?, @startDate = ?, @assembliesID = ?, @processID = ?, @deptNo = ?";
    final static String QUERY_7  = "EXEC [Project].setJobAsCompleted @jobNo = ?, @endDate = ?, @otherInfo = ?";
      final static String QUERY_8  = "EXEC [Project].addTransaction @transactionNo = ?, @cost = ?, @deptNo = ?, @assembliesID = ?, @proc
      final static String QUERY_9  = "{CALL [Project].getTotalCosts(?)}";
      final static String QUERY_10 = "{CALL [Project].getTotalLaborTime(?, ?)}";
      final static String QUERY_11 = "{CALL [Project].getProcessUpdate(?)}";
      final static String QUERY_12 = "{CALL [Project].getJobs(?, ?)}";
      final static String QUERY_13 = "{CALL [Project].getCustomers(?, ?)}";
      final static String QUERY_14 = "EXEC [Project].deleteJobs @min = ?, @max = ?";
      final static String QUERY_15 = "EXEC [Project].setPaintJob @newColor = ?, @jobNo = ?";
      final static String QUERY_17 = "{CALL [Project].getCustomersInRange(?, ?)}";

    // User input prompt
    final static String PROMPT =
            "\nPlease select one of the options below: \n" +
                    "(1) Enter a new customer \n" +
                    "(2) Enter a new department \n" +
                    "(3) Enter a new process-id and its department together with its type and information \n" +
                    "\trelevant to the type\n" +
                    "(4) Enter a new assembly with its customer-name, assembly-details, assembly-id, \n" +
                    "\tand dateordered and associate it with one or more processes\n" +
                    "(5) Create a new account and associate it with the process, assembly, or department \n" +
                    "\tto which it is applicable\n" +
                    "(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced\n" +
                    "(7) At the completion of a job, enter the date it completed and the information \n" +
                    "\trelevant to the type of job \n" +
                    "(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the \n" +
                    "\taffected accounts by adding sup-cost to their current values of details \n" +
                    "(9) Retrieve the total cost incurred on an assembly-id \n" +
                    "(10) Retrieve the total labor time within a department for jobs completed in the \n" +
                    "\tdepartment during a given date\n" +
                    "(11) Retrieve the processes through which a given assembly-id has passed so far \n" +
                    "\t(in datecommenced order) and the department responsible for each process\n" +
                    "(12) Retrieve the jobs (together with their type information and assembly-id) \n" +
                    "\tcompleted during a given date in a given department\n" +
                    "(13) Retrieve the customers (in name order) whose category is in a given range\n" +
                    "(14) Delete all cut-jobs whose job-no is in a given range\n" +
                    "(15) Change the color of a given paint job\n" +
                    "(16) Import: enter new customers from a data file until the file is empty \n" +
                    "(\tthe user must be asked to enter the input file name). \n" +
                    "(17) Export: Retrieve the customers (in name order) whose category is in a given range \n" +
```

```
                "\tand output them to a data file instead of screen (the user must be asked to enter the output file name).\n" +
                "(18) Quit\n";


// Function to read in a csv file and return a concatenated into an insert statement
public static String readCSV(String filename) throws IOException, SQLException  {

   // Number of columns in the customer table (3)
   final int NUM_CUST_COLS = 3;

   // string that will hold the insert statement
    String insertStatement = "INSERT INTO [Project].Customer VALUES (";

   // Create input reader
   BufferedReader input = new BufferedReader(new FileReader(filename));
    String line = "";
    int iterCount = 0; // keep track of iterations
    final int FIRST_ITER = 0;

        // Iterate through each 'row' of the csv
        while ((line = input.readLine()) != null) {

         // IF the first iteration, then do nothing. else concatenate parenthesis
         if (iterCount != FIRST_ITER) {
                insertStatement += ", (";
         } else {
                ++iterCount;
         }


         // Iterate through each 'column' of the csv file
         for (int col = 0; col < NUM_CUST_COLS; ++col) {

                // Add a ' in front of the string vars
                if (col != NUM_CUST_COLS - 1) {
                       insertStatement += "'";
                }

                // return the value of the row for each column index (1 through 3)
                insertStatement += line.split(",")[col];

                // If not at last column, add comma to the string
```

```
                // Add a ' at end of the string vars
                if (col != NUM_CUST_COLS - 1) {
                        insertStatement += "', ";
                }

                // End the values insert
                else {
                        insertStatement += ")";
                }
         }
        }

        // close the input method
        input.close();

        // return the insert statement
        return (insertStatement);
}

public static void main(String[] args) throws SQLException, IOException {

    System.out.println("Update Job-Shop Accounting Database:");

    // CREATE INPUT SCANNER ------------------------------------------------------------------------------
    final Scanner sc = new Scanner(System.in); // Scanner is used to collect thes user input
    String option = ""; // Initialize user option selection as nothing
    while (!option.equals(Integer.toString(QUIT_APPLICATION))) { // As user for options until quit option  is selected
        System.out.println(PROMPT); // Print the available options

        option = sc.next(); // Read in the user option selection


        // BEGIN SWITCH STATEMENTS ==========================================================================
        switch (option) { // Switch between different options

         // (1) Enter a new customer
            case "1":

                // Set the query
                String query = QUERY_1;

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

```
                sc.nextLine();
                String cName = sc.nextLine();

                System.out.println("Please enter customer address in single line:");
                String address = sc.nextLine();

                System.out.println("Please enter integer  customer category between 1 and 10:");
                int category = sc.nextInt();

                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {

                    // Activate stored procedure to enter above data in database
                      // Insert new performer record into database
                    try (
                        final PreparedStatement statement = connection.prepareStatement(query)) {
                        // Populate the query template with the data collected from the user
                        statement.setString(1,   cName);
                        statement.setString(2,   address);
                        statement.setInt(3,      category);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (2) Enter a new department
            case "2":

                // Set the query
                query = QUERY_2;

                System.out.println("Please enter a new department number:");
                int deptNo = sc.nextInt();

                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {

                      // Activate stored procedure to enter above data in database
```

```
                        // Insert new performer record into database
                        try (
                                        final PreparedStatement statement = connection.prepareStatement(query)) {
                                // Populate the query template with the data collected from the user
                                statement.setInt(1,       deptNo);

                                System.out.println("Dispatching the query...");
                                // Actually execute the populated query
                                final int rows_inserted = statement.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                        }
                }
                break;

        // (3) Enter a new process-id and its department together with its type and information
        // relevant to the type
           case "3":

                // Set the query
                query = QUERY_3;

                System.out.println("Please enter a new process id (integer):");
                int processID = sc.nextInt();

                System.out.println("Please enter its existing department number:");
                deptNo = sc.nextInt();

                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {

                        // Activate stored procedure to enter above data in database
                        // Insert new performer record into database
                        try (
                                        final PreparedStatement statement = connection.prepareStatement(query)) {
                                // Populate the query template with the data collected from the user
                                statement.setInt(1,       processID);
                                statement.setInt(2,       deptNo);

                                System.out.println("Dispatching the query...");
                                // Actually execute the populated query
                                final int rows_inserted = statement.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
```

```
                }
            }
            break;

        // (4) Enter a new assembly with its customer-name, assembly-details, assembly-id,
        // and dateordered and associate it with one or more processes
           case "4":

               // Set the query
               query = QUERY_4;

               System.out.println("Please enter a new assembly id (integer):");
               int assID = sc.nextInt();

               System.out.println("Please enter the date commenced in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
                sc.nextLine();
               String dateOrdered = sc.nextLine();

               System.out.println("Please enter the details of the order in text");
               String details = sc.nextLine();

               System.out.println("Please enter the existing customer name associated with the assembly");
               cName = sc.nextLine();

               System.out.println("Please enter the existing process ID (integer) associated with the assembly");
               processID = sc.nextInt();

               // Get a database connection and prepare a query statement
               try (final Connection connection = DriverManager.getConnection(URL)) {

                     // Activate stored procedure to enter above data in database
                     // Insert new performer record into database
                     try (
                                 final PreparedStatement statement = connection.prepareStatement(query)) {
                           // Populate the query template with the data collected from the user
                           statement.setInt(1,       assID);
                           statement.setString(2,    dateOrdered);
                           statement.setString(3,    details);
                           statement.setString(4,    cName);
                           statement.setInt(5,       processID);

                           System.out.println("Dispatching the query...");
```

```
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (5) Create a new account and associate it with the process, assembly, or department
        // to which it is applicable
            case "5":

                // Set the query
                query = QUERY_5;

                System.out.println("Please enter a new account id (integer):");
                int acctNo = sc.nextInt();

                System.out.println("Please enter the date established in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
                sc.nextLine();
                String dateEst = sc.nextLine();

                System.out.println("Please enter the associated assembly id (integer)");
                assID = sc.nextInt();

                System.out.println("Please enter the associated process id (integer)");
                processID = sc.nextInt();

                System.out.println("Please enter the associated department number");
                deptNo = sc.nextInt();

                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {

                    // Activate stored procedure to enter above data in database
                    // Insert new performer record into database
                    try (
                            final PreparedStatement statement = connection.prepareStatement(query)) {
                        // Populate the query template with the data collected from the user
                        statement.setInt(1,       acctNo);
                        statement.setString(2,    dateEst);
                        statement.setInt(3,       assID);
                        statement.setInt(4,       processID);
```

```
                        statement.setInt(5,       deptNo);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
               case "6":

                    // Set the query
                    query = QUERY_6;

                    System.out.println("Please enter a new job number:");
                    int jobNo = sc.nextInt();

                    System.out.println("Please enter the start date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
                    sc.nextLine();
                    String startDate = sc.nextLine();

                    System.out.println("Please enter the associated assembly id (integer)");
                    assID = sc.nextInt();

                    System.out.println("Please enter the associated process id (integer)");
                    processID = sc.nextInt();

                    System.out.println("Please enter the associated department number");
                    deptNo = sc.nextInt();

                    // Get a database connection and prepare a query statement
                    try (final Connection connection = DriverManager.getConnection(URL)) {

                        // Activate stored procedure to enter above data in database
                        // Insert new performer record into database
                        try (
                                final PreparedStatement statement = connection.prepareStatement(query)) {
                            // Populate the query template with the data collected from the user
                            statement.setInt(1,       jobNo);
                            statement.setString(2,    startDate);
```

```
                        statement.setInt(3,       assID);
                        statement.setInt(4,       processID);
                        statement.setInt(5,       deptNo);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (7) At the completion of a job, enter the date it completed and the information
            // relevant to the type of job
               case "7":

                    // Set the query
                    query = QUERY_7;

                    System.out.println("Please enter the existing job number:");
                    jobNo = sc.nextInt();

                    System.out.println("Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
                    sc.nextLine();
                    String endDate = sc.nextLine();

                    System.out.println("Please enter any other info about the job (as text):");
                    String otherInfo = sc.nextLine();

                    // Get a database connection and prepare a query statement
                    try (final Connection connection = DriverManager.getConnection(URL)) {

                        // Activate stored procedure to enter above data in database
                        // Insert new performer record into database
                        try (
                                final PreparedStatement statement = connection.prepareStatement(query)) {
                            // Populate the query template with the data collected from the user
                            statement.setInt(1,       jobNo);
                            statement.setString(2,    endDate);
                            statement.setString(3,    otherInfo);

                            System.out.println("Dispatching the query...");
```

```
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (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
               case "8":

                    // Set the query
                    query = QUERY_8;

                    System.out.println("Please enter the new transaction number:");
                    int transactionNo = sc.nextInt();

                    System.out.println("Please enter the cost of the transaction (as decimal number):");
                    double cost = sc.nextDouble();

                    System.out.println("Please enter the associated assembly id (integer)");
                    assID = sc.nextInt();

                    System.out.println("Please enter the associated process id (integer)");
                    processID = sc.nextInt();

                    System.out.println("Please enter the associated department number");
                    deptNo = sc.nextInt();

                    System.out.println("Please enter the associated job number");
                    jobNo = sc.nextInt();

                    // Get a database connection and prepare a query statement
                    try (final Connection connection = DriverManager.getConnection(URL)) {

                        // Activate stored procedure to enter above data in database
                        // Insert new performer record into database
                        try (
                                final PreparedStatement statement = connection.prepareStatement(query)) {
                            // Populate the query template with the data collected from the user
                            statement.setInt(1,       transactionNo);
                            statement.setDouble(2,    cost);
```

```
                        statement.setInt(3,       deptNo);
                        statement.setInt(4,       assID);
                        statement.setInt(5,       processID);
                        statement.setInt(6,       jobNo);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                }
                break;

            // (9) Retrieve the total cost incurred on an assembly-id
               case "9":

                    // Set the query
                    query = QUERY_9;

                    System.out.println("Please enter the assembly id (integer):");
                    assID = sc.nextInt();

                    // Get a database connection and prepare a query statement
                    try (final Connection connection = DriverManager.getConnection(URL)) {

                          // Prepare a call to the stored procedure
                                CallableStatement cs = connection.prepareCall(query);

                                // Set the assigned value(s) to the procedures input
                                cs.setInt("id", assID);

                                // Run the stored procedure and store values in resultSet
                                System.out.println("Dispatching the query...");
                                ResultSet resultSet = cs.executeQuery();

                                System.out.println("Done.");
                        System.out.println("\nTotal cost incurred on assembly-id: " + assID);

                        // Unpack the tuples returned by the database and print them out to the user
                        while (resultSet.next()) {
                            System.out.println(String.format("%s", resultSet.getString(1)));
                        }
```

```
            }

            break;

        // (10) Retrieve the total labor time within a department for jobs completed in the
        // department during a given date
           case "10":

            // Set the query
            query = QUERY_10;

            System.out.println("Please enter the department number:");
            deptNo = sc.nextInt();

            System.out.println("Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
            sc.nextLine();
            endDate = sc.nextLine();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  CallableStatement cs = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  cs.setInt("deptNo",  deptNo);
                  cs.setString("endDate", endDate);

                  // Run the stored procedure and store values in resultSet
                        System.out.println("Dispatching the query...");
                        ResultSet resultSet = cs.executeQuery();

                        System.out.println("Done.");
                  System.out.println("\nTotal labor time for department: " + deptNo +
                                                   " for date ending on: " + endDate);
                  System.out.println("deptNo   | timeOfLabor");

                  // Unpack the tuples returned by the database and print them out to the user
                  while (resultSet.next()) {
                        System.out.println(String.format("%s      | %s",
                           resultSet.getString(1),
                           resultSet.getString(2)));
```

```
                }
            }

            break;


        // (11) Retrieve the processes through which a given assembly-id has passed so far
        // (in datecommenced order) and the department responsible for each process
           case "11":

            // Set the query
            query = QUERY_11;

            System.out.println("Please enter the assembly id (integer):");
            assID = sc.nextInt();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  CallableStatement cs = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  cs.setInt("assID",   assID);

                  // Run the stored procedure and store values in resultSet
                        System.out.println("Dispatching the query...");
                        ResultSet resultSet = cs.executeQuery();

                        System.out.println("Done.");
                  System.out.println("\nProcess for assembly-id: " + assID +
                              ", and its departement number; Sorted by date commenced.");
                  System.out.println("dateOrdered    | processID| deptNo");

                  // Unpack the tuples returned by the database and print them out to the user
                  while (resultSet.next()) {
                        System.out.println(String.format("%s    | %s   | %s",
                                    resultSet.getString(1),
                                    resultSet.getString(2),
                                    resultSet.getString(3)));
                  }
            }
```

```
            break;

        // (12) Retrieve the jobs (together with their type information and assembly-id)
        // completed during a given date in a given department
           case "12":

            // Set the query
            query = QUERY_12;

            System.out.println("Please enter the department number:");
            deptNo = sc.nextInt();

            System.out.println("Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:");
            sc.nextLine();
            endDate = sc.nextLine();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  CallableStatement cs = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  cs.setInt("deptNo",  deptNo);
                  cs.setString("endDate", endDate);

                  // Run the stored procedure and store values in resultSet
                        System.out.println("Dispatching the query...");
                        ResultSet resultSet = cs.executeQuery();

                        System.out.println("Done.");
                  System.out.println("\nJobs from department " + deptNo +
                              " completed on: " + endDate);
                  System.out.println("jobNo    | otherInfo| assembliesID ");

                  // Unpack the tuples returned by the database and print them out to the user
                  while (resultSet.next()) {
                        System.out.println(String.format("%s      | %s | %s",
                                    resultSet.getString(1),
                                    resultSet.getString(2),
                                    resultSet.getString(3)));
```

```
                }
            }

            break;

        // (13) Retrieve the customers (in name order) whose category is in a given range
           case "13":

            // Set the query
            query = QUERY_13;

            System.out.println("Please enter MIN category number (integer from 1 - 10, inclusive):");
            int min = sc.nextInt();

            System.out.println("Please enter MAX category number (integer from 1 - 10, inclusive):");
            int max = sc.nextInt();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  CallableStatement cs = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  cs.setInt("min", min);
                  cs.setInt("max", max);

                  // Run the stored procedure and store values in resultSet
                        System.out.println("Dispatching the query...");
                        ResultSet resultSet = cs.executeQuery();

                        System.out.println("Done.");
                  System.out.println("\nCustomers with category from " + min + " to " + max);
                  System.out.println("name"); //     | otherInfo| assembliesID ");

                  // Unpack the tuples returned by the database and print them out to the user
                  while (resultSet.next()) {
                        System.out.println(String.format("%s", //       | %s | %s",
                                    resultSet.getString(1)));
                  }
            }
```

```
            break;

        // (14) Delete all cut-jobs whose job-no is in a given range
           case "14":

               // Set the query
               query = QUERY_14;

               System.out.println("Please enter MIN category number (integer from 1 - 10, inclusive):");
               min = sc.nextInt();

               System.out.println("Please enter MAX category number (integer from 1 - 10, inclusive):");
               max = sc.nextInt();

               // Get a database connection and prepare a query statement
               try (final Connection connection = DriverManager.getConnection(URL)) {

                     // Prepare a call to the stored procedure
                     PreparedStatement ps = connection.prepareCall(query);

                     // Set the assigned value(s) to the procedures input
                     ps.setInt(1, min);
                     ps.setInt(2, max);



                     System.out.println("Dispatching the query...");
                     // Actually execute the populated query
                     final int rows_deleted = ps.executeUpdate();
                     System.out.println(String.format("Done. %d rows deleted.", rows_deleted) +
                                                " from " + min + " to " + max);
               }

               break;

        // (15) Change the color of a given paint job
           case "15":

               // Set the query
               query = QUERY_15;

               System.out.println("Please enter the new color:");
```

```
            sc.nextLine();
            String newColor = sc.nextLine();

            System.out.println("Please enter the job number associated:");
            jobNo = sc.nextInt();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  PreparedStatement ps = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  ps.setString(1, newColor);
                  ps.setInt(2, jobNo);



                  System.out.println("Dispatching the query...");
                  // Actually execute the populated query
                  final int rows_changed = ps.executeUpdate();
                  System.out.println(String.format("Done. %d rows changed.", rows_changed) +
                              " for job number: " + jobNo);
            }

            break;

      // (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).

         case "16":

            System.out.println("Please enter the location and name of a CSV file with customer data:" +
                                          "\n>> PLEASE DO NOT INCLUDE COMMAS EXCEPT FOR THE DELIMITER <<");
            sc.nextLine();
            String filename = sc.nextLine();

            // create insert statement with values from csv file
            query = readCSV(filename);


            // Get a database connection and prepare a query statement
```

