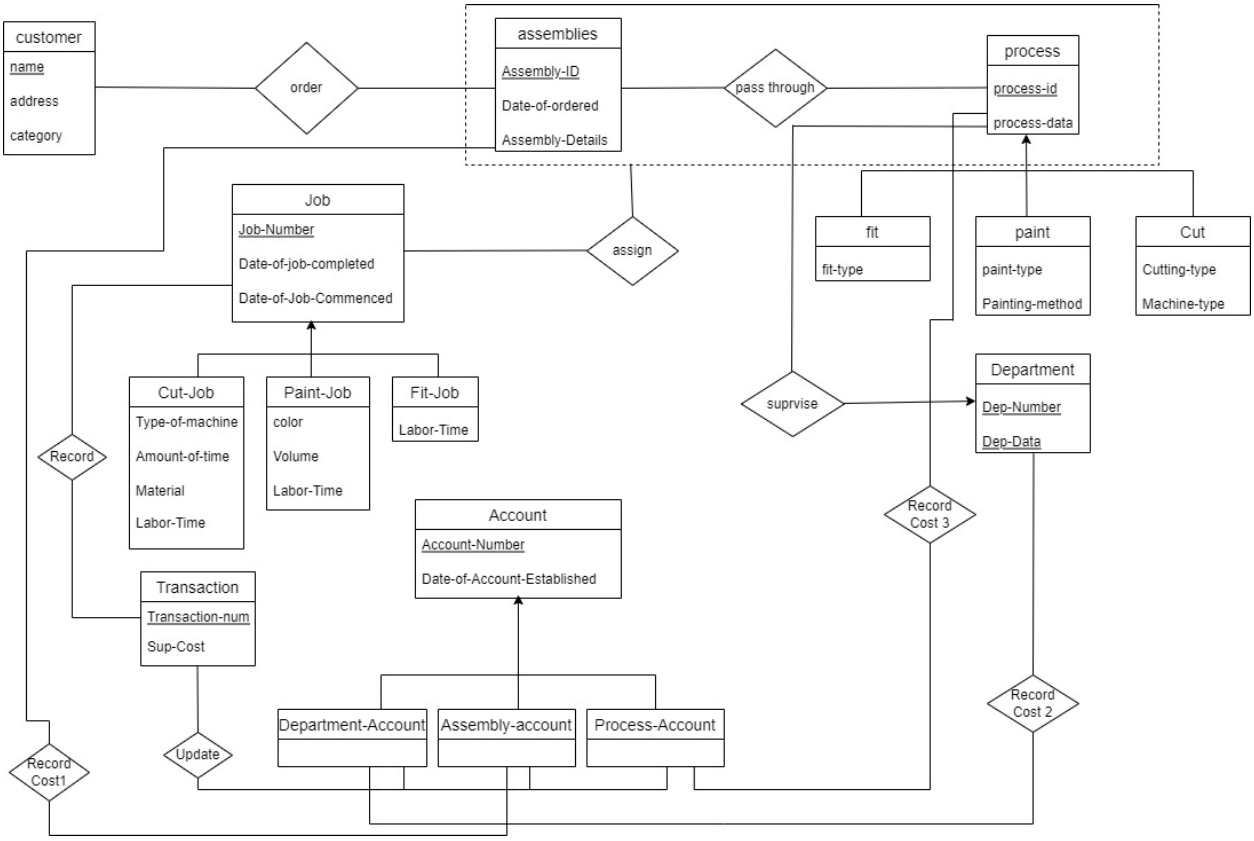
Cs-4513-001-Fall 2021, Dr. Le Gruenwald,
Naeem Shahabi Sani, 113526767, shahabi@ou.edu,
INDIVIDUAL PROJECT: A JOB-SHOP ACCOUNTING SYSTEM

Tasks Performed **Page Number**

Tasks Performed **Page Number**

Task 1.

1.1. ER Diagram



1.2. Relational Database Schema

Customer (name ,address ,category)

Assembly (Assembly-id ,date-of-ordered ,Assembly-details)

Process (Process-Id ,Process-data)

Order (Assembly-id ,cname)

Fit (Process-Id ,Fit-type)

Paint (Process-Id ,paint-type ,Painting-method)

Cut (process-Id, Cutting-type ,Machine-type)

Job (Job-number ,Date-of-job-commenced ,Date-of-job-completed)

Cut-job (Job-number , type-of-machine ,amount-of-time ,material ,labor-time)

Paintjob (job-number , color, volume ,labor-time)

Fitjob (Job-number ,labor-time)

Account (Account-number ,date-of-account-established)

Department-account (Account-number)

Assembly-account (Account-number)

Process-account (account-number)

Transaction (Transaction-num, sup-cost)

Department (Dep-num, Dep-data)

Assign(Assembly-id, process-Id, job-number)

Pass-through(Assembly-id, process-Id)

Record(transaction-num, job-number)

Record cost 3(Account-number, process-Id)

Record cost 2 (Account-number, dep-number)

Record cost 1 (Account-number, Assembly-id)

Supervise (process-Id, Dep-number, Dep-data)

Update (transaction-num, Account-Number)

Task 2. Data Dictionary

<Customer>

Field Name	Data Type	Size (Byte)	Constraints
Name	Varchar	32 Byte	Primary Key
Address	Varchar	32 Byte	
Category	Integer	4 Byte	1-10

<Assembly>

Field Name	Data Type	Size (Byte)	Constraints
Assembly-ID	INT	5	Primary Key, Not Null
Date-Of-Ordered	Datetime	8	
Assembly-Details	Varchar	32 Byte	

<Assign>

Field Name	Data Type	Size (Byte)	Constraints
Assembly-ID	INT	5	FK, Primary Key, Not Null
Process-ID	INT	5	FK, Primary Key, Not Null
Job number	INT	5	FK, Primary Key, Not Null

<Pass-through>

Field Name	Data Type	Size (Byte)	Constraints
Assembly-Id	INT	5	FK, Primary Key, not null
Process-ID	INT	5	FK, Primary Key, not null

<Process>

Field Name	Data Type	Size (Byte)	Constraints
Process-ID	INT	5	Primary Key, not null
Process-Data	Varchar	32	

<Order>

Field Name	Data Type	Size (Byte)	Constraints
Assembly-Id	INT	5	FK, Primary Key, not null
CName	Varchar	32	FK, Primary Key, not null

<Fit>

Field Name	Data Type	Size (Byte)	Constraints
Process-Id	INT	5	FK, Primary Key, not null
Fit-Type	Varchar	32	

<Paint>

Field Name	Data Type	Size (Byte)	Constraints
Process-Id	INT	5	FK, Primary Key, Not null
Paint Type	Varchar	32	
Painting-Method	Varchar	32	

<Cut>

Field Name	Data Type	Size (Byte)	Constraints
Process- Id	INT	5	FK, Primary Key, Not null
Machine-Type	Varchar	32	
Cutting-Type	Varchar	32	

<Job>

Field Name	Data Type	Size (Byte)	Constraints
Job-Nom	INT	5	Primary Key, Not null
Date-Of-Job-Commenced	datetime	8	
Date-Of-Job-Completed	datetime	8	

<Cut-Job>

Field Name	Data Type	Size (Byte)	Constraints
Type-Of-Machine	Varchar	32	
Amount-Of-Time	INT	5	
Material	Varchar	32	
Labor-Time	INT	5	
Job-Number	INT	5	FK, Primary Key, Not null

<Paint-Job>

Field Name	Data Type	Size (Byte)	Constraints
Job-Number	INT	5	FK, Primary Key, Not null
Color	Varchar	32	
Volume	Varchar	32	
Labor-Time	INT	5	

<Fit-Job>

Field Name	Data Type	Size (Byte)	Constraints
Job-Number	INT	5	FK, Primary Key, Not null
Labor-Time	INT	5	

<Account>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	Primary Key, Not null
Date-Of-Account-Established	Datetime	8	

<Department>

Field Name	Data Type	Size (Byte)	Constraints
Dep-Number	INT	5	Primary Key, Not null
Dep-Data	Varchar	32	Primary Key, Not null

<Department-Account>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	FK, Primary Key, Not null

<Assembly-Account>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	Fk, Primary Key, Not null

<Process-Account>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	FK, Primary Key, Not null

<Transaction>

Field Name	Data Type	Size (Byte)	Constraints
Transaction-Num	INT	5	Primary Key, Not null
Sup Cost	REAL	5	

<Record>

Field Name	Data Type	Size (Byte)	Constraints
Transaction-Num	INT	5	FK, Primary Key, Not null
Sup Cost	REAL	5	

<Record cost2>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	FK, Primary Key, Not null
Dep-number	INT	5	FK, Primary Key, Not null

<Record Cost1>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	FK, Primary Key, Not null
Assembly-id	INT	5	FK, Primary Key, Not null

<Record Cost3>

Field Name	Data Type	Size (Byte)	Constraints
Account-Number	INT	5	FK, Primary Key, Not null
Process-id	INT	5	FK, Primary Key, Not null

<Supervise>

Field Name	Data Type	Size (Byte)	Constraints
Process-Id	INT	5	FK, Primary Key, Not null
Dept Number	INT	5	FK, Primary Key, Not null
Dep- Data	Varchar	32	FK, Primary Key, Not null

<Update>

Field Name	Data Type	Size (Byte)c	Constraints
Transaction-num	INT	5	FK, Primary Key, Not null
Account number	INT	5	FK, Primary Key, Not null

Task 3.