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

                  // Prepare a call to the stored procedure
                  PreparedStatement ps = connection.prepareCall(query);

                  System.out.println("Dispatching the query...");
                  // Actually execute the populated query
                  final int rows_inserted = ps.executeUpdate();
                  System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
            }
            break;


      // (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).

         case "17":

            // Set the query
            query = QUERY_17;

            System.out.println("Please enter MIN category number (integer from 1 - 10, inclusive):");
            min = sc.nextInt();

            System.out.println("Please enter MAX category number (integer from 1 - 10, inclusive):");
            max = sc.nextInt();

            System.out.println("Please enter the file output name:");
            sc.nextLine();
            filename = sc.nextLine();

            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                  // Prepare a call to the stored procedure
                  CallableStatement cs = connection.prepareCall(query);

                  // Set the assigned value(s) to the procedures input
                  cs.setInt("min", min);
                  cs.setInt("max", max);

                  // Run the stored procedure and store values in resultSet
```

```
                        System.out.println("Dispatching the query...");
                    ResultSet resultSet = cs.executeQuery();

                    try {
                        FileWriter myWriter = new FileWriter(filename + ".csv");
                        myWriter.write("name,address,category\n");

                        // Unpack the tuples returned by the database and print them out to the user
                        while (resultSet.next()) {
                          myWriter.write(String.format("%s,%s,%s\n",
                                  resultSet.getString(1),
                                  resultSet.getString(2),
                                  resultSet.getString(3)));
                        }

                        // close the writer
                        myWriter.close();

                      } catch (IOException e) {
                        System.out.println("Error with file name.");
                        e.printStackTrace();
                      }
                }

                    System.out.println("Done. File Location here:");
                    System.out.println(filename + ".csv");


                    break;

             // (18) Quit
                case "18":
                    System.out.println("Finished! Your work here is done.");
            }
        }

        sc.close(); // Close the scanner before exiting the application
    }
}
```

5.2. Continued: Java Program Screenshot of Successful Compilation

Carpenter_Daniel_IP_Task5b.java

```
1  import java.sql.CallableStatement;
10
11 // ============================================================================
12 // @class:  DSA 4513
13 // @asnmt:  Class Project
14 // @task:   5 (b)
15 // @author: Daniel Carpenter, ID: 113009743
16 // @description:
17 //     Program to implement job-shop accounting system queries
18 // ============================================================================
19
20 public class Carpenter_Daniel_IP_Task5b {
21
```

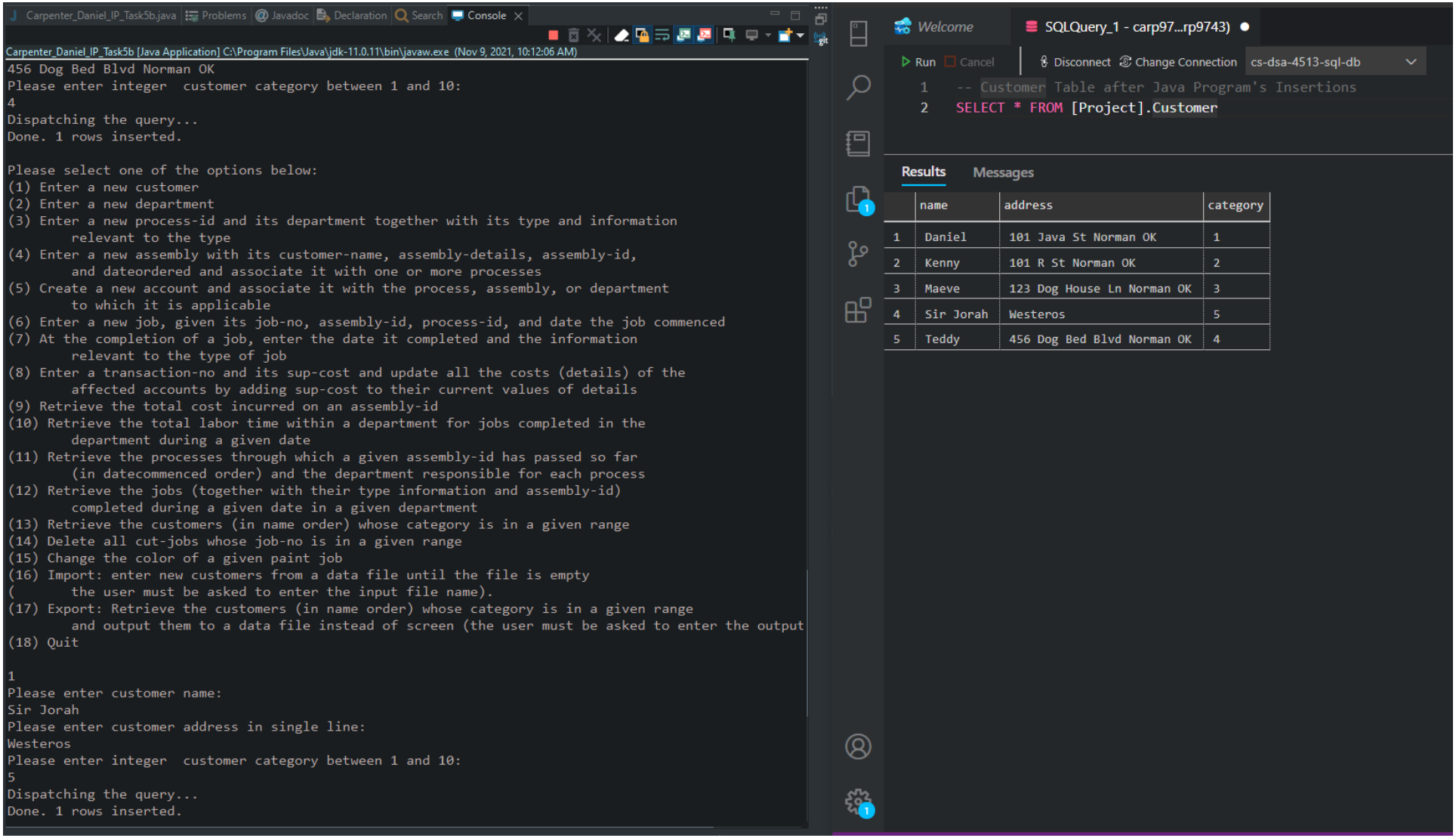
Problems | Javadoc | Declaration | Search | Console

Carpenter_Daniel_IP_Task5b [Java Application] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe (Nov 9, 2021, 10:12:06 AM)

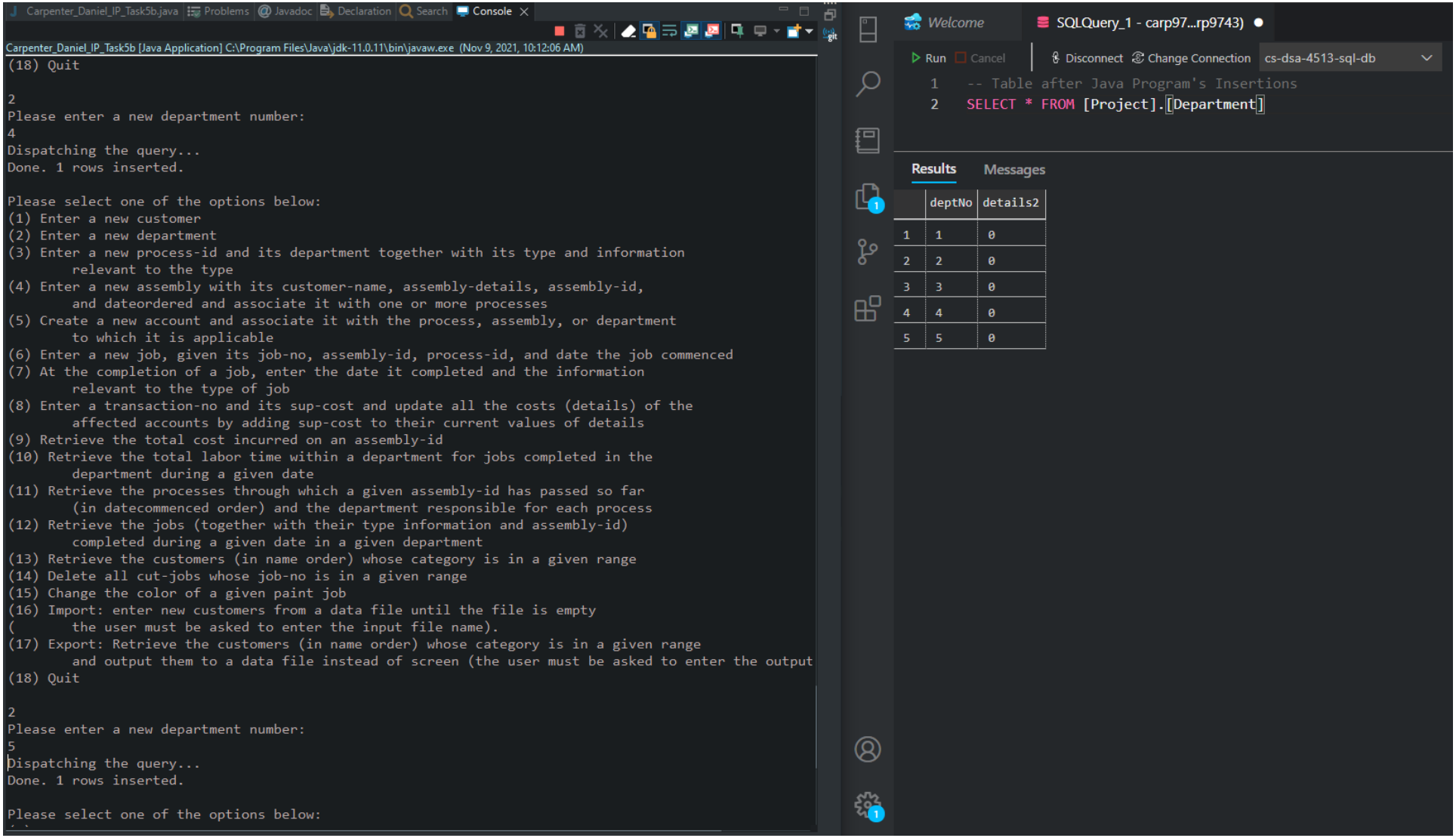
```
Update Job-Shop Accounting Database:

Please select one of the options below:
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new process-id and its department together with its type and information
       relevant to the type
(4) Enter a new assembly with its customer-name, assembly-details, assembly-id,
       and dateordered and associate it with one or more processes
(5) Create a new account and associate it with the process, assembly, or department
       to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information
       relevant to the type of job
(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
(9) Retrieve the total cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the
       department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far
       (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id)
       completed during a given date in a given department
(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
(15) Change the color of a given paint job
(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
```

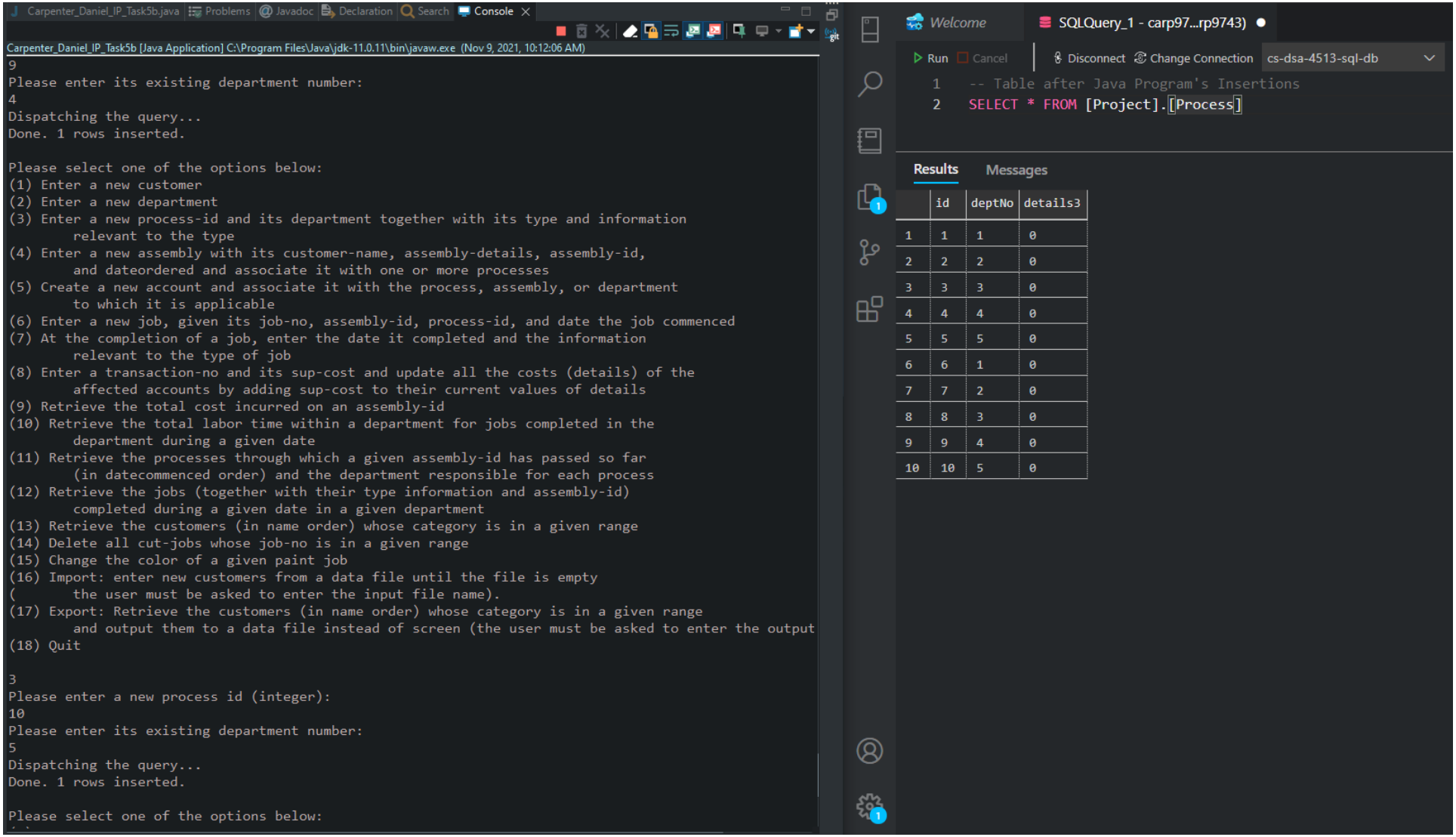
6.1. Screenshots showing the testing of Query 1



6.2. Screenshots showing the testing of Query 2



6.3. Screenshots showing the testing of Query 3



6.3. Screenshots showing the testing of Query 3 *(Continued)*

```
1   SELECT * FROM [Project].[P_Cut]
2   SELECT * FROM [Project].[P_Fit]
3   SELECT * FROM [Project].[P_Paint]
```

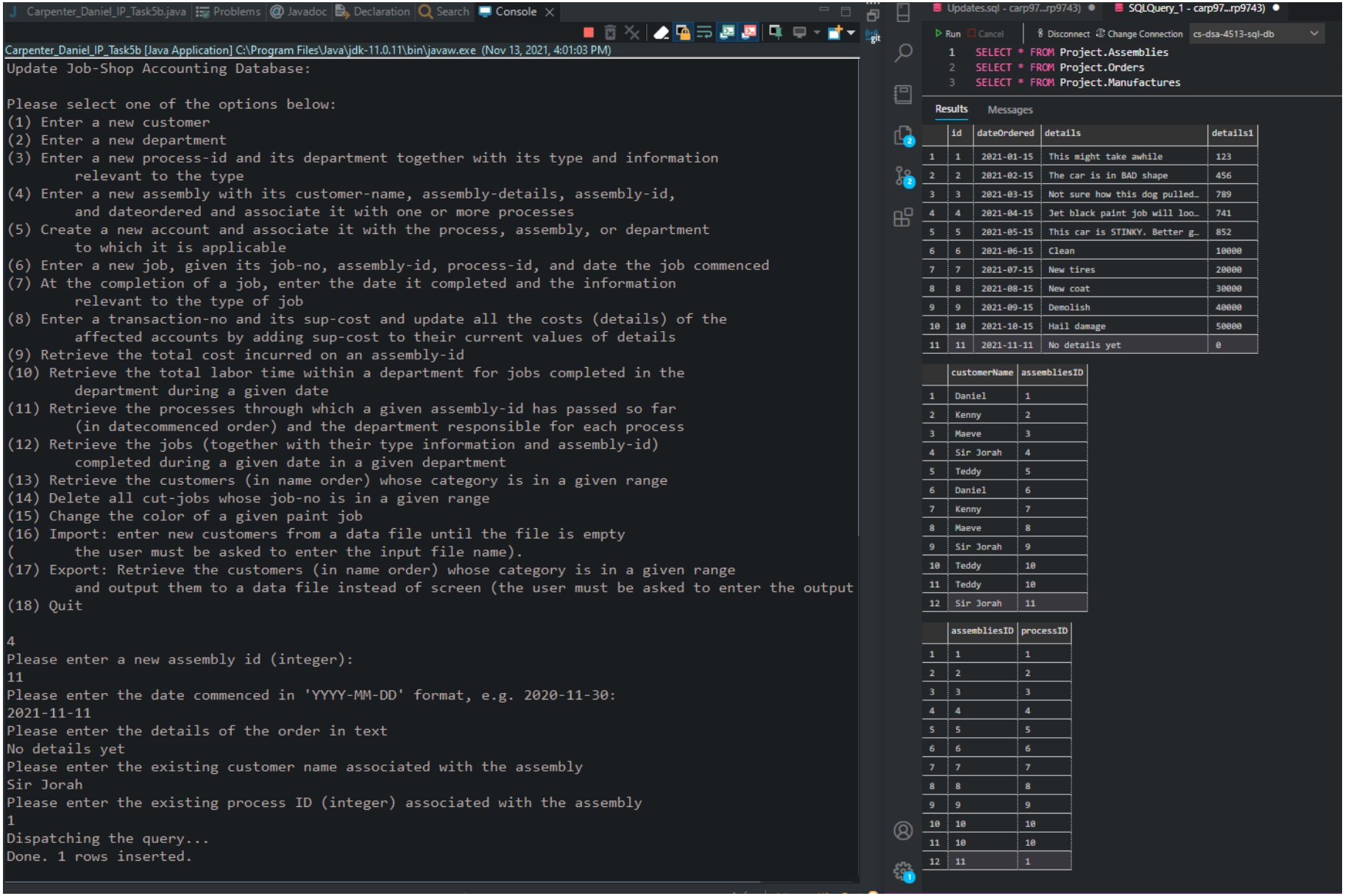
Results Messages

	processID	cutType	machineType
1	1	NULL	NULL
2	2	NULL	NULL
3	3	NULL	NULL
4	4	NULL	NULL
5	5	NULL	NULL
6	6	NULL	NULL
7	7	NULL	NULL
8	8	NULL	NULL
9	9	NULL	NULL
10	10	NULL	NULL

	processID	type
1	1	NULL
2	2	NULL
3	3	NULL
4	4	NULL
5	5	NULL
6	6	NULL
7	7	NULL
8	8	NULL
9	9	NULL
10	10	NULL

	processID	type	method
1	1	NULL	NULL
2	2	NULL	NULL
3	3	NULL	NULL
4	4	NULL	NULL
5	5	NULL	NULL
6	6	NULL	NULL
7	7	NULL	NULL
8	8	NULL	NULL
9	9	NULL	NULL
10	10	NULL	NULL

6.4. Screenshots showing the testing of Query 4



```
Carpenter_Daniel_IP_Task5b [Java Application] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe  (Nov 13, 2021, 4:01:03 PM)
Update Job-Shop Accounting Database:

Please select one of the options below:
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new process-id and its department together with its type and information
        relevant to the type
(4) Enter a new assembly with its customer-name, assembly-details, assembly-id,
        and dateordered and associate it with one or more processes
(5) Create a new account and associate it with the process, assembly, or department
        to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information
        relevant to the type of job
(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
(9) Retrieve the total cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the
        department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far
        (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id)
        completed during a given date in a given department
(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
(15) Change the color of a given paint job
(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
(18) Quit

4
Please enter a new assembly id (integer):
11
Please enter the date commenced in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-11
Please enter the details of the order in text
No details yet
Please enter the existing customer name associated with the assembly
Sir Jorah
Please enter the existing process ID (integer) associated with the assembly
1
Dispatching the query...
Done. 1 rows inserted.
```

```
SELECT * FROM Project.Assemblies
SELECT * FROM Project.Orders
SELECT * FROM Project.Manufactures
```

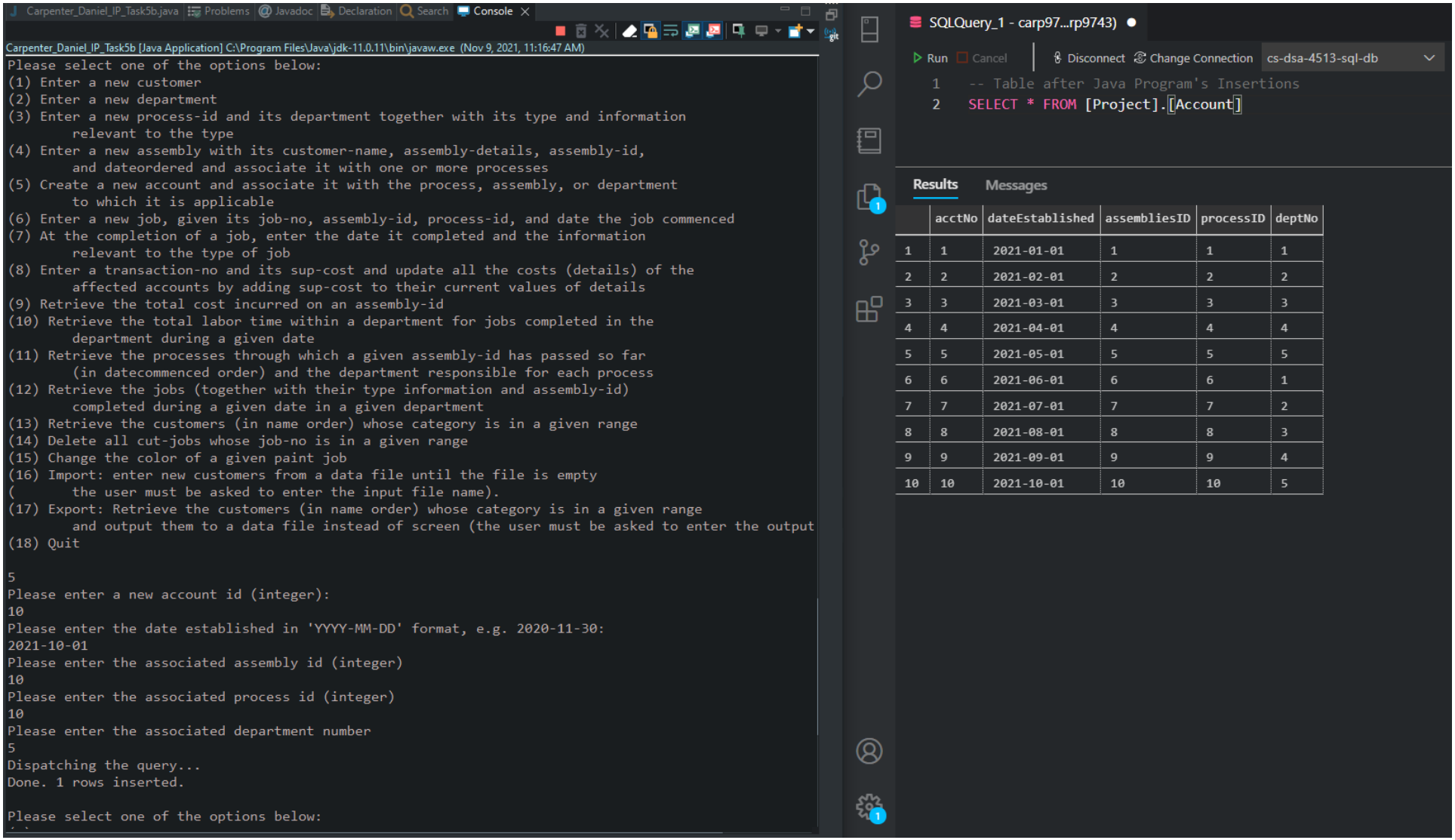
Results

	id	dateOrdered	details	details1
1	1	2021-01-15	This might take awhile	123
2	2	2021-02-15	The car is in BAD shape	456
3	3	2021-03-15	Not sure how this dog pulled…	789
4	4	2021-04-15	Jet black paint job will loo…	741
5	5	2021-05-15	This car is STINKY. Better g…	852
6	6	2021-06-15	Clean	10000
7	7	2021-07-15	New tires	20000
8	8	2021-08-15	New coat	30000
9	9	2021-09-15	Demolish	40000
10	10	2021-10-15	Hail damage	50000
11	11	2021-11-11	No details yet	0

	customerName	assembliesID
1	Daniel	1
2	Kenny	2
3	Maeve	3
4	Sir Jorah	4
5	Teddy	5
6	Daniel	6
7	Kenny	7
8	Maeve	8
9	Sir Jorah	9
10	Teddy	10
11	Teddy	10
12	Sir Jorah	11

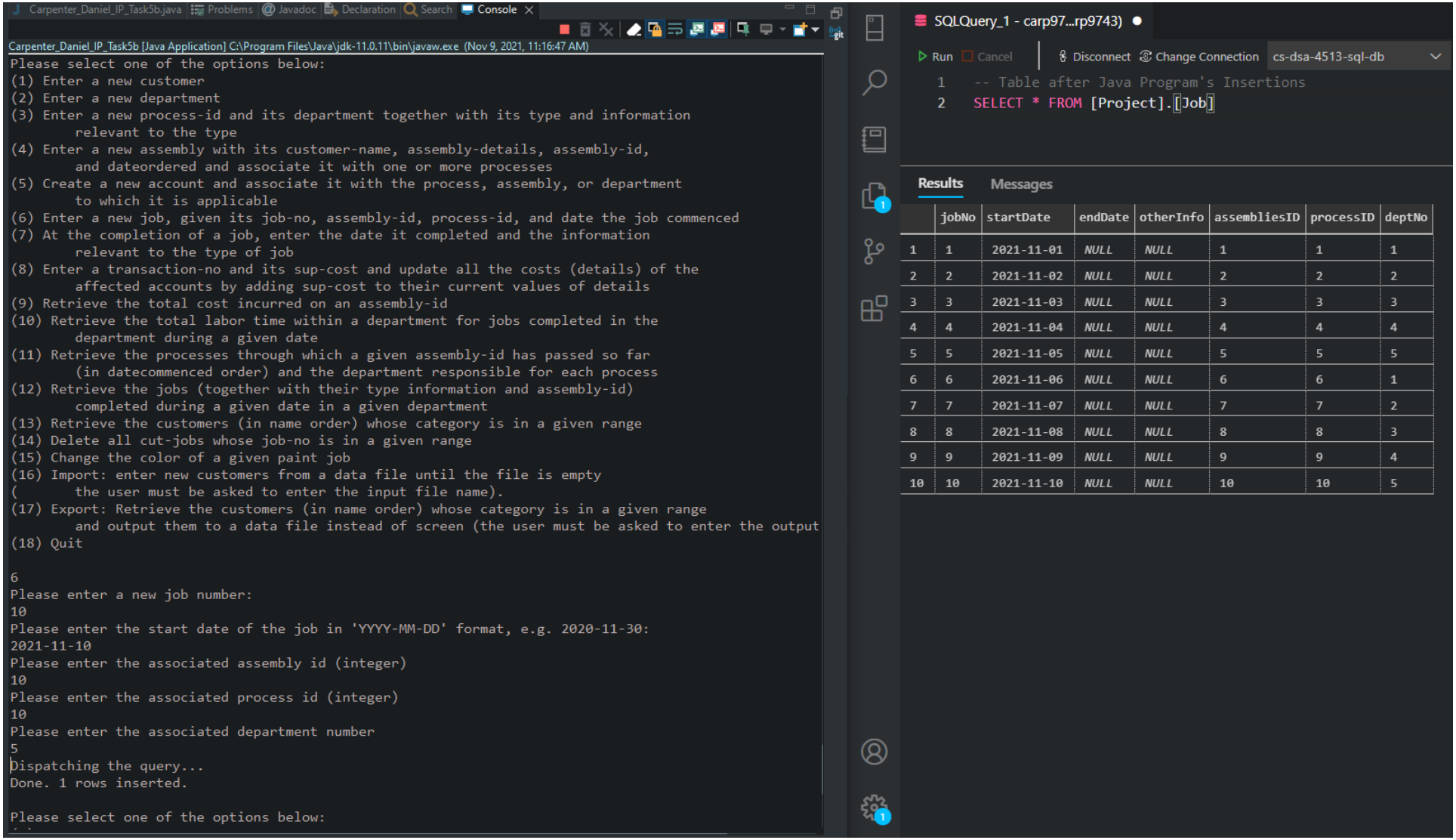
	assembliesID	processID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	10	10
12	11	1

6.5. Screenshots showing the testing of Query 5



	acctNo	dateEstablished	assembliesID	processID	deptNo
1	1	2021-01-01	1	1	1
2	2	2021-02-01	2	2	2
3	3	2021-03-01	3	3	3
4	4	2021-04-01	4	4	4
5	5	2021-05-01	5	5	5
6	6	2021-06-01	6	6	1
7	7	2021-07-01	7	7	2
8	8	2021-08-01	8	8	3
9	9	2021-09-01	9	9	4
10	10	2021-10-01	10	10	5

6.6. Screenshots showing the testing of Query 6



6.6. Screenshots showing the testing of Query 6 *(Continued)*