3.1. Discussion of storage structures for tables

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
customer	(1).insert 13. range search	 category	30/day 100/day	Index sequential On search key category	Because it is Range search We use index sequential

Assembly	4.insert		40/day	Heap file	Because it is insert
Record cost 3	5. insert		10/day	Heap file	Because it is insert
Assign	6. insert		50/day	Heap file	Because it is insert
Record cost 1	5.insert		10/day	Heap file	Because it is insert
Process	3.insert		infrequent	Heap file	Because it is insert
Department	3.insert		infrequent	Heap file	Because it is insert
Order	4.insert		40/day	Heap file	Because it is insert
Pass through	4.enter 11. Retrieve	Process-id	40/day 100/day	Dynamic hashing With hash key Process id	Because retrie Is like random search
Account	5.create		10/day	Heap file	Because it is insert
Record cost 2	5.create		10/day	Heap file	Because it is insert
Record cost 1	5.create		10/day	Heap file	Because it is insert
Job	6.insert 12. Retrieve 12. Random searc	Given date	50/day 20/day 20/day	Heap file	Because it is insert
Cut-job	7.enter 14.Delete 14.range search	Job -no	50/day 1/month 1/month	Heap file	Because it is insert
Pain-job	7.enter 14. change		50/day 1/month	Heap file	Because it is insert
Fit-job	7.insert		50/day	Heap file	Because it is insert
Transaction	8.insert		50/day	Heap file	Because it is insert
Record cost 1	9.Retrieve		200/day	Dynamic hashing With hash key Assembly id or Account number	Because retrie Is like random search

3.2. Discussion of storage structures for tables (Azure SQL Database)

Azure Storage is a Microsoft-managed cloud service that provides storage that is highly available, secure, durable, scalable and redundant. Whether it is images, audio, video, logs, configuration files, or sensor data from an IoT array, data needs to be stored in a way that can be easily accessible for analysis purposes, and Azure Storage provides options for each one of these possible use cases.

Within Azure there are two types of storage accounts, four types of storage, four levels of data redundancy and three tiers for storing files. We will explore each one of these options in detail to help you understand which offering meets your big data storage needs.

Azure Storage Account Types

An Azure storage account is an access point to all the elements that compose the Azure storage realm. Once the user creates the storage account, they can select the level of resilience needed and Azure will take care of the rest. A single storage account can store up to 500TB of data and like any other Azure service, users can take advantage of the pay-per-use pricing model.
There are two different storage account types. With the "standard" storage account, users get access to Blob Storage, Table Storage, Queue Storage and File Storage. The alternative, "premium" account, is the most recent storage option which provides users with data storage on SSD drives for better I/O performance; this option supports only page blobs.

Azure Blob Storage
Blob Storage is Microsoft Azure's service for storing binary large objects or blobs which are typically composed of unstructured data such as text, images, and videos, along with their metadata. Blobs are stored in directory-like structures called "containers."
The blob service includes:

- Blobs, which are the data objects of any type
- Containers, which wrap multiple blobs together
- Azure storage account, which contains all of your Azure storage data objects

Blob Storage Categories
Although blob allows for storage of large binary objects in Azure, these are optimized for three different storage scenarios:

- Block blobs: These are blobs that are intended to store discrete objects such as images, log files and more. Block blobs can store data up to ~5TB, or 50,000 blocks of up to 100MB each.
- Page blobs: These are optimized for random read and write operations and can grow up to 8TB in size. Within the page blob category, Azure offers two types of storage: standard and premium. The latter is the most ideal for virtual machine (VM) storage disks (including the operating system disk).
- Append Blobs: Optimized for append scenarios like log storage, append blogs are composed of several blocks of different sizes — up to a maximum of 4MB. Each append blob can hold up to 50,000 blocks, therefore allowing each append blob to grow up to 200GB.

Blob storage accounts offer three types of tiers that are selected at the time of creation of the storage account.

- Hot Access Tier: Out of the three options, the hot access tier is the most optimized for data that is accessed frequently. It offers the lowest access (read-write) cost, but the highest storage cost.
- Cool Access Tier: This option is better suited for use cases where data will remain stored for at least 30 days and is not accessed frequently. Compared to hot access tiers, this tier offers lower storage costs and higher access costs.
- Archive Access Tier: Archive storage is designed for data that doesn't need to be accessed immediately. This tier offers higher data retrieval costs, and also higher data access latency. It is designed for use cases where data will be stored for more than 180 days and is rarely accessed.

Why Use Blob Storage?

Much of what data consumers do with storage is focused on dealing with unstructured data such as logs, files, images, videos, etc. Using Azure's blob storage is a way to overcome the challenge of having to deploy different database systems for different types of data. Blob storage provides users with strong data consistency, storage and access flexibility that adapts to the user's needs, and it also provides high availability by implementing geo-replication.

Azure Table Storage

Azure Table Storage is a scalable, NoSQL, key-value data storage system that can be used to store large amounts of data in the cloud. This storage offering has a schema less design, and each table has rows that are composed of key-value pairs. Microsoft describes it as an ideal solution for storing structured and non-relational data, covering use cases ranging from storing terabytes of structured data that serves web applications, to storing datasets that do not require complex joins or foreign keys, to accessing data using .NET libraries.

Azure Table Storage Components

Table storage includes:

- A storage account, which contains all your tables.
- Tables, which are composed of collections of "entities."
- Entities, which are sets of properties, similar to database rows. An entity can grow to up to 1MB in size.
- Properties, the most granular element in the list, are composed of name-value pairs. Entities can wrap up to 252 properties to store data, and each entity contains three system properties that define its partition key, row key and timestamp.

Because Azure Storage tables are represented in a tabular format, they can be easily confused with RDBMS tables. However, Azure tables don't have the notion of columns, constraints or 1:1 or 1:* relationships and any of their variations.

Azure Table Storage vs. Azure SQL Database

These two technologies, while very similar, are designed to tackle very different use cases. However, one of the main differences between the two is their capacity. Azure tables can have rows of up to 1MB in size with no more than 255 properties including the three identifying keys:

partition, row and timestamp. Meaning, that when users add the size of all 255 properties, they can't exceed 1MB.
On the other side, Azure SQL databases can have rows up to 2GB in size. Naturally, this would make the user think that Azure SQL databases are a no-brainer when it comes to storing large amounts of data. However, Azure SQL databases can scale up to 150GB only, while the maximum data size for Azure tables is 200TB per table.

Azure Queue Storage

Queues have been around for a long time — their simple FIFO (first in, first out) architecture makes queues a versatile solution for storing messages that do not need to be in a certain order. In simple terms, Azure Queue Storage is a service that allows users to store high volumes of messages, process them asynchronously and consume them when needed while keeping costs down by leveraging a pay-per-use pricing model.
Azure Storage Queues Components
Queue storage is composed of the following elements:
Storage Account, which contains all your storage services.
Queue, composed of a set of messages.
Message, includes any kind of information. For example, a message could be a text message that is supposed to trigger an event on an app, or information about an event that has happened on a website. A message, in any format, can only be up to 64KB in size and the maximum time that a message can remain in a queue is seven days.
However, a single queue can hold up to 200TB worth of messages. Messages can be text strings or arrays of bytes containing any kind of information in formats such as XML, CSV, etc.

Azure File Storage

Azure Files is a shared network file storage service that provides administrators a way to access native SMB file shares in the cloud. These shares — as well as the rest of the Azure storage offerings — can be set as part of the Azure storage account. The Azure File service provides a way for applications running on cloud VMs to share files among them by using standard protocols like WriteFile or ReadFile.

Reference (https://www.dremio.com/data-lake/adls/)

Task 4.

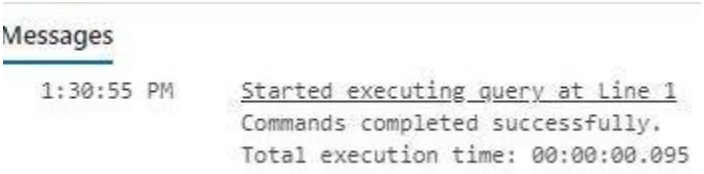
```
CREATE TABLE process()
 processId   int,
 processData VARCHAR(32),

 PRIMARY key (processId)
);

CREATE TABLE fit (
fittype varchar(32),
processId int,
primary key (processId),
foreign KEY (processId) references process(processId)
);

CREATE TABLE paint (
painttype varchar(32),
paintingmethod varchar(32),
processId INT,
primary key (processId),
foreign KEY (processId) references process(processId)
);

CREATE TABLE cut (
cuttingtype varchar(32),
machinetype varchar(32),
processId INT,
primary key (processId),
foreign KEY (processId) references process(processId)
);

create table Job(
jobNomber int,
dateofjobcommenced int,
dateofjobcompleted int,
primary key(jobNomber)
);

create table fitjob(
jobNomber int,
labortime int,
primary key(jobNomber),
foreign key (jobNomber) references Job(jobNomber)
);
```

```
create table painjob(
jobNomber int,
labortime int,
color varchar(15),
amountoftime int,
primary key(jobNomber),
foreign key (jobNomber) references Job(jobNomber)
);

create table cutjob(
jobNomber int,
labortime int,
color varchar(15),
amountoftime int,
typeofmachine varchar(32),
primary key(jobNomber),
foreign key (jobNomber) references Job(jobNomber)
);

CREATE TABLE passthrugh(
   Assemblyid int,
   processId int,
   PRIMARY key(Assemblyid, processId),
   FOREIGN KEY (Assemblyid) REFERENCES Assembly(Assemblyid),
   FOREIGN KEY (processId) REFERENCES process(processId)
   );

CREATE TABLE account (
 accountnumber int,
 dateofaccount int,
 primary key (accountnumber)
 );

create TABLE departmentaccount(
   accountnumber INT,
   sup_cost REAL,
   PRIMARY KEY(accountnumber)
   FOREIGN KEY(accountnumber) REFERENCES account(accountnumber)
);

create TABLE assemblyaccount(
   accountnumber INT,
   sup_cost REAL,
   PRIMARY KEY(accountnumber),
   FOREIGN KEY(accountnumber) REFERENCES account(accountnumber)
```

```
   );
create TABLE processaccount(
   accountnumber INT,
   sup_cost REAL,
   PRIMARY KEY(accountnumber),
   FOREIGN KEY(accountnumber) REFERENCES account(accountnumber)
   );

CREATE TABLE Transaction1(
   transactionnumber ,
   sup_cost REAL,
   PRIMARY KEY(transactionnumber)
);
CREATE TABLE Assembly (
Assemblyid int,
address varchar(32),
dateofordered INT,
AssemblyDeatails varchar(80),
primary key(Assemblyid)
);

CREATE TABLE assign(
   Assemblyid int,
   processId int,
   jobnomber INT
   PRIMARY key(Assemblyid, processId, jobnomber),
   FOREIGN KEY (Assemblyid) REFERENCES Assembly(Assemblyid),
   FOREIGN KEY (processId) REFERENCES process(processId),
   FOREIGN KEY (jobnomber) REFERENCES Job(jobnomber)
   );

CREATE TABLE record(
   transactionnumber int,
   jobnomber INT
   PRIMARY key(transactionnumber, jobnomber)
   );

CREATE TABLE order1(
   Assemblyid int,
   cname int,
   PRIMARY key(Assemblyid, cname),
   FOREIGN KEY (cname) REFERENCES customer(cname),
   FOREIGN KEY (Assemblyid) REFERENCES Assembly(Assemblyid)
   );

CREATE TABLE department(
```

```
    departmentnumber int,
    departmentdata VARCHAR(70),
    PRIMARY KEY(departmentnumber,departmentdata),
);

CREATE TABLE supervise(
    processId int,
    departmentnumber int,
    departmentdata VARCHAR(32),
    PRIMARY key(processId, departmentnumber, departmentdata),
    FOREIGN KEY (processId) REFERENCES process(processId),
    FOREIGN KEY (departmentnumber) REFERENCES department(departmentnumber),

    );
create table record_dept_cost(
    account_num INT,
    dept_num INT,
    PRIMARY KEY(account_num, dept_num),
    FOREIGN KEY(account_num) REFERENCES departmentaccount(accountnumber),
    FOREIGN KEY(dept_num) REFERENCES department(departmentnumber),
);
create table record_assembly_cost(
    account_num INT,
    assembly_id INT,
    PRIMARY KEY(account_num, assembly_id),
    FOREIGN KEY(account_num) REFERENCES assemblyaccount(accountnumber),
    FOREIGN KEY(assembly_id) REFERENCES assembly(Assemblyid),
);
create table record_process_cost(
    account_num INT,
    process_id INT,
    PRIMARY KEY(account_num, process_id),
    FOREIGN KEY(account_num) REFERENCES processaccount(accountnumber),
    FOREIGN KEY(process_id) REFERENCES process(processId),
);
create table record_cost(
    account_num INT,
    process_id INT,
    PRIMARY KEY(account_num, process_id),
    FOREIGN KEY(account_num) REFERENCES account(accountnumber),
    FOREIGN KEY(process_id) REFERENCES process(processId)
);


CREATE TABLE customer(
    cname int,
```

```
    caddress VARCHAR(32),
    category INT,
    PRIMARY KEY(cname),
    CONSTRAINT CHK_category CHECK (category IN ('1','2','3','4','5','6','7','8','9','10'))
);

CREATE TABLE update1(
    transactionnumber INT,
    FOREIGN KEY (transactionnumber) REFERENCES transaction1(transactionnumber),
    FOREIGN KEY (accountnumber) REFERENCES account(accountnumber)
    );
```



Task 5.

5.1.

(Attention: I talked to the TA (Taras Basiuk) and he said that showing the implementation of the queries in the Java program is enough for this exercise. And it does not need to be implemented in azure SQL. However, I implemented most of the queries in SQL, but the full version is at the end of this section, which was implemented in Java.)

Query 1.

```
1   INSERT INTO customer (cname, caddress, category)
2   VALUES ('emad', 'norman', '3');
```

Messages

```
9:58:02 PM    Started executing query at Line 1
              (1 row affected)
              Total execution time: 00:00:00.100
```

Query 2.

```
INSERT INTO process (processId,processData)
VALUES ('999', 'zzz')
INSERT INTO fit (fittype,processId)
VALUES ('besttt', '999')
INSERT INTO paint (painttype,paintingmethod,processId)
VALUES ('ttt', 'nnn', '999')
```

Messages

```
10:40:11 PM    Started executing query at Line 1
               (1 row affected)
               (1 row affected)
               (1 row affected)
               Total execution time: 00:00:00.079
```

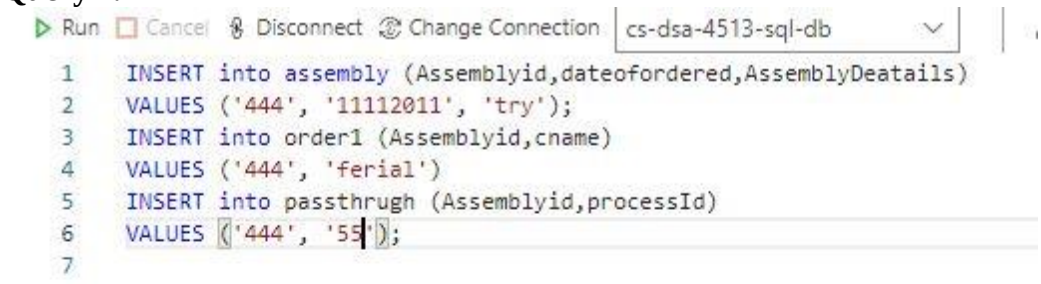
Query 3.

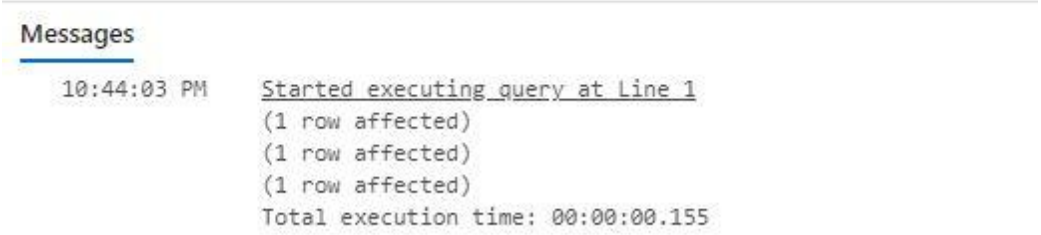
```
INSERT INTO department (departmentnumber, departmentdata)
VALUES ('34', 'photo');
```

Messages

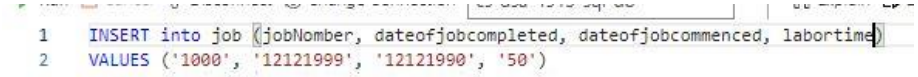
```
10:00:26 PM    Started executing query at Line 1
               (1 row affected)
               Total execution time: 00:00:00.081
```

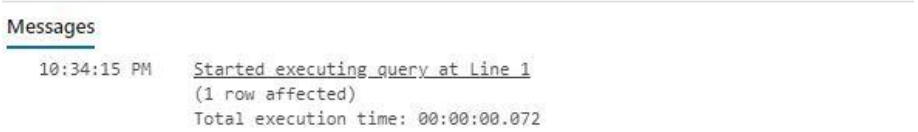
Query 4.