```
1  SELECT * FROM [Project].[J_Cut]
2  SELECT * FROM [Project].[J_Fit]
3  SELECT * FROM [Project].[J_Paint]
```

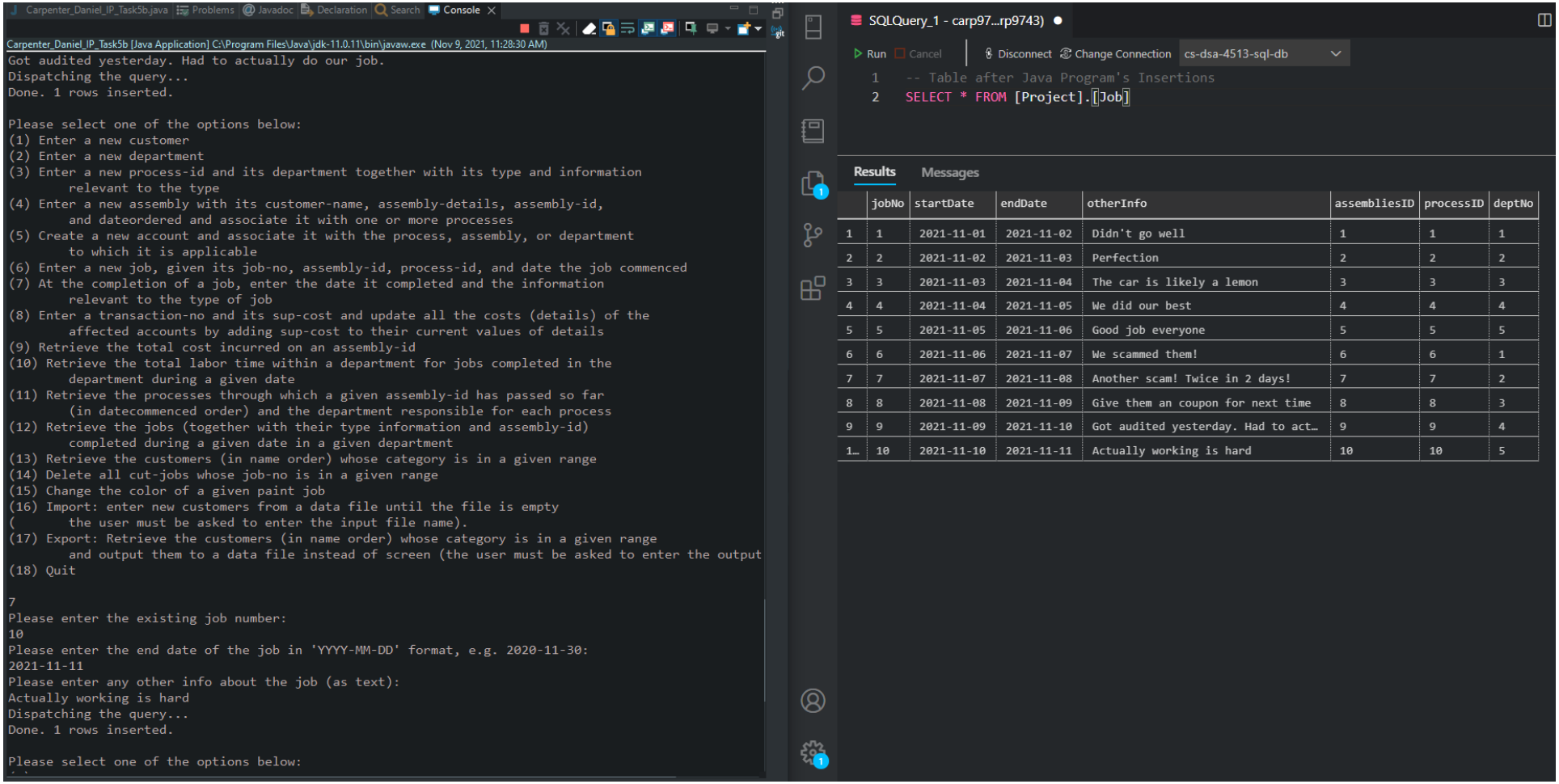
Results Messages

	jobNo	material	timeOfLabor	type	amount
1	1	NULL	0	NULL	0
2	2	NULL	0	NULL	0
3	3	NULL	0	NULL	0
4	4	NULL	0	NULL	0
5	5	NULL	0	NULL	0
6	6	NULL	0	NULL	0
7	7	NULL	0	NULL	0
8	8	NULL	0	NULL	0
9	9	NULL	0	NULL	0
10	10	NULL	0	NULL	0

	jobNo	timeOfLabor
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0
9	9	0
10	10	0

	jobNo	color	volume	timeOfLabor
1	1	NULL	0	0
2	2	NULL	0	0
3	3	NULL	0	0
4	4	NULL	0	0
5	5	NULL	0	0
6	6	NULL	0	0
7	7	NULL	0	0
8	8	NULL	0	0
9	9	NULL	0	0
10	10	NULL	0	0

6.7. Screenshots showing the testing of Query 7



```
Carpenter_Daniel_IP_Task5b [Java Application] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe  (Nov 9, 2021, 11:28:30 AM)
Got audited yesterday. Had to actually do our job.
Dispatching the query...
Done. 1 rows inserted.

Please select one of the options below:
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new process-id and its department together with its type and information
       relevant to the type
(4) Enter a new assembly with its customer-name, assembly-details, assembly-id,
       and dateordered and associate it with one or more processes
(5) Create a new account and associate it with the process, assembly, or department
       to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information
       relevant to the type of job
(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
(9) Retrieve the total cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the
       department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far
       (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id)
       completed during a given date in a given department
(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
(15) Change the color of a given paint job
(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
(18) Quit

7
Please enter the existing job number:
10
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-11
Please enter any other info about the job (as text):
Actually working is hard
Dispatching the query...
Done. 1 rows inserted.

Please select one of the options below:
```

SQLQuery_1 - carp97...rp9743)

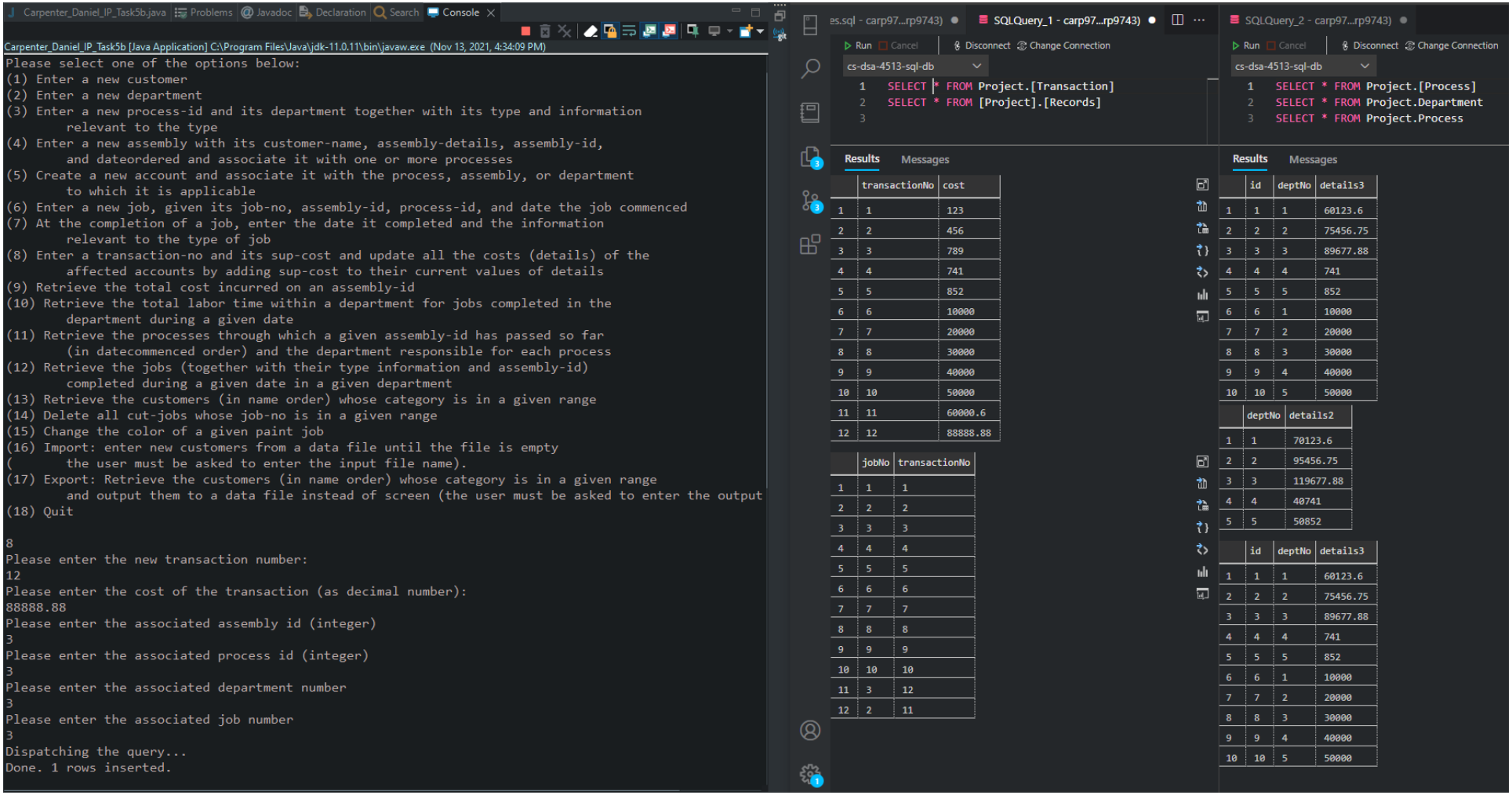
Run | Cancel | Disconnect | Change Connection | cs-dsa-4513-sql-db

```
1   -- Table after Java Program's Insertions
2   SELECT * FROM [Project].[Job]
```

Results | Messages

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 days!	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next time	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had to act…	9	9	4
1…	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

6.8. Screenshots showing the testing of Query 8



6.9. Screenshots showing the testing of Query 9

Example 1

```
(18) Quit

9
Please enter the assembly id (integer):
1
Dispatching the query...
Done.

Total cost incurred on assembly-id: 1
123.0
```

Example 2

```
(18) Quit

9
Please enter the assembly id (integer):
2
Dispatching the query...
Done.

Total cost incurred on assembly-id: 2
456.0
```

Example 3

```
(18) Quit

9
Please enter the assembly id (integer):
3
Dispatching the query...
Done.

Total cost incurred on assembly-id: 3
789.0
```

6.10. Screenshots showing the testing of Query 10

Please note that these values are 0 since no values of `timeOfLabor` have been supplied in the database yet.

Example 1

```
(18) Quit

10
Please enter the department number:
1
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-02
Dispatching the query...
Done.

Total labor time for department: 1 for date ending on: 2021-11-02
deptNo    | timeOfLabor
1         | 0
```

Example 2

```
(18) Quit

10
Please enter the department number:
2
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-03
Dispatching the query...
Done.

Total labor time for department: 2 for date ending on: 2021-11-03
deptNo    | timeOfLabor
2         | 0
```

Example 3

```
(18) Quit

10
Please enter the department number:
3
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-04
Dispatching the query...
Done.

Total labor time for department: 3 for date ending on: 2021-11-04
deptNo    | timeOfLabor
3         | 0
```

6.11. Screenshots showing the testing of Query 11

Example 1

```
(18) Quit

11
Please enter the assembly id (integer):
1
Dispatching the query...
Done.

Process for assembly-id: 1, and its departement number; Sorted by date commenced.
dateOrdered       | processID   | deptNo
2021-01-15      | 1     | 1
```

Example 2

```
(18) Quit

11
Please enter the assembly id (integer):
2
Dispatching the query...
Done.

Process for assembly-id: 2, and its departement number; Sorted by date commenced.
dateOrdered       | processID   | deptNo
2021-02-15      | 2     | 2
```

Example 3

```
(18) Quit

11
Please enter the assembly id (integer):
3
Dispatching the query...
Done.

Process for assembly-id: 3, and its departement number; Sorted by date commenced.
dateOrdered       | processID   | deptNo
2021-03-15      | 3     | 3
```

6.12. Screenshots showing the testing of Query 12

Example 1

```
(18) Quit

12
Please enter the department number:
1
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-02
Dispatching the query...
Done.

Jobs from department 1 completed on: 2021-11-02
jobNo     | otherInfo    | assembliesID
1         | Didn't go well      | 1
```

Example 2

```
(18) Quit

12
Please enter the department number:
2
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-03
Dispatching the query...
Done.

Jobs from department 2 completed on: 2021-11-03
jobNo     | otherInfo    | assembliesID
2         | Perfection   | 2
```

Example 3

```
(18) Quit

12
Please enter the department number:
4
Please enter the end date of the job in 'YYYY-MM-DD' format, e.g. 2020-11-30:
2021-11-10
Dispatching the query...
Done.

Jobs from department 4 completed on: 2021-11-10
jobNo     | otherInfo    | assembliesID
9         | Got audited yesterday. Had to actually do our job.  | 9
```

6.13. Screenshots showing the testing of Query 13

Example 1

```
(18) Quit

13
Please enter MIN category number (integer from 1 - 10, inclusive):
1
Please enter MAX category number (integer from 1 - 10, inclusive):
3
Dispatching the query...
Done.

Customers with category from 1 to 3
name
Daniel
Kenny
Maeve
```

Example 2

```
(18) Quit

13
Please enter MIN category number (integer from 1 - 10, inclusive):
3
Please enter MAX category number (integer from 1 - 10, inclusive):
5
Dispatching the query...
Done.

Customers with category from 3 to 5
name
Maeve
Sir Jorah
Teddy
```

Example 3

```
(18) Quit

13
Please enter MIN category number (integer from 1 - 10, inclusive):
1
Please enter MAX category number (integer from 1 - 10, inclusive):
10
Dispatching the query...
Done.