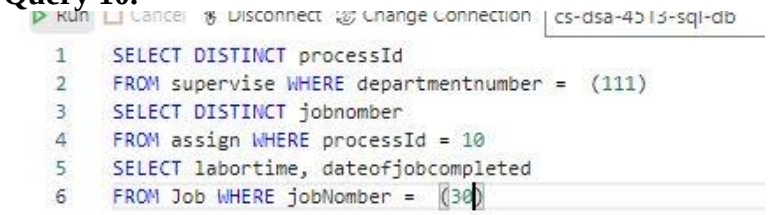


Query 6.





Query 10.



Results Messages

	processId
1	45
2	77
3	99
4	110

	jobnomber

	labortime	dateofjobcompleted
1	45	12122015

Query 11.

```
select supervise.processId, departmentnumber
from Assembly, process, supervise
where supervise.processId = process.processId
and Assembly.Assemblyid = 12
order by dateofordered ASC
```

Results Messages

	processId	departmentnumber
1	10	110
2	12	110
3	20	110
4	33	120
5	45	111
6	55	134
7	60	130
8	77	111
9	99	111
10	110	111

Query 12.

```
select processId from supervise
where departmentnumber = (110)
select jobNomber, Assemblyid from assign
where processId = (12)
select job_type from job
where jobnomber = (110) and dateofjobcompleted = (12122015)
```

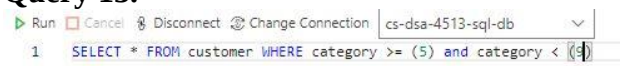
Results Messages

	processId
1	10
2	12
3	20

	jobNomber	Assemblyid

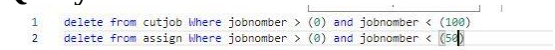
	job_type

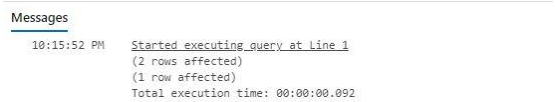
Query 13.



	cname	caddress	category
1	hesam	boston	8
2	jorj	newyork	7
3	mary	alberta	6
4	mohamamd	asas	5
5	sahra	yazd	8

Query 14.





Query 15.

```
1   UPDATE painjob
2   SET color = ('dark')
3   WHERE jobnomber = 60
```

Messages

```
10:13:58 PM    Started executing query at Line 1
               (1 row affected)
               Total execution time: 00:00:00.083
```

All Queries in Java.

```
1  import java.sql.Connection;
2  import java.sql.Statement;
3  import java.sql.Types;
4  import java.util.Scanner;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.DriverManager;
8  import java.sql.PreparedStatement;
9
10 import java.io.*;
11
12 public class Final1 {
13
```

```java
import java.sql.Connection;

public class Final1 {

    // Database credentials
    final static String HOSTNAME = "shah0037-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "shah0037";
    final static String PASSWORD = "Lenovoe460";

    // Database connection string
    final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
            HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Query templates
    final static String QUERY_TEMPLATE_1 = "INSERT INTO customer " +
                                           "VALUES (?, ?, ?);";

    final static String QUERY_TEMPLATE_2 = "INSERT INTO department " +
                                            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3 = "INSERT INTO process " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3_1 = "INSERT INTO fit " +
                                             "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3_2 = "INSERT INTO paint " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_3_3 = "INSERT INTO cut " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_3_4 ="INSERT into supervise " +
                                            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_4_1 ="INSERT into assembly " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_4_2 ="INSERT into order1 " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_4_3 ="INSERT into passthrugh " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_1 ="INSERT into account " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_2 ="INSERT into departmentaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_3 ="INSERT into assemblyaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_4 ="INSERT into processaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_5 ="INSERT into record_dept_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_6 ="INSERT into record_assembly_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_7 ="INSERT into record_process_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_6_1 ="INSERT into job  " +
            "VALUES (?, ?, ?, ?, ?);";
    final static String QUERY_TEMPLATE_6_2 ="INSERT into assign  " +
```

```
62    final static String QUERY_TEMPLATE_6_2 ="INSERT into assign  " +
63            "VALUES (?, ?, ?);";
64
65    final static String QUERY_TEMPLATE_7_1 ="UPDATE job " +
66            "SET dateofjobcompleted = (?), labortime = (?), job_type = (?)" +
67            "Where jobnomber = (?);";
68    final static String QUERY_TEMPLATE_7_2 ="INSERT into cutjob  " +
69            "VALUES (?, ?, ?, ?, ?);";
70    final static String QUERY_TEMPLATE_7_3 ="INSERT into painjob  " +
71            "VALUES (?, ?, ?, ?);";
72    final static String QUERY_TEMPLATE_7_4 ="INSERT into fitjob  " +
73            "VALUES (?, ?);";
74    final static String QUERY_TEMPLATE_8_1 ="INSERT into transaction1  " +
75            "VALUES (?, ?);";
76    final static String QUERY_TEMPLATE_8_2 ="INSERT into record  " +
77            "VALUES (?, ?);";
78    final static String QUERY_TEMPLATE_8_3 ="SELECT Assemblyid, processId from assign  " +
79            " WHERE jobnomber = (?);";
80    final static String QUERY_TEMPLATE_8_4 ="SELECT departmentnumber FROM supervise " +
81            "WHERE processId =(?);";
82    final static String QUERY_TEMPLATE_8_5 ="SELECT account_num FROM record_dept_cost " +
83            "WHERE dept_num =(?);";
84    final static String QUERY_TEMPLATE_8_6 ="SELECT account_num FROM record_assembly_cost " +
85            "WHERE assembly_id =(?);";
86    final static String QUERY_TEMPLATE_8_7 ="SELECT account_num FROM record_process_cost " +
87            "WHERE process_id =(?);";
88    final static String QUERY_TEMPLATE_8_8 ="SELECT sup_cost FROM departmentaccount " +
89            "WHERE accountnumber =(?);";
90    final static String QUERY_TEMPLATE_8_9 ="SELECT sup_cost FROM assemblyaccount " +
91            "WHERE accountnumber =(?);";
92    final static String QUERY_TEMPLATE_8_10 ="SELECT sup_cost FROM processaccount " +
93            "WHERE accountnumber =(?);";
94    final static String QUERY_TEMPLATE_8_11 = "UPDATE departmentaccount " +
95            "SET sup_cost = (?)" +
96            "Where accountnumber = (?);";
97    final static String QUERY_TEMPLATE_8_12 = "UPDATE assemblyaccount " +
98            "SET sup_cost = (?)" +
99            "Where accountnumber = (?);";
100   final static String QUERY_TEMPLATE_8_13 = "UPDATE processaccount " +
101           "SET sup_cost = (?)" +
102           "Where accountnumber = (?);";
103
104   final static String QUERY_TEMPLATE_10_1 ="SELECT DISTINCT processId " +
105           "FROM supervise WHERE departmentnumber =  (?); ";
106   final static String QUERY_TEMPLATE_10_2 ="SELECT DISTINCT jobnomber " +
107           "FROM assign WHERE processId =  (?); ";
108   final static String QUERY_TEMPLATE_10_3 ="SELECT labortime, dateofjobcompleted " +
109           "FROM Job WHERE jobNomber =  (?); ";
110   final static String QUERY_TEMPLATE_11 = "select supervise.processId, departmentnumber "
111          + "from Assembly, process, supervise "
112          + "where supervise.processId = process.processId "
113          + "and Assembly.Assemblyid = (?) "
114          + "order by dateofordered ASC";
```

```
            + "order by dateofordered ASC";
    final static String QUERY_TEMPLATE_12_1 = "select processId from supervise "+
            "where departmentnumber = (?)";
    final static String QUERY_TEMPLATE_12_2 = "select jobNomber, Assemblyid from assign " +
            "where processId = (?)";
    final static String QUERY_TEMPLATE_12_3 = "select job_type from job "+
            "where jobnomber = (?) and dateofjobcompleted = (?)";
    final static String QUERY_TEMPLATE_13 = "SELECT * FROM customer WHERE category >= (?) and category < (?) "   ;

    final static String QUERY_TEMPLATE_14_1 = "delete from cutjob Where jobnomber > (?) and jobnomber < (?) " ;
    final static String QUERY_TEMPLATE_14_2 = "delete from assign Where jobnomber > (?) and jobnomber < (?) " ;
    final static String QUERY_TEMPLATE_14_3 = "delete from job Where jobnomber > (?) and jobnomber < (?) " ;
    final static String QUERY_TEMPLATE_15 ="UPDATE painjob " +
            "SET color = (?)" +
            "Where jobnomber = (?);";
    // User input prompt//
    final static String PROMPT =
             "\nPlease select one of the options below: \n" +
                        "1) Enter new customer; \n" +
                        "2) Enter a new department; \n" +
                        "3) Enter a new Process ; \n" +
                        "4) Enter a new Assembly: \n" +
                        "5) Enter a new Account Number:\n" +
                        "6) Enter a new Job:\n"+
                        "7) At the completion of a job, enter the date it completed: \n " +
                        "8) Enter a transaction-no and its sup-cost and update all the costs: \n" +
                        "9) Retrieve the total cost incurred on an assembly-id \n " +
                        "10)Retrieve total labor time within a Department for a given day: \n " +
                        "11)Retrieve the process through which a given assembly-id has passed: \n" +
                        "12)Retrieve the jobs completed during 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: \n"+
                        "17)Export: Retrieve the customers \n" +
                        "18) Quit!";

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

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

        final Scanner sc = new Scanner(System.in); // Scanner is used to collect the user input
        String option = ""; // Initialize user option selection as nothing
        while (!option.equals("3")) { // As user for options until option 3 is selected
            System.out.println(PROMPT); // Print the available options
            option = sc.next(); // Read in the user option selection

            switch (option) { // Switch between different options
                case "1": // Insert a new student option
                    // Collect the new student data from the user
                    System.out.println("Please enter customer cname:");
                    final String cname = sc.next(); // Read in the user input of student ID
```