Customers with category from 1 to 10
name
Daniel
Kenny
Maeve
Sir Jorah
Teddy
```

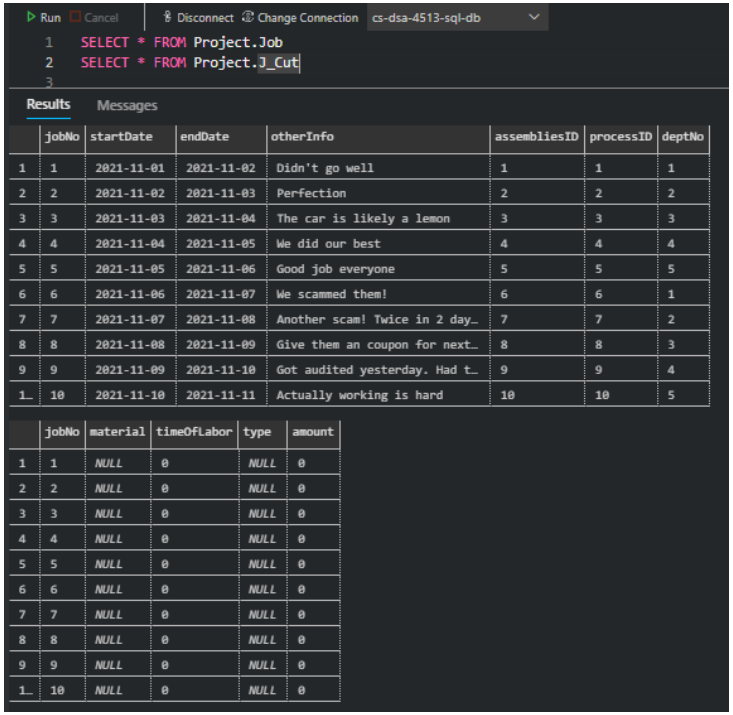
6.14. Screenshots showing the testing of Query 14

None deleted since no values of J_Cut have been supplied that specify if the job will be cut. I created the procedure this way to avoid deleting rows when not needed. If another procedure were to specify that the job was a cut job, then it would make more sense.

Example 1

```
(18) Quit

14
Please enter MIN category number (integer from 1 - 10, inclusive):
1
Please enter MAX category number (integer from 1 - 10, inclusive):
3
Dispatching the query...
Done. 0 rows deleted. from 1 to 3
```



Run Cancel | Disconnect Change Connection cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.Job
2  SELECT * FROM Project.J_Cut
3
```

Results Messages

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
1…	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

	jobNo	material	timeOfLabor	type	amount
1	1	NULL	0	NULL	0
2	2	NULL	0	NULL	0
3	3	NULL	0	NULL	0
4	4	NULL	0	NULL	0
5	5	NULL	0	NULL	0
6	6	NULL	0	NULL	0
7	7	NULL	0	NULL	0
8	8	NULL	0	NULL	0
9	9	NULL	0	NULL	0
1…	10	NULL	0	NULL	0

6.14. Screenshots showing the testing of Query 14 *(Continued)*

Example 2

```
(18) Quit

14
Please enter MIN category number (integer from 1 - 10, inclusive):
5
Please enter MAX category number (integer from 1 - 10, inclusive):
10
Dispatching the query...
Done. 0 rows deleted. from 5 to 10
```

Run Cancel | Disconnect Change Connection cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.Job
2  SELECT * FROM Project.J_Cut
3
```

Results Messages

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
1…	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

	jobNo	material	timeOfLabor	type	amount
1	1	NULL	0	NULL	0
2	2	NULL	0	NULL	0
3	3	NULL	0	NULL	0
4	4	NULL	0	NULL	0
5	5	NULL	0	NULL	0
6	6	NULL	0	NULL	0
7	7	NULL	0	NULL	0
8	8	NULL	0	NULL	0
9	9	NULL	0	NULL	0
1…	10	NULL	0	NULL	0

6.14. Screenshots showing the testing of Query 14 *(Continued)*

Example 3

```
(18) Quit

14
Please enter MIN category number (integer from 1 - 10, inclusive):
1
Please enter MAX category number (integer from 1 - 10, inclusive):
10
Dispatching the query...
Done. 0 rows deleted. from 1 to 10
```

Run | Cancel | Disconnect | Change Connection | cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.Job
2  SELECT * FROM Project.J_Cut
3
```

Results | Messages

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
1…	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

	jobNo	material	timeOfLabor	type	amount
1	1	NULL	0	NULL	0
2	2	NULL	0	NULL	0
3	3	NULL	0	NULL	0
4	4	NULL	0	NULL	0
5	5	NULL	0	NULL	0
6	6	NULL	0	NULL	0
7	7	NULL	0	NULL	0
8	8	NULL	0	NULL	0
9	9	NULL	0	NULL	0
1…	10	NULL	0	NULL	0

6.15. Screenshots showing the testing of Query 15

Example 1

```
(18) Quit

15
Please enter the new color:
Black
Please enter the job number associated:
1
Dispatching the query...
Done. 1 rows changed. for job number: 1
```

Run | Cancel | Disconnect | Change Connection | cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.J_Paint
2  SELECT * FROM Project.Job
3
```

Results | Messages

Results grid

	jobNo	color	volume	timeOfLabor
1	1	Black	0	0
2	2	*NULL*	0	0
3	3	*NULL*	0	0
4	4	*NULL*	0	0
5	5	*NULL*	0	0
6	6	*NULL*	0	0
7	7	*NULL*	0	0
8	8	*NULL*	0	0
9	9	*NULL*	0	0
10	10	*NULL*	0	0

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
10	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

6.15. Screenshots showing the testing of Query 15 *(Continued)*

Example 2

```
(18) Quit

15
Please enter the new color:
Red
Please enter the job number associated:
2
Dispatching the query...
Done. 1 rows changed. for job number: 2
```

Run | Cancel | Disconnect | Change Connection | cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.J_Paint
2  SELECT * FROM Project.Job
3
```

Results | Messages

	jobNo	color	volume	timeOfLabor
1	1	Black	0	0
2	2	Red	0	0
3	3	*NULL*	0	0
4	4	*NULL*	0	0
5	5	*NULL*	0	0
6	6	*NULL*	0	0
7	7	*NULL*	0	0
8	8	*NULL*	0	0
9	9	*NULL*	0	0
10	10	*NULL*	0	0

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
10	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

6.15. Screenshots showing the testing of Query 15 *(Continued)*

Example 3

```
(18) Quit

15
Please enter the new color:
Hot-Rod Red
Please enter the job number associated:
3
Dispatching the query...
Done. 1 rows changed. for job number: 3
```

Run Cancel | Disconnect Change Connection cs-dsa-4513-sql-db

```
1  SELECT * FROM Project.J_Paint
2  SELECT * FROM Project.Job
```

Results Messages

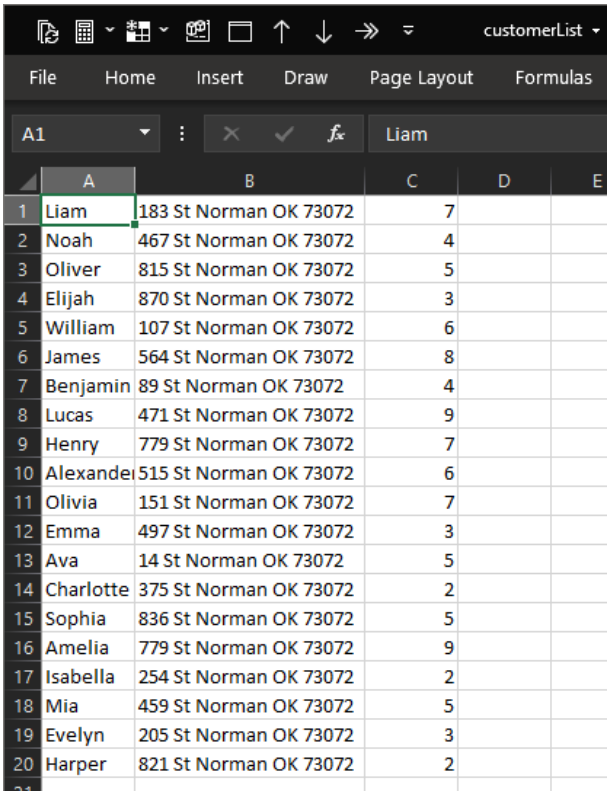
	jobNo	color	volume	timeOfLabor
1	1	Black	0	0
2	2	Red	0	0
3	3	Hot-Rod Red	0	0
4	4	*NULL*	0	0
5	5	*NULL*	0	0
6	6	*NULL*	0	0
7	7	*NULL*	0	0
8	8	*NULL*	0	0
9	9	*NULL*	0	0
10	10	*NULL*	0	0

	jobNo	startDate	endDate	otherInfo	assembliesID	processID	deptNo
1	1	2021-11-01	2021-11-02	Didn't go well	1	1	1
2	2	2021-11-02	2021-11-03	Perfection	2	2	2
3	3	2021-11-03	2021-11-04	The car is likely a lemon	3	3	3
4	4	2021-11-04	2021-11-05	We did our best	4	4	4
5	5	2021-11-05	2021-11-06	Good job everyone	5	5	5
6	6	2021-11-06	2021-11-07	We scammed them!	6	6	1
7	7	2021-11-07	2021-11-08	Another scam! Twice in 2 day…	7	7	2
8	8	2021-11-08	2021-11-09	Give them an coupon for next…	8	8	3
9	9	2021-11-09	2021-11-10	Got audited yesterday. Had t…	9	9	4
10	10	2021-11-10	2021-11-11	Actually working is hard	10	10	5

6.16. Screenshots Showing the Testing of the Import and Export Options

Import Option (16):

customerList.csv contents:



customerList

File Home Insert Draw Page Layout Formulas

A1 Liam

	A	B	C	D	E
1	Liam	183 St Norman OK 73072	7		
2	Noah	467 St Norman OK 73072	4		
3	Oliver	815 St Norman OK 73072	5		
4	Elijah	870 St Norman OK 73072	3		
5	William	107 St Norman OK 73072	6		
6	James	564 St Norman OK 73072	8		
7	Benjamin	89 St Norman OK 73072	4		
8	Lucas	471 St Norman OK 73072	9		
9	Henry	779 St Norman OK 73072	7		
10	Alexander	515 St Norman OK 73072	6		
11	Olivia	151 St Norman OK 73072	7		
12	Emma	497 St Norman OK 73072	3		
13	Ava	14 St Norman OK 73072	5		
14	Charlotte	375 St Norman OK 73072	2		
15	Sophia	836 St Norman OK 73072	5		
16	Amelia	779 St Norman OK 73072	9		
17	Isabella	254 St Norman OK 73072	2		
18	Mia	459 St Norman OK 73072	5		
19	Evelyn	205 St Norman OK 73072	3		
20	Harper	821 St Norman OK 73072	2		

Input file name and path to upload to database:

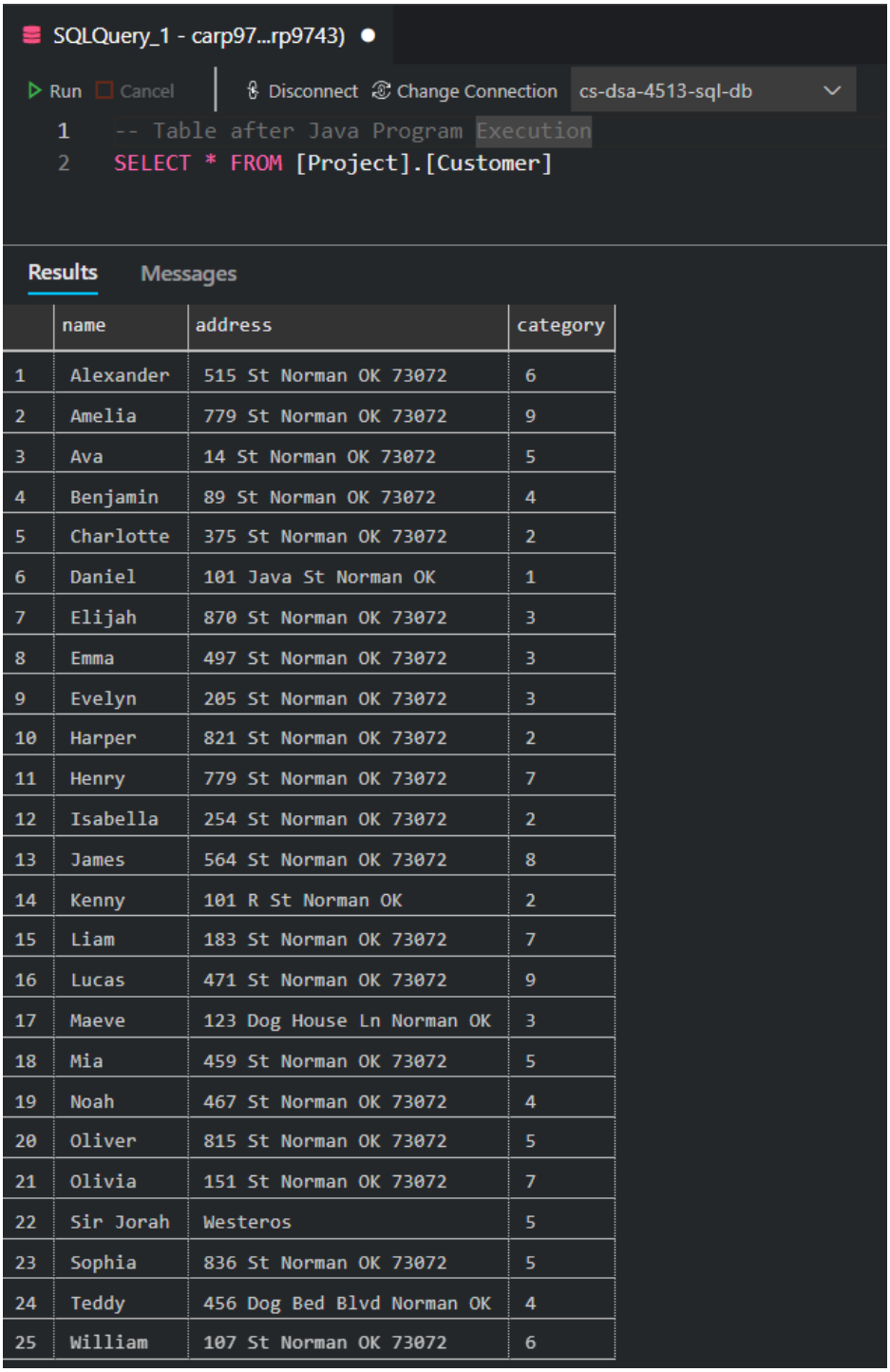
```
(18) Quit

16
Please enter the location and name of a CSV file with customer data:
>> PLEASE DO NOT INCLUDE COMMAS EXCEPT FOR THE DELIMITER <<
C:\\Users\\daniel.carpenter\\OneDrive - the Chickasaw Nation\\Documents\\GitHub\\OU-DSA\\db MG
Dispatching the query...
Done. 20 rows inserted.
```

6.16. Screenshots showing the testing of the import and export options *(Continued)*

Import Option (16) *(Continued)*:

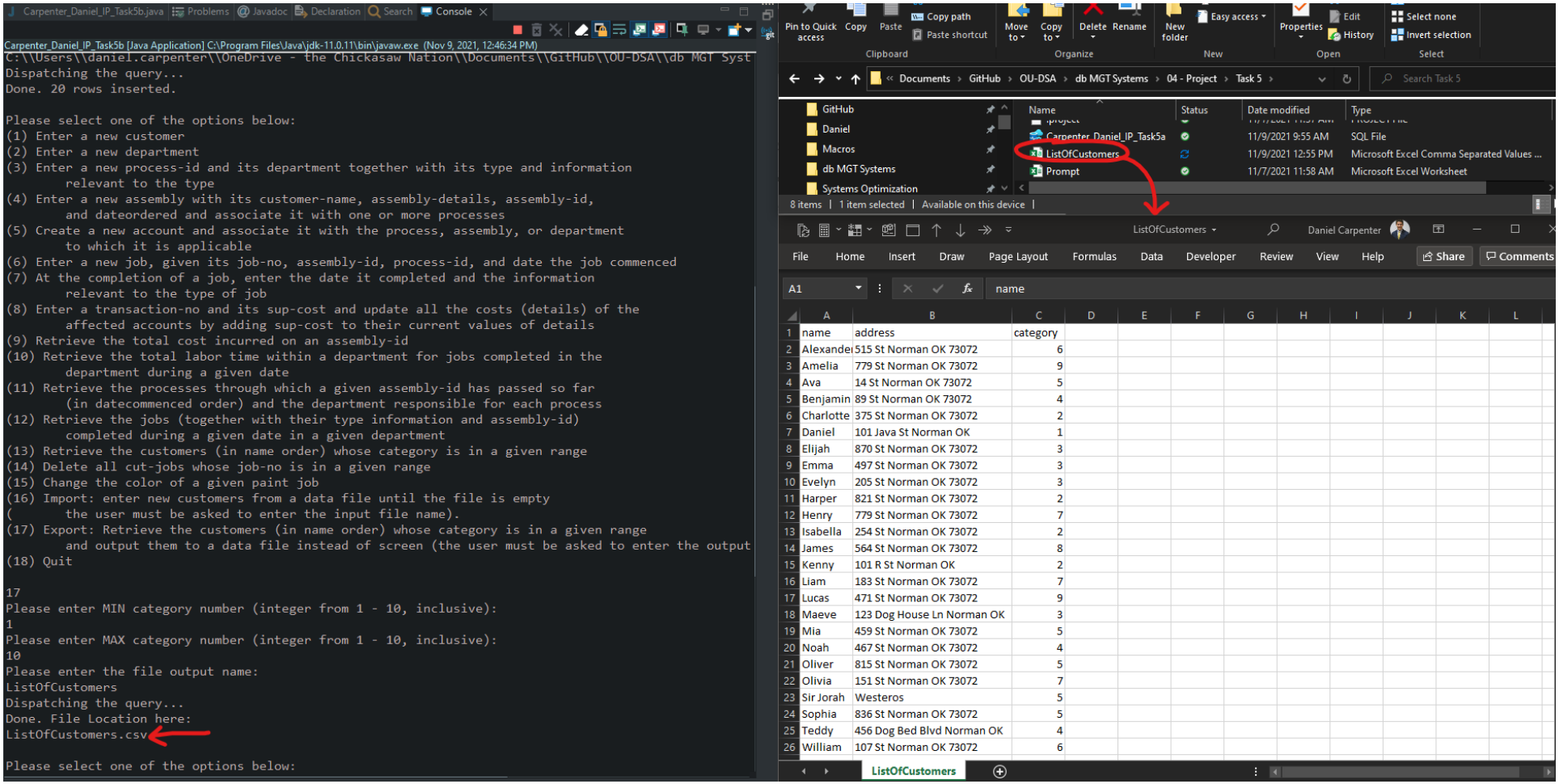
Customer table after uploading file



	name	address	category
1	Alexander	515 St Norman OK 73072	6
2	Amelia	779 St Norman OK 73072	9
3	Ava	14 St Norman OK 73072	5
4	Benjamin	89 St Norman OK 73072	4
5	Charlotte	375 St Norman OK 73072	2
6	Daniel	101 Java St Norman OK	1
7	Elijah	870 St Norman OK 73072	3
8	Emma	497 St Norman OK 73072	3
9	Evelyn	205 St Norman OK 73072	3
10	Harper	821 St Norman OK 73072	2
11	Henry	779 St Norman OK 73072	7
12	Isabella	254 St Norman OK 73072	2
13	James	564 St Norman OK 73072	8
14	Kenny	101 R St Norman OK	2
15	Liam	183 St Norman OK 73072	7
16	Lucas	471 St Norman OK 73072	9
17	Maeve	123 Dog House Ln Norman OK	3
18	Mia	459 St Norman OK 73072	5
19	Noah	467 St Norman OK 73072	4
20	Oliver	815 St Norman OK 73072	5
21	Olivia	151 St Norman OK 73072	7
22	Sir Jorah	Westeros	5
23	Sophia	836 St Norman OK 73072	5
24	Teddy	456 Dog Bed Blvd Norman OK	4
25	William	107 St Norman OK 73072	6

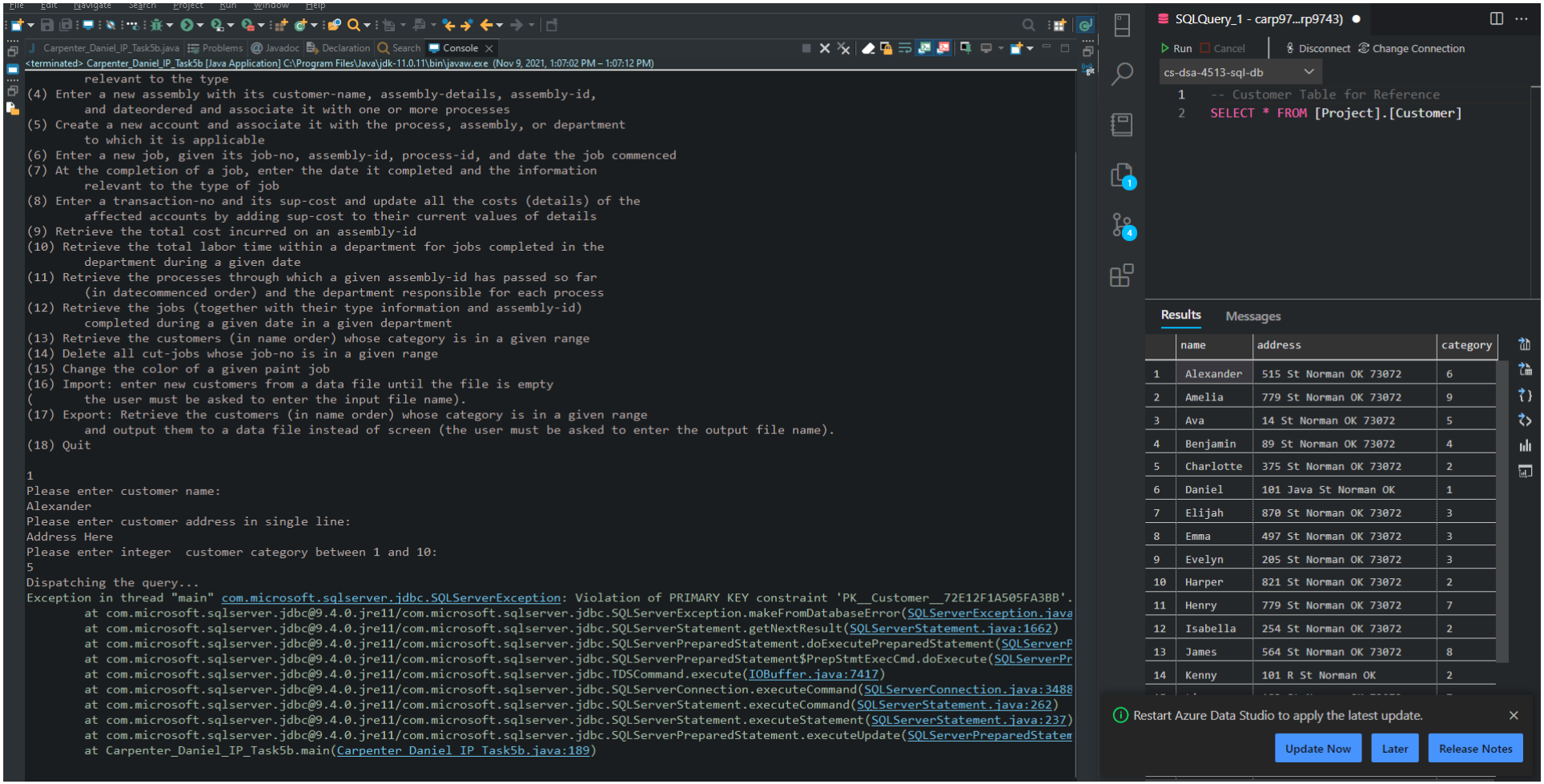
6.16. Screenshots showing the testing of the import and export options *(Continued)*

Export Option (17):



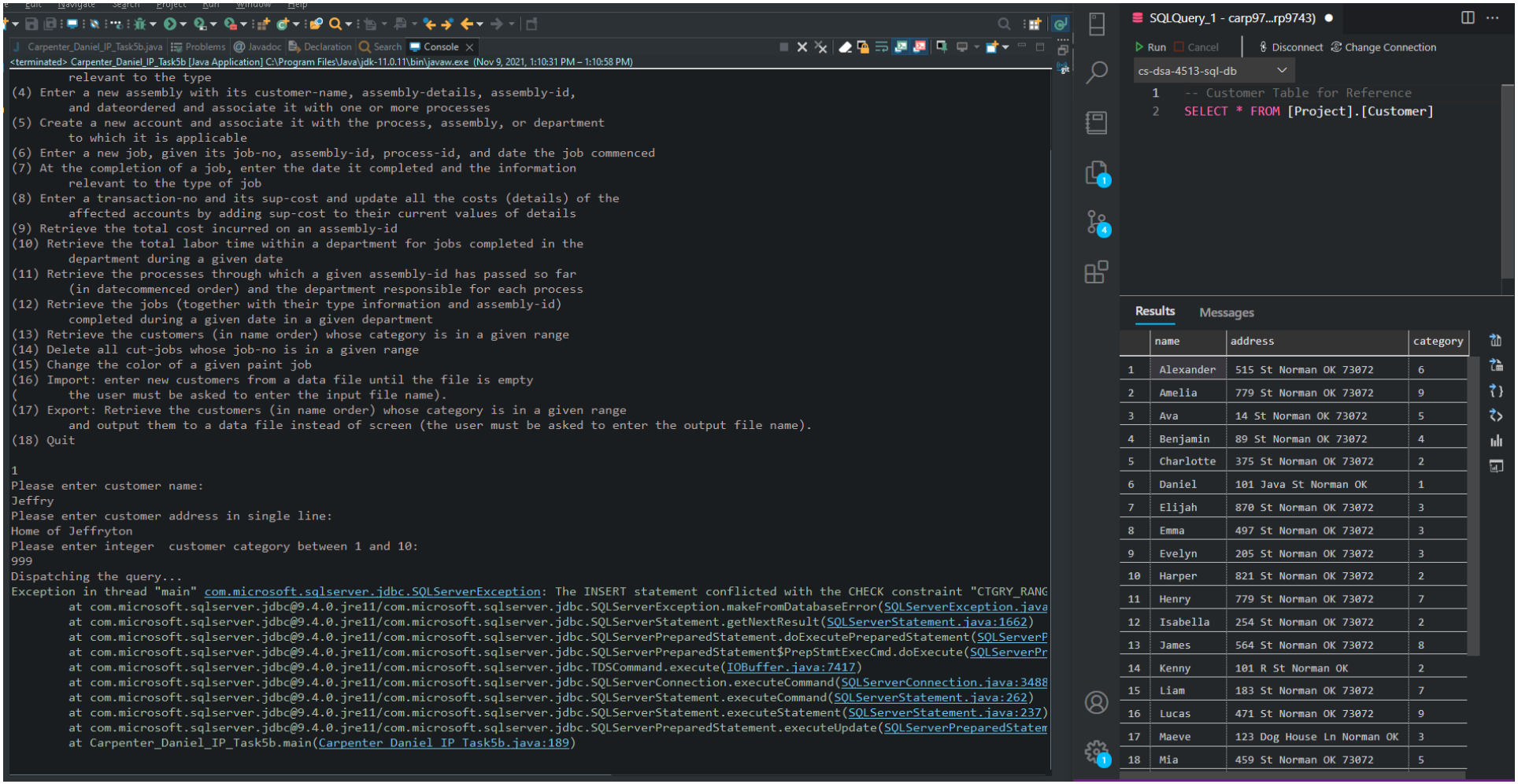
6.17. Screenshots showing the Testing of Three Types of Errors

Error Example 1 – Violating a Primary Key constraint on the Table 'Customer':



6.17. Screenshots showing the Testing of Three Types of Errors *(Continued)*

Error Example 2: Entering data outside of constraint range of table 'Customer' on attribute 'category'. Expected values 1 through 10



```
Carpenter_Daniel_IP_Task5b.java  Problems  Javadoc  Declaration  Search  Console
<terminated> Carpenter_Daniel_IP_Task5b [Java Application] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe  (Nov 9, 2021, 1:10:31 PM – 1:10:58 PM)
        relevant to the type
(4) Enter a new assembly with its customer-name, assembly-details, assembly-id,
        and dateordered and associate it with one or more processes
(5) Create a new account and associate it with the process, assembly, or department
        to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information
        relevant to the type of job
(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
(9) Retrieve the total cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the
        department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far
        (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id)
        completed during a given date in a given department
(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
(15) Change the color of a given paint job
(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 customer name:
Jeffry
Please enter customer address in single line:
Home of Jeffryton
Please enter integer  customer category between 1 and 10:
999
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The INSERT statement conflicted with the CHECK constraint "CTGRY_RANG
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1662)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerP
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPr
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7417)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3488
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:262)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:237)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatem
        at Carpenter_Daniel_IP_Task5b.main(Carpenter Daniel IP Task5b.java:189)
```

SQLQuery_1 - carp97...rp9743)

Run Cancel Disconnect Change Connection

cs-dsa-4513-sql-db

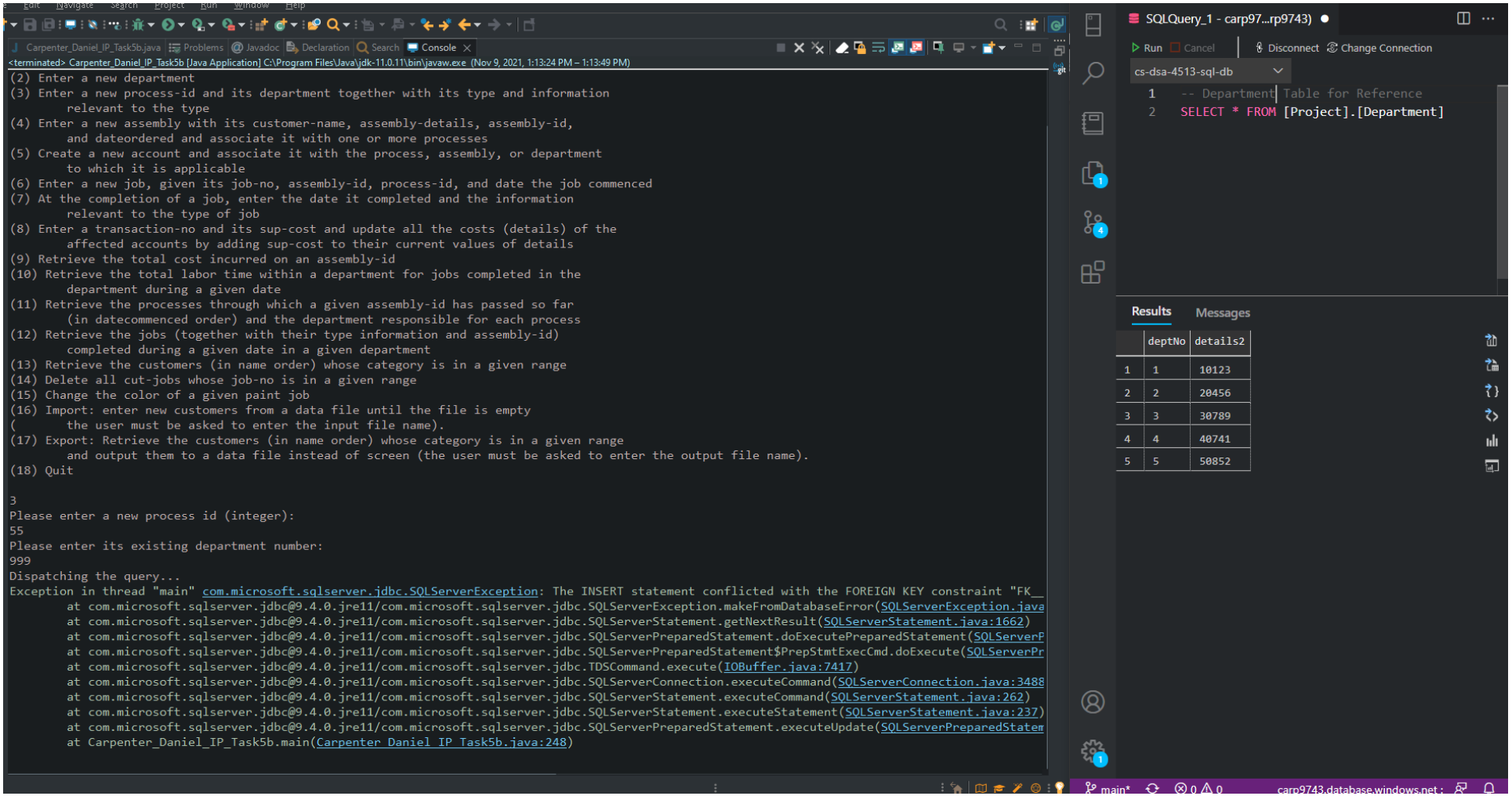
```
1   -- Customer Table for Reference
2   SELECT * FROM [Project].[Customer]
```