Task 5.2.

```java
import java.sql.Connection;
import java.sql.Statement;
import java.sql.Types;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import java.io.*;

public class Final1 {

```

```java
import java.sql.Connection;

public class Final1 {

    // Database credentials
    final static String HOSTNAME = "shah0037-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "shah0037";
    final static String PASSWORD = "Lenovoe460";

    // Database connection string
    final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
            HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Query templates
    final static String QUERY_TEMPLATE_1 = "INSERT INTO customer " +
                                           "VALUES (?, ?, ?);";

    final static String QUERY_TEMPLATE_2 = "INSERT INTO department " +
                                            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3 = "INSERT INTO process " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3_1 = "INSERT INTO fit " +
                                             "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_3_2 = "INSERT INTO paint " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_3_3 = "INSERT INTO cut " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_3_4 ="INSERT into supervise " +
                                            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_4_1 ="INSERT into assembly " +
            "VALUES (?, ?, ?);";
    final static String QUERY_TEMPLATE_4_2 ="INSERT into order1 " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_4_3 ="INSERT into passthrugh " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_1 ="INSERT into account " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_2 ="INSERT into departmentaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_3 ="INSERT into assemblyaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_4 ="INSERT into processaccount " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_5 ="INSERT into record_dept_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_6 ="INSERT into record_assembly_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_5_7 ="INSERT into record_process_cost " +
            "VALUES (?, ?);";
    final static String QUERY_TEMPLATE_6_1 ="INSERT into job  " +
            "VALUES (?, ?, ?, ?);";
    final static String QUERY_TEMPLATE_6_2 ="INSERT into assign  " +
```

```
62    final static String QUERY_TEMPLATE_6_2 ="INSERT into assign  " +
63            "VALUES (?, ?, ?);";
64
65    final static String QUERY_TEMPLATE_7_1 ="UPDATE job " +
66            "SET dateofjobcompleted = (?), labortime = (?), job_type = (?)" +
67            "Where jobnomber = (?);";
68    final static String QUERY_TEMPLATE_7_2 ="INSERT into cutjob  " +
69            "VALUES (?, ?, ?, ?, ?);";
70    final static String QUERY_TEMPLATE_7_3 ="INSERT into painjob  " +
71            "VALUES (?, ?, ?, ?);";
72    final static String QUERY_TEMPLATE_7_4 ="INSERT into fitjob  " +
73            "VALUES (?, ?);";
74    final static String QUERY_TEMPLATE_8_1 ="INSERT into transaction1  " +
75            "VALUES (?, ?);";
76    final static String QUERY_TEMPLATE_8_2 ="INSERT into record  " +
77            "VALUES (?, ?);";
78    final static String QUERY_TEMPLATE_8_3 ="SELECT Assemblyid, processId from assign  " +
79            " WHERE jobnomber = (?);";
80    final static String QUERY_TEMPLATE_8_4 ="SELECT departmentnumber FROM supervise " +
81            "WHERE processId =(?);";
82    final static String QUERY_TEMPLATE_8_5 ="SELECT account_num FROM record_dept_cost " +
83            "WHERE dept_num =(?);";
84    final static String QUERY_TEMPLATE_8_6 ="SELECT account_num FROM record_assembly_cost " +
85            "WHERE assembly_id =(?);";
86    final static String QUERY_TEMPLATE_8_7 ="SELECT account_num FROM record_process_cost " +
87            "WHERE process_id =(?);";
88    final static String QUERY_TEMPLATE_8_8 ="SELECT sup_cost FROM departmentaccount " +
89            "WHERE accountnumber =(?);";
90    final static String QUERY_TEMPLATE_8_9 ="SELECT sup_cost FROM assemblyaccount " +
91            "WHERE accountnumber =(?);";
92    final static String QUERY_TEMPLATE_8_10 ="SELECT sup_cost FROM processaccount " +
93            "WHERE accountnumber =(?);";
94    final static String QUERY_TEMPLATE_8_11 = "UPDATE departmentaccount " +
95            "SET sup_cost = (?)" +
96            "Where accountnumber = (?);";
97    final static String QUERY_TEMPLATE_8_12 = "UPDATE assemblyaccount " +
98            "SET sup_cost = (?)" +
99            "Where accountnumber = (?);";
100   final static String QUERY_TEMPLATE_8_13 = "UPDATE processaccount " +
101           "SET sup_cost = (?)" +
102           "Where accountnumber = (?);";
103
104   final static String QUERY_TEMPLATE_10_1 ="SELECT DISTINCT processId " +
105           "FROM supervise WHERE departmentnumber =  (?); ";
106   final static String QUERY_TEMPLATE_10_2 ="SELECT DISTINCT jobnomber " +
107           "FROM assign WHERE processId =  (?); ";
108   final static String QUERY_TEMPLATE_10_3 ="SELECT labortime, dateofjobcompleted " +
109           "FROM Job WHERE jobNomber =  (?); ";
110   final static String QUERY_TEMPLATE_11 = "select supervise.processId, departmentnumber "
111          + "from Assembly, process, supervise "
112          + "where supervise.processId = process.processId "
113          + "and Assembly.Assemblyid = (?) "
114          + "order by dateofordered ASC";
```

```
114             + "order by dateofordered ASC";
115     final static String QUERY_TEMPLATE_12_1 = "select processId from supervise "+
116             "where departmentnumber = (?)";
117     final static String QUERY_TEMPLATE_12_2 = "select jobNomber, Assemblyid from assign " +
118             "where processId = (?)";
119     final static String QUERY_TEMPLATE_12_3 = "select job_type from job "+
120             "where jobnomber = (?) and dateofjobcompleted = (?)";
121     final static String QUERY_TEMPLATE_13 = "SELECT * FROM customer WHERE category >= (?) and category < (?) "   ;
122
123     final static String QUERY_TEMPLATE_14_1 = "delete from cutjob Where jobnomber > (?) and jobnomber < (?) " ;
124     final static String QUERY_TEMPLATE_14_2 = "delete from assign Where jobnomber > (?) and jobnomber < (?) " ;
125     final static String QUERY_TEMPLATE_14_3 = "delete from job Where jobnomber > (?) and jobnomber < (?) " ;
126     final static String QUERY_TEMPLATE_15 ="UPDATE painjob " +
127             "SET color = (?)" +
128             "Where jobnomber = (?);";
129     // User input prompt//
130     final static String PROMPT =
131              "\nPlease select one of the options below: \n" +
132                         "1) Enter new customer; \n" +
133                         "2) Enter a new department; \n" +
134                         "3) Enter a new Process ; \n" +
135                         "4) Enter a new Assembly: \n" +
136                         "5) Enter a new Account Number:\n" +
137                         "6) Enter a new Job:\n"+
138                         "7) At the completion of a job, enter the date it completed: \n " +
139                         "8) Enter a transaction-no and its sup-cost and update all the costs: \n" +
140                         "9) Retrieve the total cost incurred on an assembly-id \n " +
141                         "10)Retrieve total labor time within a Department for a given day: \n " +
142                         "11)Retrieve the process through which a given assembly-id has passed: \n" +
143                         "12)Retrieve the jobs completed during given date in a given department: \n"+
144                         "13)Retrieve the customers (in name order) whose category is in a given range: \n" +
145                         "14)Delete all cut-jobs whose job-no is in a given range: \n"+
146                         "15)Change the color of a given paint job\n" +
147                         "16)Import: enter new customers from a data file: \n"+
148                         "17)Export: Retrieve the customers \n" +
149                         "18) Quit!";
150
151     public static void main(String[] args) throws SQLException {
152
153         System.out.println("WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM");
154
155         final Scanner sc = new Scanner(System.in); // Scanner is used to collect the user input
156         String option = ""; // Initialize user option selection as nothing
157         while (!option.equals("3")) { // As user for options until option 3 is selected
158             System.out.println(PROMPT); // Print the available options
159             option = sc.next(); // Read in the user option selection
160
161             switch (option) { // Switch between different options