Results Messages

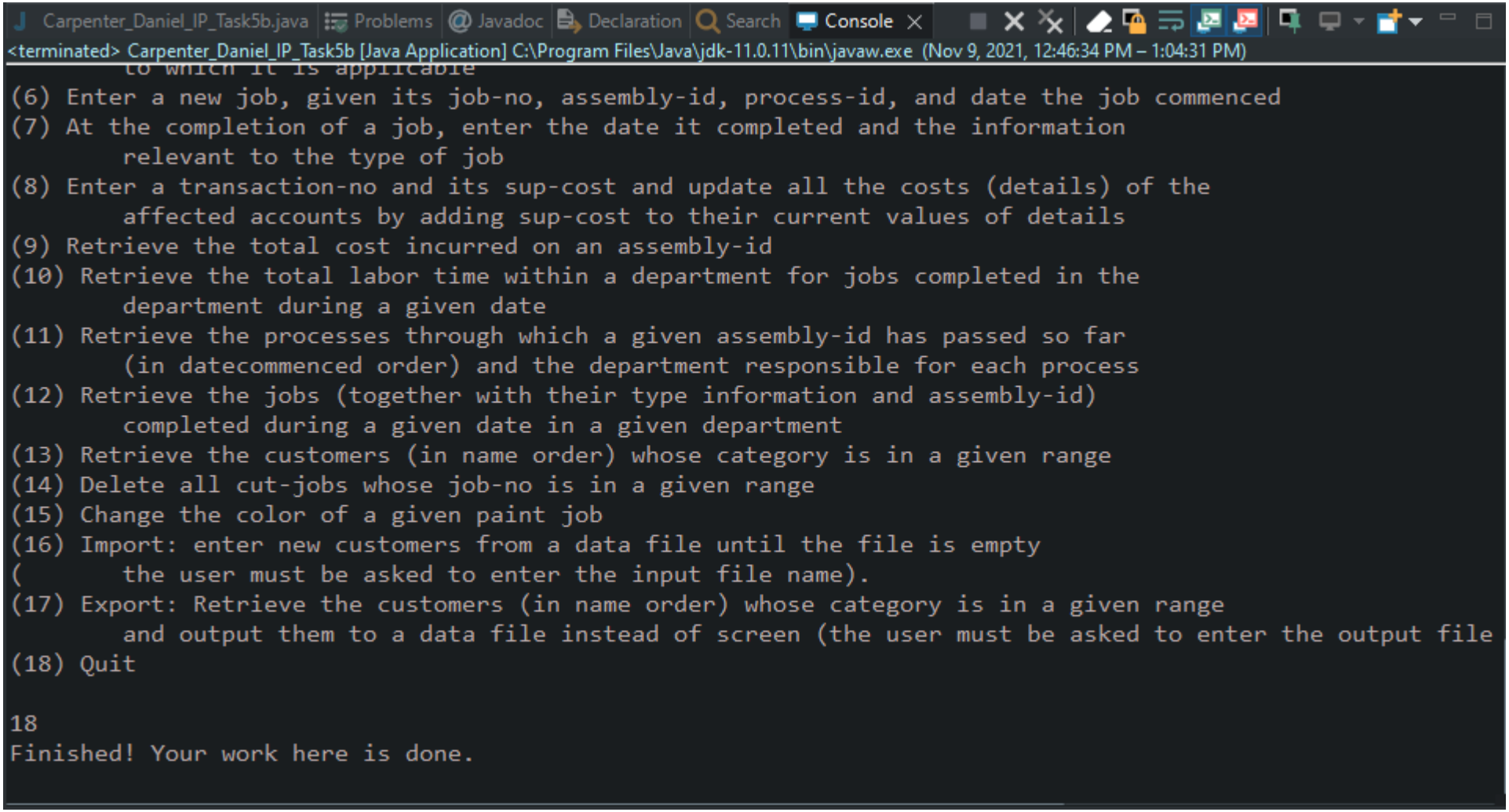
	name	address	category
1	Alexander	515 St Norman OK 73072	6
2	Amelia	779 St Norman OK 73072	9
3	Ava	14 St Norman OK 73072	5
4	Benjamin	89 St Norman OK 73072	4
5	Charlotte	375 St Norman OK 73072	2
6	Daniel	101 Java St Norman OK	1
7	Elijah	870 St Norman OK 73072	3
8	Emma	497 St Norman OK 73072	3
9	Evelyn	205 St Norman OK 73072	3
10	Harper	821 St Norman OK 73072	2
11	Henry	779 St Norman OK 73072	7
12	Isabella	254 St Norman OK 73072	2
13	James	564 St Norman OK 73072	8
14	Kenny	101 R St Norman OK	2
15	Liam	183 St Norman OK 73072	7
16	Lucas	471 St Norman OK 73072	9
17	Maeve	123 Dog House Ln Norman OK	3
18	Mia	459 St Norman OK 73072	5

6.17. Screenshots showing the Testing of Three Types of Errors *(Continued)*

Error Example 3: Violating foreign key constraint by adding new Assembly with a non-existing 'deptNo' (which is the foreign key)



6.18. Screenshots showing the Testing of the Quit Option



7.1. Web Database Application Source Program

and Associated Screenshots of Successful Compilation

Carpenter_Daniel_IP_Task7_DataHandler.java

```java
package JobShopProject;

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

//=========================================================================
//@class:  DSA 4513
//@asnmt:  Class Project
//@task:   7
//@author: Daniel Carpenter, ID: 113009743
//@description:
//  Program to implement job-shop accounting system query 1 and 13
//=========================================================================

public class Carpenter_Daniel_IP_Task7_DataHandler {

    private Connection conn;

    // Azure SQL connection credentials
    private String server   = "carp9743.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "carp9743";
    private String password = "tacoBout$97315!";
```

```java
// Resulting connection string
final private String url =
        String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;host
                server, database, username, password);

// Initialize and save the database connection
private void getDBConnection() throws SQLException {
    if (conn != null) {
        return;
    }

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

// Adds a customer to the Customer table and returns true if executed correctly
public boolean addCustomer(String name, String address, int category) throws SQLException {
    getDBConnection();

    // Prepare the query
    final String sqlQuery = "EXEC [Project].addCustomer @name = ?,  @address = ?, @category = ?;";

    // Replace the '?' in the above statement with the given attribute values
    final PreparedStatement statement = conn.prepareStatement(sqlQuery);
        statement.setString(1,  name);
        statement.setString(2,  address);
        statement.setInt(3,     category);

    // Return true if successful
    return statement.executeUpdate() == 1;
}

// Inserts a record into the movie_night table with the given attribute values
```

```
    public ResultSet getCustomersInRange(int min, int max) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Query
        String sqlQuery = "{CALL [Project].getCustomers(?, ?)}";

        // Prepare query call
        CallableStatement statement = conn.prepareCall(sqlQuery);

        // Set the assigned value(s) to the procedures input '?'
        statement.setInt("min", min);
        statement.setInt("max", max);

        // Execute the query
        return statement.executeQuery();
    }
}
```

Carpenter_Daniel_IP_Task7_getCustomersInRange.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="JobShopProject.Carpenter_Daniel_IP_Task7_DataHandler"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Array"%>
<%
// The handler is the one in charge of establishing the connection.
Carpenter_Daniel_IP_Task7_DataHandler handler = new Carpenter_Daniel_IP_Task7_DataHandler();

// Get the attribute values passed from the input form.
String min  = request.getParameter("min");
String max  = request.getParameter("max");

 // Assume all categories if query fails
 int minAsInt = 1;
 int maxAsInt = 10;
 ResultSet customers;

// detect input
if (min.equals("") || max.equals("")) {
    response.sendRedirect("Carpenter_Daniel_IP_Task7_getCustomersInRange.jsp");
} else {
  // Get the actual input
  minAsInt = Integer.parseInt(min);
  maxAsInt = Integer.parseInt(max);

}
// Now perform the query with the data from the form.
customers = handler.getCustomersInRange(minAsInt, maxAsInt);


%>
<!-- 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>
        </tr>
        <%
           while(customers.next()) { // For each Customer record returned...
               // Extract the attribute values for every row returned
               final String name     = customers.getString("name");

               out.println("<tr>"); // Start printing out the new table row
               out.println( // Print each attribute value
                    "<td align=\"center\">" + name + "</td>");
               out.println("</tr>");
           }
           %>
      </table>
      <a href="Carpenter_Daniel_IP_Task7_addCustomerForm.jsp">Add more customers.</a>

    </body>
</html>
```

Carpenter_Daniel_IP_Task7_getCustomersInRangeForm.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Get customers whose category is in a given range </title>
    </head>
    <body>
        <h2>Get Customers</h2>
```

```
        <!--
            Form for collecting user input for the new Customer record.
            Upon form submission, addCustomer.jsp file will be invoked.
        -->
        <form action="Carpenter_Daniel_IP_Task7_getCustomersInRange.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Retrieve the customers (in name order) whose category is in range 1 through 10
(integer):</th>
                </tr>
                <tr>
                    <td>Min Value:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=min>
                    </div></td>
                </tr>
                <tr>
                    <td>Max Value:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=max>
                    </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=Submit>
                    </div></td>
                </tr>
            </table>
        </form>
```

```
    </body>
</html>
```

Carpenter_Daniel_IP_Task7_addCustomer.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="JobShopProject.Carpenter_Daniel_IP_Task7_DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    Carpenter_Daniel_IP_Task7_DataHandler handler = new Carpenter_Daniel_IP_Task7_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 fields. This is very simple checking.
     */
    if (name.equals("") || address.equals("") || category.equals("")) {
        response.sendRedirect("Carpenter_Daniel_IP_Task7_addCustomerForm.jsp");
    } else {
        int categoryAsInt = Integer.parseInt(category);

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

```
        boolean success = handler.addCustomer(name, address, categoryAsInt);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The Customer:</h2>

            <ul>
                <li>Customer Name: <%=name%></li>
                <li>Address: <%=address%></li>
                <li>Category: <%=categoryAsInt%></li>
            </ul>

            <h2>Was successfully inserted.</h2>

            <a href="Carpenter_Daniel_IP_Task7_getCustomersInRangeForm.jsp">Retrieve list of customers.</a>
            <%
        }
    }
    %>
    </body>
</html>
```