162                 case "1": // Insert a new student option
163                     // Collect the new student data from the user
164                     System.out.println("Please enter customer cname:");
165                     final String cname = sc.next(); // Read in the user input of student ID
166
```

```
166
167                 sc.nextLine();
168                 System.out.println("Please enter customer address:");
169                 // Preceding nextInt, nextFloar, etc. do not consume new line characters from the user input.
170                 // We call nextLine to consume that newline character, so that subsequent nextLine doesn't return nothing.
171                 //sc.nextLine();
172                 final String caddress = sc.nextLine(); // Read in user input of student First Name (white-spaces allowed).
173
174                 System.out.println("Please enter customer category:");
175                 // No need to call nextLine extra time here, because the preceding nextLine consumed the newline character.
176                 final float category = sc.nextFloat(); // Read in user input of student Last Name (white-spaces allowed).
177
178                 System.out.println("Connecting to the database...");
179                 // Get a database connection and prepare a query statement
180                 try (final Connection connection = DriverManager.getConnection(URL)) {
181                     try (
182                         final PreparedStatement statement = connection.prepareStatement(QUERY_TEMPLATE_1)) {
183                         // Populate the query template with the data collected from the user
184                         System.out.print(cname);
185                         statement.setString(1, cname);
186                         statement.setString(2, caddress);
187                         statement.setFloat(3, category);
188
189                         System.out.println("Dispatching the query...");
190                         // Actually execute the populated query
191                         final int rows_inserted = statement.executeUpdate();
192                         System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
193                     }
194                 }
195                 break;
196             case "2":
197                 System.out.println("Please enter department number");
198                 final int departmentnumber = sc.nextInt(); // Read in the user input of student ID
199
200                 //sc.nextLine();
201                 System.out.println("Please enter department data:");
202                 // Preceding nextInt, nextFloar, etc. do not consume new line characters from the user input.
203                 // We call nextLine to consume that newline character, so that subsequent nextLine doesn't return nothing.
204                 sc.nextLine();
205                 final String departmentdata = sc.nextLine(); // Read in user input of student First Name (white-spaces allowed).
206
207
208                 System.out.println("Connecting to the database...");
209                 // Get a database connection and prepare a query statement
210                 try (final Connection connection = DriverManager.getConnection(URL)) {
211                     try (
212                         final PreparedStatement statement = connection.prepareStatement(QUERY_TEMPLATE_2)) {
213                         // Populate the query template with the data collected from the user
214
215                         statement.setInt(1, departmentnumber);
216                         statement.setString(2, departmentdata);
217
218
```

```
218
219                         System.out.println("Dispatching the query...");
220                         // Actually execute the populated query
221                         final int rows_inserted = statement.executeUpdate();
222                         System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
223                     }
224                 }
225
226                 break;
227             case "3": // Insert a new process option
228                 // Collect the new process data from the user
229                 System.out.println("Please enter process Id:");
230                 final int processId = sc.nextInt(); // Read in the user input of student ID
231                 sc.nextLine();
232                 System.out.println("Please enter the process Data:");
233                 final String processData = sc.nextLine();
234
235                 sc.nextLine();
236                 System.out.println("Please enter the Supervising Department ID:" );
237                 final int deptnum = sc.nextInt();
238                 sc.nextLine();
239                 System.out.println("Please enter the Department data:");
240                 final String deptData = sc.nextLine();
241
242                 System.out.println("Please enter : 1. For Fit , 2. For Paint  , 3. for Cut Process:");
243                 final int processmodel = sc.nextInt();
244                 sc.nextLine();
245                 if (processmodel == 1) {
246                     //Fit Process
247                     System.out.println("Please enter Fit type:");
248                     final String fittype = sc.nextLine();
249                     sc.nextLine();
250                     System.out.println("Connecting to the database...");
251                     // Get a database connection and prepare a query statement
252                     try (final Connection connection = DriverManager.getConnection(URL)) {
253                         try (
254                                 final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_3)) {
255                                 // Populate the query template with the data collected from the user
256
257                                 statement1.setInt(1, processId);
258                                 statement1.setString(2, processData);
259
260                                 System.out.println("Dispatching the query...");
261                                 // Actually execute the populated query
262                                 final int rows_inserted = statement1.executeUpdate();
263                                 System.out.println(String.format("Done. %d rows inserted in \"process\" table.", rows_inserted));
264
265                             }
266                         try (
267                             final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_3_1)) {
268                             // Populate the query template with the data collected from the user
269
270                             statement2.setInt(2, processId);
```

```
                    statement2.setInt(2, processId);
                    statement2.setString(1, fittype);

                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final int rows_inserted = statement2.executeUpdate();
                    System.out.println(String.format("Done. %d rows inserted.", rows_inserted));

                }

            }


        }
        else if (processmodel == 2) {
            //information for Paint Process
            System.out.println("Please enter Paint type:");
            final String painttype = sc.nextLine();
            //sc.nextLine();
            System.out.println("Please enter Paint method:");
            final String paintingmethod = sc.nextLine();
            //sc.nextLine();

            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {

                 try (
                         final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_3)) {
                         // Populate the query template with the data collected from the user

                         statement1.setInt(1, processId);
                         statement1.setString(2, processData);

                         System.out.println("Dispatching the query...");
                         // Actually execute the populated query
                         final int rows_inserted = statement1.executeUpdate();
                         System.out.println(String.format("Done. %d rows inserted in the process table.", rows_inserted));

                     }
                try (
                    final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_3_2)) {
                    // Populate the query template with the data collected from the user

                    statement2.setInt(3, processId);
                    statement2.setString(1, painttype);
                    statement2.setString(2, paintingmethod);
                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final int rows_inserted = statement2.executeUpdate();
                    System.out.println("Executing Statement 3(2)...");
                    System.out.println(String.format("Done. %d rows inserted.", rows_inserted));

```

```
322
323
324                         }
325
326                     }
327
328                 }
329                 else {
330                     // information for Cutting
331                     System.out.println("Please enter the Cutting Type:");
332                     final String cuttingType = sc.nextLine();
333                     System.out.println("Please enter the Machine Type:");
334                     final String machineType = sc.nextLine();
335                     System.out.println("Connecting to the database...");
336                     // Get a database connection and prepare a query statement
337                     try (final Connection connection = DriverManager.getConnection(URL)) {
338
339                         try (
340                                 final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_3)) {
341                                 // Populate the query template with the data collected from the user
342
343                                 statement1.setInt(1, processId);
344                                 statement1.setString(2, processData);
345
346                                 System.out.println("Dispatching the query...");
347                                 // Actually execute the populated query
348                                 final int rows_inserted = statement1.executeUpdate();
349                                 System.out.println(String.format("Done. %d rows inserted in process table.", rows_inserted));
350
351                             }
352
353                         try (
354                             final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_3_3)) {
355                             // Populate the query template with the data collected from the user
356                             statement3.setInt(3, processId);
357                             statement3.setString(1, cuttingType);
358                             statement3.setString(2, machineType);
359
360                             System.out.println("Dispatching the query...");
361                             // Actually execute the populated query
362                             final int rows_inserted = statement3.executeUpdate();
363                             System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
364
365                         }
366
367                     }
368
369                 }
370                 try (final Connection connection = DriverManager.getConnection(URL)) {
371                     try (
372                         final PreparedStatement statement4 = connection.prepareStatement(QUERY_TEMPLATE_3_4)) {
373                         // Populate the query template with the data collected from the user
374
```

```
374
375                         statement4.setInt(1, processId);
376                         statement4.setInt(2, deptnum);
377                         statement4.setString(3, deptData);
378
379                         System.out.println("Dispatching the query 3(4)...");
380                         // Actually execute the populated query
381                         final int rows_inserted = statement4.executeUpdate();
382                         System.out.println(String.format("Done. %d rows inserted in supervise table.", rows_inserted));
383
384                     }
385                 }
386                 break;
387             case "4":
388                 System.out.println(" enter a new Assembly ID:");
389                 final int assemblyid = sc.nextInt();
390                 sc.nextLine();
391                 System.out.println("Please enter the Date of ordered of assembly:");
392                 final int dateof_ordered = sc.nextInt();
393                 sc.nextLine();
394                 System.out.println("Please enter the Assembly details:");
395                 final String assemblyDetails = sc.nextLine();
396                 System.out.println("Please enter the associated Customer Name:");
397                 final String assembly_customername = sc.nextLine();
398                 System.out.println("Please enter the number of associated process IDs with this assembly");
399                 final int numassemblies = sc.nextInt();
400                 sc.nextLine();
401                 int[] assemblyrep = new int[numassemblies];
402                 for (int j = 0; j < numassemblies; j++)
403                 {
404                     System.out.println(" enter %dth process ID:");
405                     assemblyrep[j] = sc.nextInt();
406                     sc.nextLine();
407                 }
408
409                 try (final Connection connection = DriverManager.getConnection(URL)) {
410                     try (
411                             final PreparedStatement statement4 = connection.prepareStatement(QUERY_TEMPLATE_4_1)) {
412                             // Populate the query template with the data collected from the user
413
414                             statement4.setInt(1, assemblyid);
415                             statement4.setInt(2, dateof_ordered);
416                             statement4.setString(3, assemblyDetails);
417
418                             System.out.println("Dispatching the query 4(1)...");
419                             // Actually execute the populated query
420                             final int rows_inserted = statement4.executeUpdate();
421                             System.out.println(String.format("Done. %d rows inserted in Assembly table.", rows_inserted));
422
423                         }
424                     try (
425                             final PreparedStatement statement5 = connection.prepareStatement(QUERY_TEMPLATE_4_2)) {
426                             // Populate the query template with the data collected from the user
```

```
426
427                             statement5.setInt(1, assemblyid);
428                             statement5.setString(2, assembly_customername);
429
430
431                             System.out.println("Dispatching the query 4(2)...");
432                             // Actually execute the populated query
433                             final int rows_inserted = statement5.executeUpdate();
434                             System.out.println(String.format("Done. %d rows inserted in order1 table.", rows_inserted));
435
436                         }
437                     try (
438                             final PreparedStatement statement6 = connection.prepareStatement(QUERY_TEMPLATE_4_3)) {
439                             // Populate the query template with the data collected from the user
440
441                             for (int i = 0; i< numassemblies; i++)
442                             {
443                                 statement6.setInt(1, assemblyid);
444                                 statement6.setInt(2, assemblyrep[i]);
445                                 System.out.println("Dispatching the query 4(3)...");
446                                 // Actually execute the populated query
447                                 final int rows_inserted = statement6.executeUpdate();
448                                 System.out.println(String.format("Done. %d rows inserted in passthrugh table.", rows_inserted));
449                             }
450
451                         }
452                 }
453                 break;
454             case "5":
455                 System.out.println(" enter a new Account number");
456                 final int accountnumb = sc.nextInt(); // Read in the user input of student ID
457                 sc.nextLine();
458                 System.out.println(" enter the date of account commenced please:");
459                 final int accountdate = sc.nextInt();
460
461
462                 System.out.println(" enter the account type (1,2,3): 1.Departmentt Account, 2.Assembly Account, 3.Process Account:");
463                 final int Type = sc.nextInt();
464                 sc.nextLine();
465                 if (Type == 1) {
466                     System.out.println(" enter the department number which is associated with the account:");
467                     final int deptNo = sc.nextInt();
468                     sc.nextLine();
469                     System.out.println("Connecting to the database...");
470                     // Get a database connection and prepare a query statement
471                     try (final Connection connection = DriverManager.getConnection(URL)) {
472                         try (
473                                 final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_5_1)) {
474                                 // Populate the query template with the data collected from the user
475
476                                 statement1.setInt(1, accountnumb);
477                                 statement1.setInt(2, accountdate);
478
```

```
478
479                                 System.out.println("Dispatching the query 5(1)...");
480                                 // Actually execute the populated query
481                                 final int rows_inserted = statement1.executeUpdate();
482                                 System.out.println(String.format("Done. %d rows inserted in account table.", rows_inserted));
483
484                             }
485                         try (
486                             final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_5_2)) {
487                             // Populate the query template with the data collected from the user
488
489                             statement2.setInt(1, accountnumb);
490                             statement2.setInt(2, 0);
491
492                             System.out.println("Dispatching the query 5(2)...");
493                             // Actually execute the populated query
494                             final int rows_inserted = statement2.executeUpdate();
495                             System.out.println(String.format("Done. %d rows inserted in department account table .", rows_inserted));
496
497                         }
498                         try (
499                                 final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_5_5)) {
500                                 // Populate the query template with the data collected from the user
501
502                                 statement3.setInt(1, accountnumb );
503                                 statement3.setInt(2, deptNo);
504
505                                 System.out.println("Dispatching the query 5(5)...");
506                                 // Actually execute the populated query
507                                 final int rows_inserted = statement3.executeUpdate();
508                                 System.out.println(String.format("Done. %d rows inserted in record dept cost table .", rows_inserted));
509
510                             }
511
512                     }
513
514
515                 }
516                 else if (Type == 2) {
517                     //information for Paint Process
518                     System.out.println("Please enter the associated Assembly ID: ");
519                     final int AssemblyID = sc.nextInt();
520                     sc.nextLine();
521
522                     System.out.println("Connecting to the database...");
523                     // Get a database connection and prepare a query statement
524                     try (final Connection connection = DriverManager.getConnection(URL)) {
525
526                          try (
527                                  final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_5_1)) {
528                                  // Populate the query template with the data collected from the user
529
530                              statement1.setInt(1, accountnumb);
```

```
                        statement1.setInt(1, accountnumb);
                        statement1.setInt(2, accountdate);

                        System.out.println("Dispatching the query 5(a)...");
                        // Actually execute the populated query
                        final int rows_inserted = statement1.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted in account table.", rows_inserted));

                        }
                    try (
                        final PreparedStatement statement9 = connection.prepareStatement(QUERY_TEMPLATE_5_3)) {
                        // Populate the query template with the data collected from the user

                        statement9.setInt(1, accountnumb);
                        statement9.setInt(2, 0);
                        System.out.println("Dispatching the query 5(3)...");
                        // Actually execute the populated query
                        final int rows_inserted = statement9.executeUpdate();
                        System.out.println("Executing Statement 5(c)...");
                        System.out.println(String.format("Done. %d rows inserted in assemblyaccount table.", rows_inserted));


                    }
                    try (
                            final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_5_6)) {
                            // Populate the query template with the data collected from the user
                            //System.out.print(cname);
                            statement3.setInt(1, accountnumb);
                            statement3.setInt(2, AssemblyID);
                            System.out.println("Dispatching the query 5(6)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement3.executeUpdate();
                            System.out.println("Executing Statement 5(6)...");
                            System.out.println(String.format("Done. %d rows inserted in record assembly cost table.", rows_inserted));

                        }

                }

            }
            else {
                // information for Cut Process
                System.out.println(" enter the associated process ID:");
                final int processID = sc.nextInt();
                sc.nextLine();

                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {

                    try (
                            final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_5_1)) {
                            // Populate the query template with the data collected from the user
```

```
582                                 // Populate the query template with the data collected from the user
583                                 //System.out.print(cname);
584                                 statement1.setInt(1, accountnumb);
585                                 statement1.setInt(2, accountdate);
586 
587                                 System.out.println("Dispatching the query 5(1)...");
588                                 // Actually execute the populated query
589                                 final int rows_inserted = statement1.executeUpdate();
590                                 System.out.println(String.format("Done. %d rows inserted in account table.", rows_inserted));
591 
592                             }
593 
594                         try (
595                             final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_5_4)) {
596                             // Populate the query template with the data collected from the user
597                             //System.out.print(cname);
598                             statement3.setInt(1, accountnumb);
599                             statement3.setInt(2, 0);
600                             System.out.println("Dispatching the query 5(4)...");
601                             // Actually execute the populated query
602                             final int rows_inserted = statement3.executeUpdate();
603                             System.out.println(String.format("Done. %d rows inserted in process account table..", rows_inserted));
604 
605                         }
606 
607                         try (
608                                 final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_5_7)) {