Carpenter_Daniel_IP_Task7_addCustomerForm.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
```

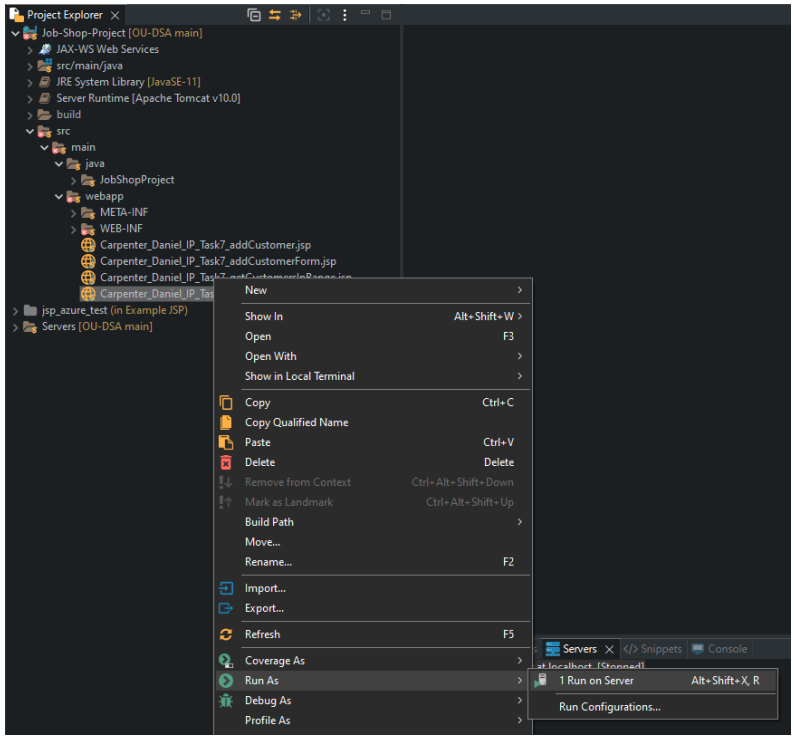
```
        <title>Add Customer</title>
    </head>
    <body>
        <h2>Add Customer</h2>
        <!--
            Form for collecting user input for the new Customer record.
            Upon form submission, addCustomer.jsp file will be invoked.
        -->
        <form action="Carpenter_Daniel_IP_Task7_addCustomer.jsp">
            <!-- 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>Customer 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>
```

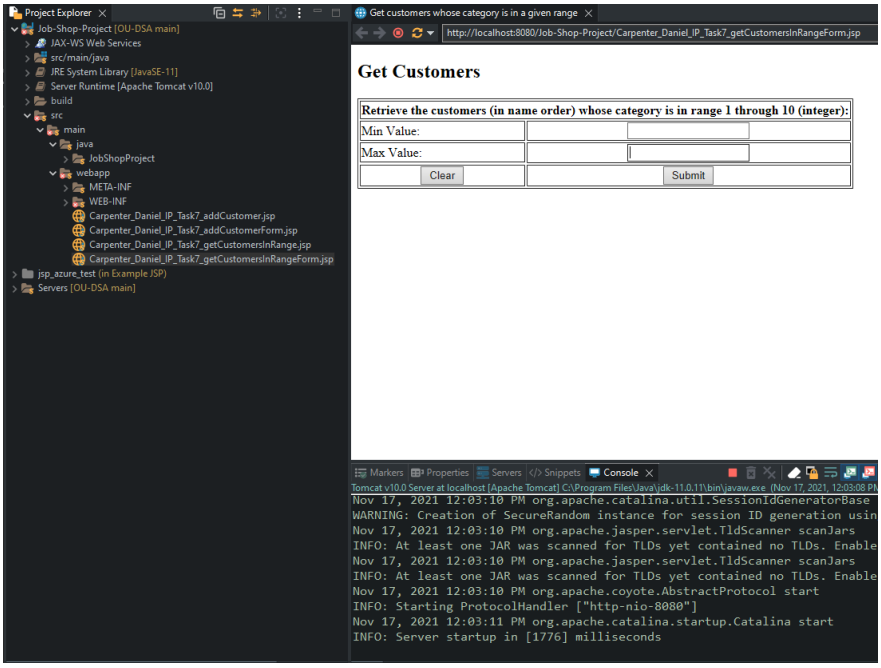
7.1. Web Database Application Source Program

and Associated Screenshots of Successful Compilation *(Continued)*

Prior to clicking Project

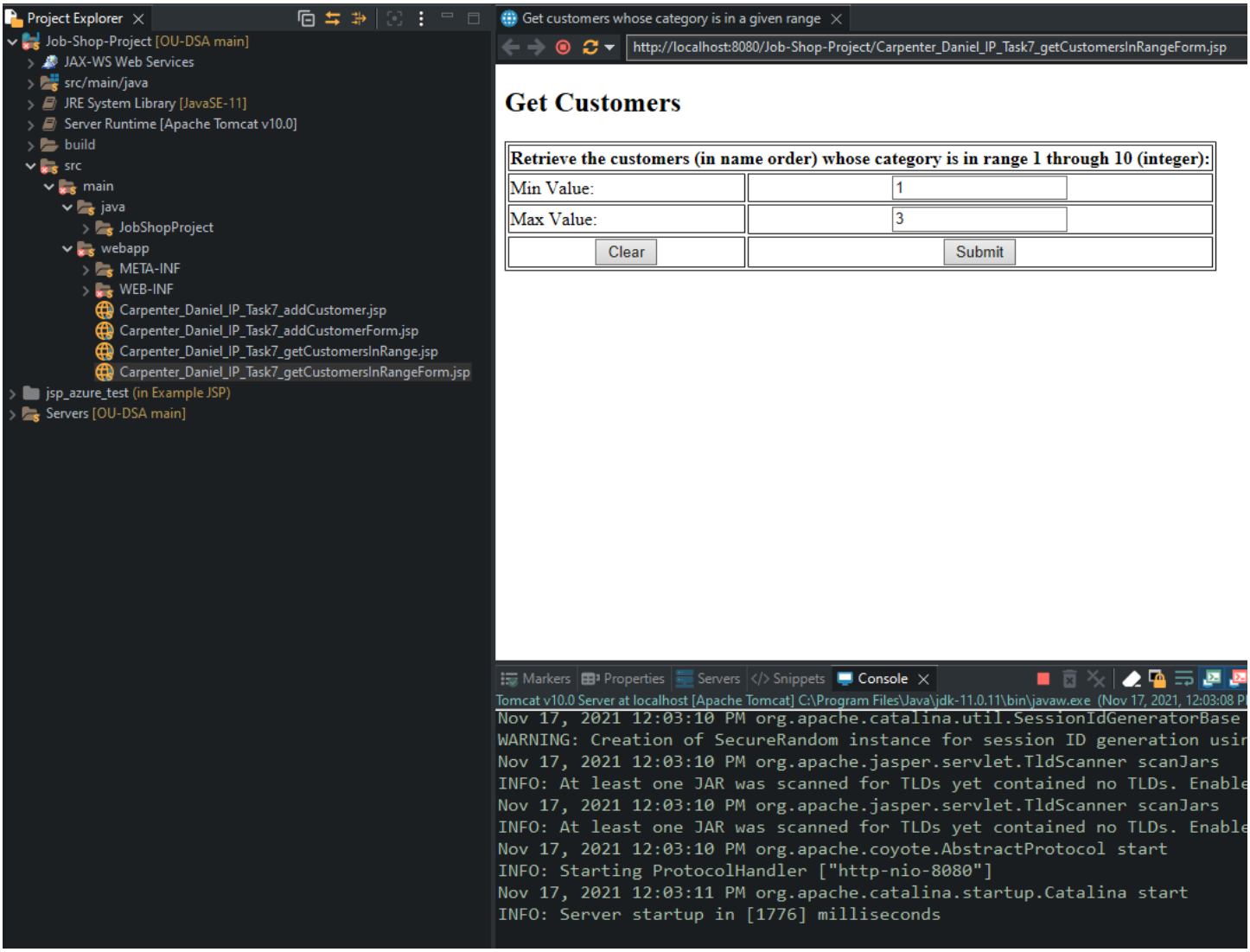


Upon Clicking “Run on Server”



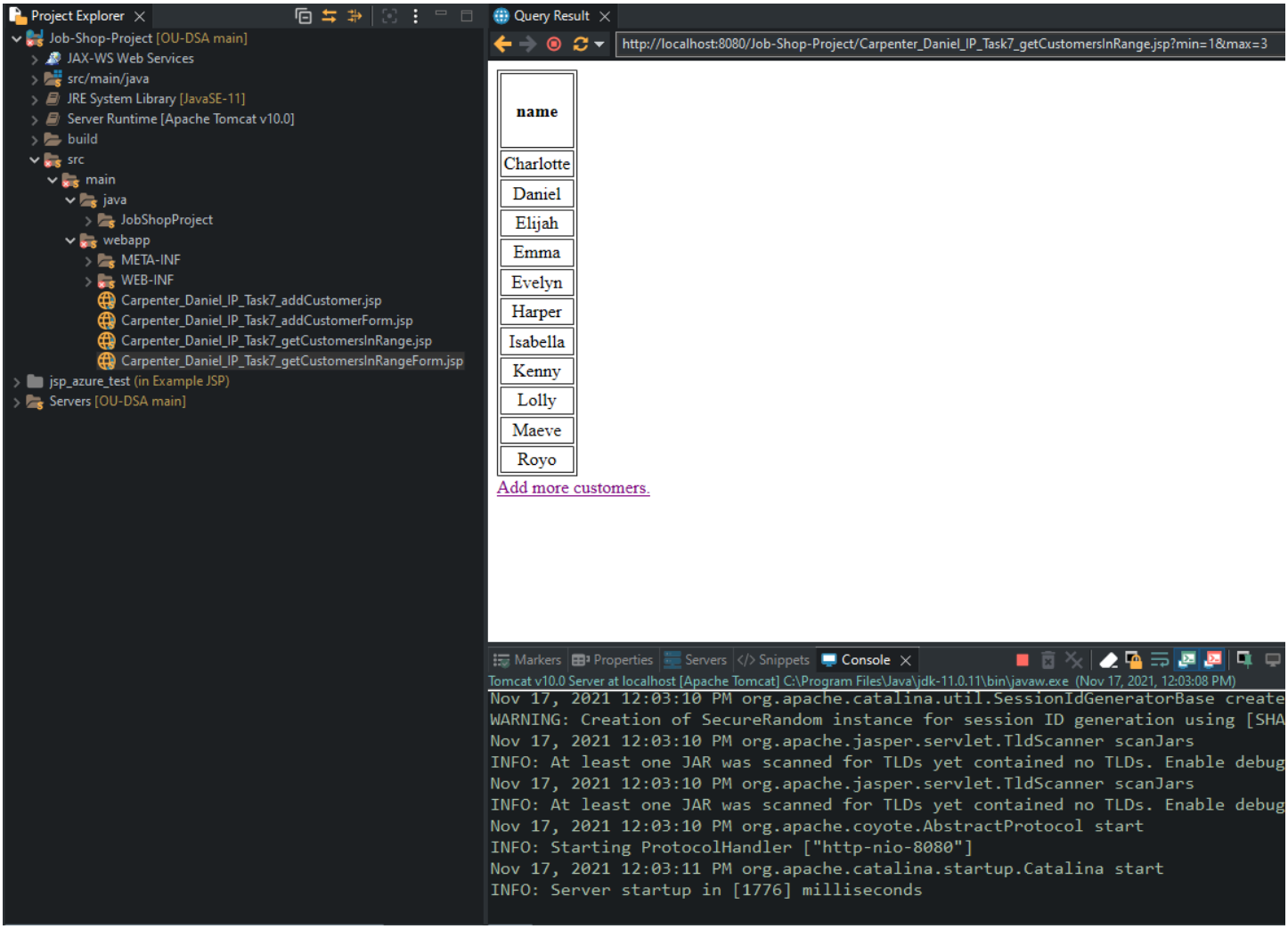
7.2. Screenshots Showing the Testing of the Web Database Application

Screenshot 1: Application to get Customers, *(with inputs range inputs from 1 to 5)*



7.2. Screenshots Showing the Testing of the Web Database Application *(Continued)*

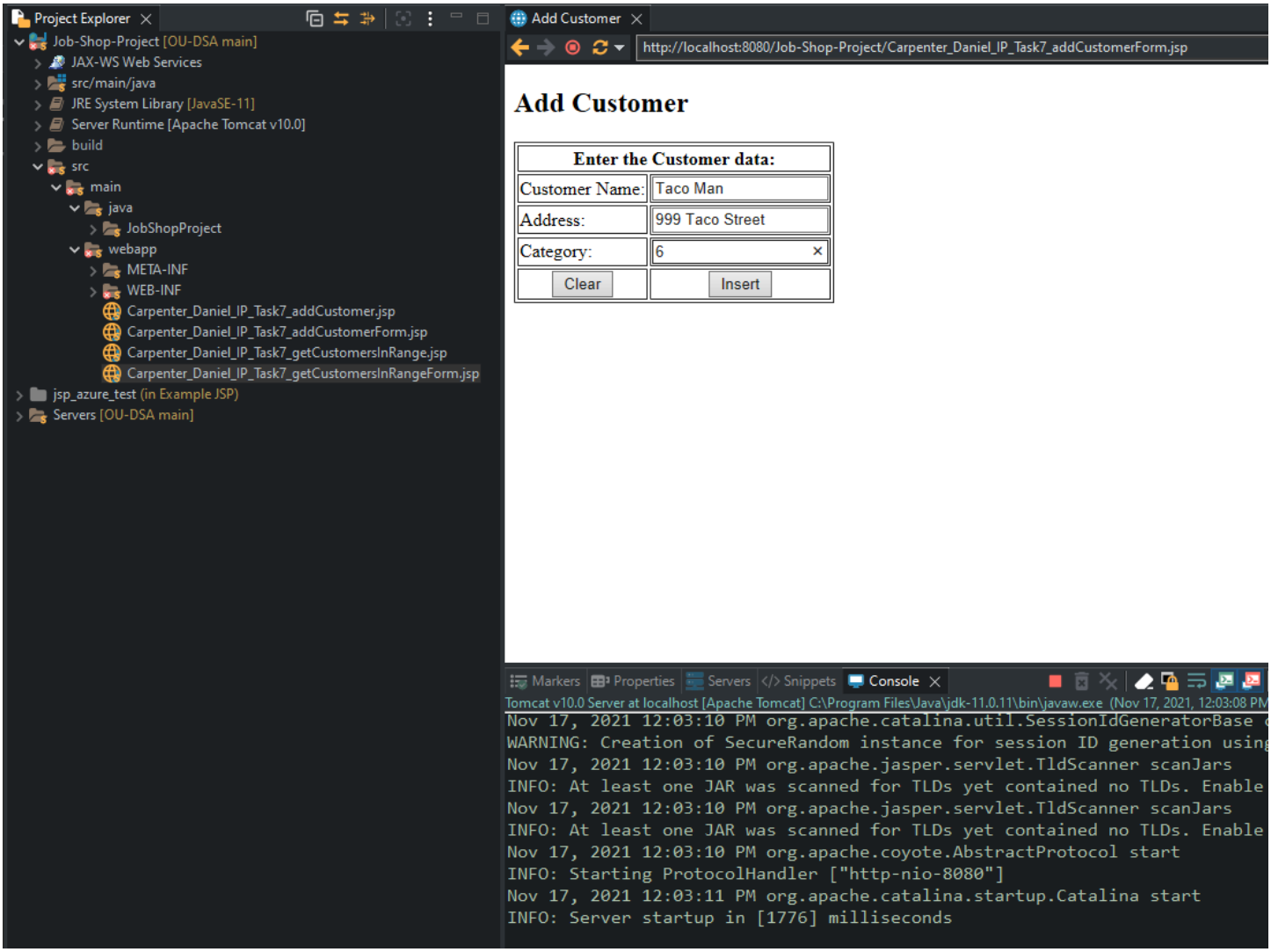
Screenshot 2: Application after clicking "Submit" on the prior page



7.2. Screenshots Showing the Testing of the Web Database Application *(Continued)*

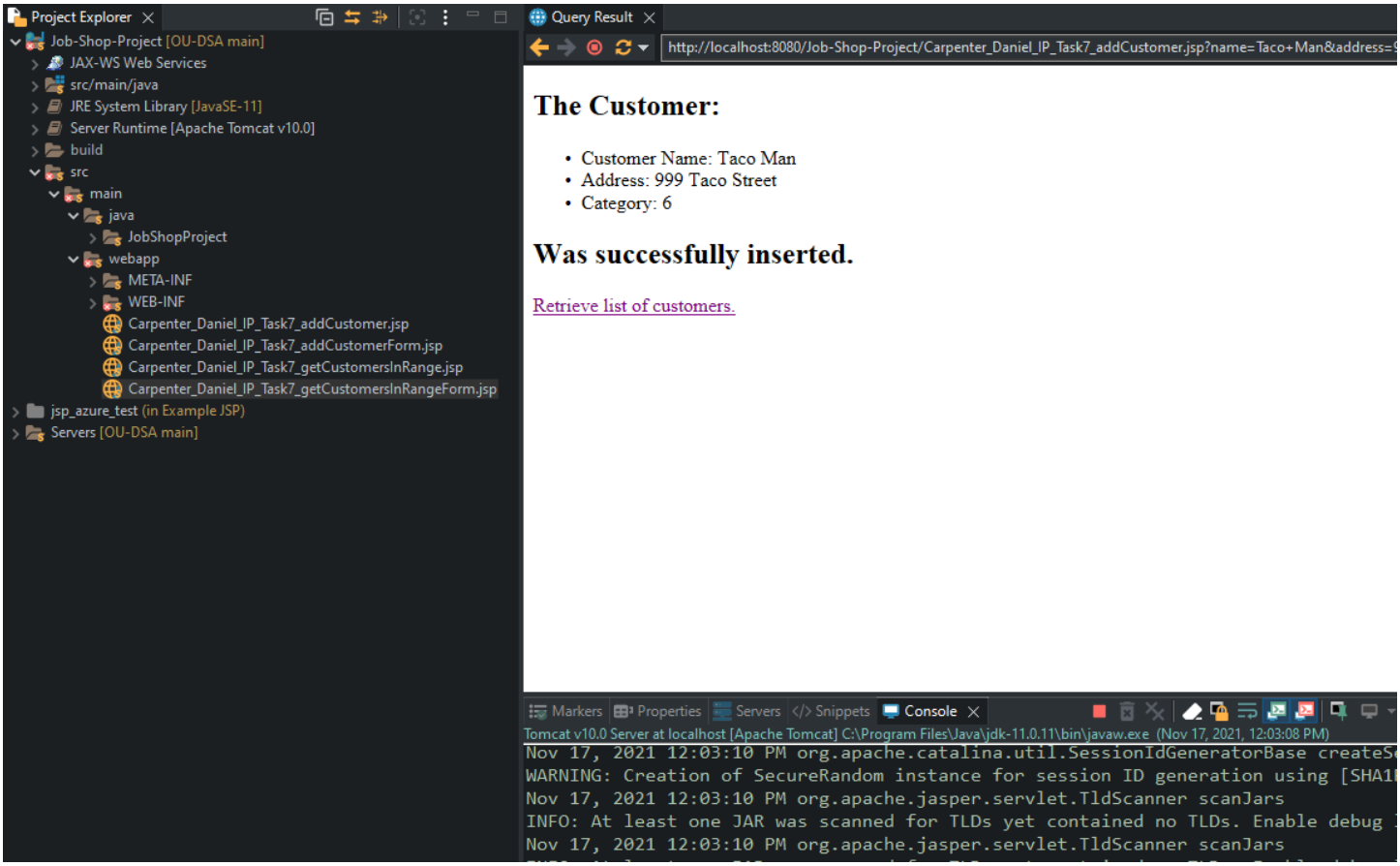
Screenshot 3: Application after clicking the "Add more customers" button on prior page

(with relevant inputs to the Customer table)



7.2. Screenshots Showing the Testing of the Web Database Application *(Continued)*

Screenshot 4: Application after clicking the "Insert" button on prior page



7.2. Screenshots Showing the Testing of the Web Database Application *(Continued)*

Screenshot 5: Application after clicking the "Retrieve list of customers." button on prior page