609                                 // Populate the query template with the data collected from the user
610 
611                                 statement3.setInt(1, accountnumb);
612                                 statement3.setInt(2, processID);
613                                 System.out.println("Dispatching the query 5(7)...");
614                                 // Actually execute the populated query
615                                 final int rows_inserted = statement3.executeUpdate();
616                                 System.out.println(String.format("Done. %d rows inserted in record process cost table..", rows_inserted));
617 
618                             }
619 
620                     }
621 
622                 }
623                 break;
624             case "6":
625                 System.out.println("Please enter new Job Number:");
626                 final int jobNum = sc.nextInt();
627                 sc.nextLine();
628 
629                 System.out.println("Please enter the date of job commenced:");
630                 final int datejobCommenced = sc.nextInt();
631                 sc.nextLine();
632 
633                 System.out.println("Please enter the Process ID of to this job:");
634                 final int processID = sc.nextInt();
```

```
                final int processID = sc.nextInt();
                sc.nextLine();

                System.out.println(" enter the Assembly ID of process for this job:");
                final int assemblyID = sc.nextInt();
                sc.nextLine();

                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                            final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_6_1)) {
                            // Populate the query template with the data collected from the user

                            statement1.setInt(1, jobNum);
                            statement1.setInt(3, datejobCommenced);
                            statement1.setNull(2, Types.NULL);
                            statement1.setNull(4, Types.NULL);
                            statement1.setNull(5, Types.NULL);
                            System.out.println("Dispatching the query 6(1)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement1.executeUpdate();
                            System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));

                        }
                    try (
                            final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_6_2)) {
                            // Populate the query template with the data collected from the user

                            statement2.setInt(1, assemblyID);
                            statement2.setInt(3, jobNum);
                            statement2.setInt(2, processID);
                            System.out.println("Dispatching the query 6(2)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement2.executeUpdate();
                            System.out.println(String.format("Done. %d rows inserted in assign table..", rows_inserted));

                        }

                }
                break;
            case "7":
                System.out.println("Please enter the Job Number which is in case 6:");
                final int jobnum = sc.nextInt();
                sc.nextLine();

                System.out.println("Please enter the date the job completed:");
                final int datejobCompleted = sc.nextInt();
                sc.nextLine();

                System.out.println("Please enter type of Job(1,2,3): 1.Cut Job, 2. Paint Job , 3.Fit Job");
                final int type = sc.nextInt();
                sc.nextLine();

                if (type == 1)
```

```
686                 if (type == 1)
687                 {
688                     System.out.println("Please enter the type of Machine :");
689                     final String machinetype = sc.nextLine();
690                     System.out.println(" enter the amount of time in hours:");
691                     final int time = sc.nextInt();
692                     sc.nextLine();
693                     System.out.println("Please enter the material :");
694                     final String material = sc.nextLine();
695                     System.out.println("Please enter the labor time  in hours:");
696                     final int laborTime = sc.nextInt();
697                     sc.nextLine();
698
699
700                     System.out.println("Connecting to the database...");
701                     // Get a database connection and prepare a query statement
702                     try (final Connection connection = DriverManager.getConnection(URL)) {
703                         try (
704                                 final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_7_1)) {
705                                 // Populate the query template with the data collected from the user
706                                 //System.out.print(cname);
707                                 statement1.setInt(1, datejobCompleted);
708                                 statement1.setInt(2, laborTime);
709                                 statement1.setInt(3, type);
710                                 statement1.setInt(4, jobnum);
711                                 System.out.println("Dispatching the query 7(1)...");
712                                 // Actually execute the populated query
713                                 final int rows_inserted = statement1.executeUpdate();
714                                 System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));
715
716                             }
717
718                         try (
719                                 final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_2)) {
720                                 // Populate the query template with the data collected from the user
721                                 //System.out.print(cname);
722                                 statement3.setString(1, machinetype);
723                                 statement3.setString(2, material);
724                                 statement3.setInt(3, jobnum);
725                                 statement3.setInt(4, time);
726                                 statement3.setInt(5, laborTime);
727                                 System.out.println("Dispatching the query 6(3)...");
728                                 // Actually execute the populated query
729                                 final int rows_inserted = statement3.executeUpdate();
730                                 System.out.println(String.format("Done. %d rows inserted in cutjob table..", rows_inserted));
731
732                             }
733                     }
734                 }
735                 else if (type == 2)
736                 {
737                     System.out.println("Please enter the color:");
738                     final String color = sc.nextLine();
```

```
                    final String color = sc.nextLine();

                    System.out.println("Please enter the volume of paint :");
                    final int volume = sc.nextInt();
                    sc.nextLine();

                    System.out.println("Please enter the labor time in hours:");
                    final int laborTime = sc.nextInt();
                    sc.nextLine();

                    System.out.println("Connecting to the database...");
                    // Get a database connection and prepare a query statement
                     try (final Connection connection = DriverManager.getConnection(URL)) {
                        try (
                                 final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_7_1)) {
                                 // Populate the query template with the data collected from the user
                                //System.out.print(cname);
                                    statement1.setInt(1, datejobCompleted);
                                    statement1.setInt(2, laborTime);
                                    statement1.setInt(3, type);
                                    statement1.setInt(4, jobnum);
                                 System.out.println("Dispatching the query 7(1)...");
                                 // Actually execute the populated query
                                 final int rows_inserted = statement1.executeUpdate();
                                 System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));

                             }

                        try (
                                 final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_3)) {
                                 // Populate the query template with the data collected from the user
                                //System.out.print(cname);
                                 statement3.setString(1, color);
                                 statement3.setInt(2, volume);
                                 statement3.setInt(3, jobnum);
                                 statement3.setInt(4, laborTime);

                                 System.out.println("Dispatching the query 7(3)...");
                                 // Actually execute the populated query
                                 final int rows_inserted = statement3.executeUpdate();
                                 System.out.println(String.format("Done. %d rows inserted in painjob table..", rows_inserted));

                             }
                     }
                }
                else
                {
                    System.out.println("Please enter the labor time in hours:");
                    final int laborTime = sc.nextInt();
                    sc.nextLine();
                    System.out.println("Connecting to the database...");
                    // Get a database connection and prepare a query statement
                     try (final Connection connection = DriverManager.getConnection(URL)) {
```

```
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                       try (
                                final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_7_1)) {
                                // Populate the query template with the data collected from the user

                               statement1.setInt(1, datejobCompleted);
                               statement1.setInt(2, laborTime);
                               statement1.setInt(3, type);
                               statement1.setInt(4, jobnum);
                                System.out.println("Dispatching the query 7(1)...");
                                // Actually execute the populated query
                                final int rows_inserted = statement1.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));

                           }

                       try (
                                final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_4)) {
                                // Populate the query template with the data collected from the user
                               //System.out.print(cname);

                                statement3.setInt(1, jobnum);
                                statement3.setInt(2, laborTime);

                                System.out.println("Dispatching the query 7(4)...");
                                // Actually execute the populated query
                                final int rows_inserted = statement3.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted in fitjob table..", rows_inserted));

                            }
                        }
                    }
                    break;
                case "11":
                    System.out.println("Please enter the assembly ID which has passed so far :");
                    final int aID = sc.nextInt();
                    sc.nextLine();

                    try (final Connection connection = DriverManager.getConnection(URL)) {
                        try (
                                final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_11)) {
                                statement1.setInt(1, aID);
                                System.out.println("Dispatching the query 11...");
                                ResultSet rs1 = statement1.executeQuery();
                                System.out.println("Process ID  Department Number");
                                while(rs1.next()) {
                                    System.out.println(rs1.getInt(1) + "\t" + rs1.getInt(2));
                                }
                    }
                    }
                    break;
                case "12":
                    System.out.println("Please enter the date that job completed:");
```

```
                    System.out.println("Please enter the date that job completed:");
                    final int jobcomp = sc.nextInt();
                    sc.nextLine();
                    System.out.println("Please enter the Department for the job:");
                    final int depjob = sc.nextInt();
                    sc.nextLine();
                    System.out.println("Connecting to the database...");
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                        try (final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_12_1)) {
                            statement1.setInt(1, depjob);
                            System.out.println("Dispatching the query 12(1)...");
                            ResultSet rs1 = statement1.executeQuery();
                            while(rs1.next()) {
                                int processidd = rs1.getInt(1);
                                try (final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_12_2)) {
                                    statement2.setInt(1, processidd);
                                    System.out.println("Dispatching the query 12(2)");
                                    ResultSet rs2 = statement2.executeQuery();
                                    while (rs2.next()) {
                                        int jobid = rs2.getInt(1);
                                        int assemblyiID = rs2.getInt(2);
                                        try (final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_12_3)) {
                                            statement3.setInt(1, jobid);
                                            statement3.setInt(2, jobcomp);
                                            System.out.println("Dispatching query 12(3)");
                                            ResultSet rs3 = statement3.executeQuery();
                                            while(rs3.next()) {
                                                int Type1 = rs3.getInt(1);
                                                if(Type1 == 1) {
                                                    System.out.println(jobid + "\t" + assemblyiID + "\t" + "Cut Job");
                                                }
                                                else if(Type1 == 2) {
                                                    System.out.println(jobid + "\t" + assemblyiID + "\t" + "Paint Job");
                                                }
                                                else
                                                    System.out.println(jobid + "\t" + assemblyiID + "\t" + "Fit Job");
                                            }
                                        }
                                    }
                                }
                            }
                        }

                    }
                    break;
                case "10":
                    System.out.println("Please enter the Department number that you want find the labor cost:");
                    final int department = sc.nextInt();
                    sc.nextLine();

                    System.out.println("Please enter the date of completion for the jobs:");
                    final int comdate = sc.nextInt();
                    sc.nextLine();
```

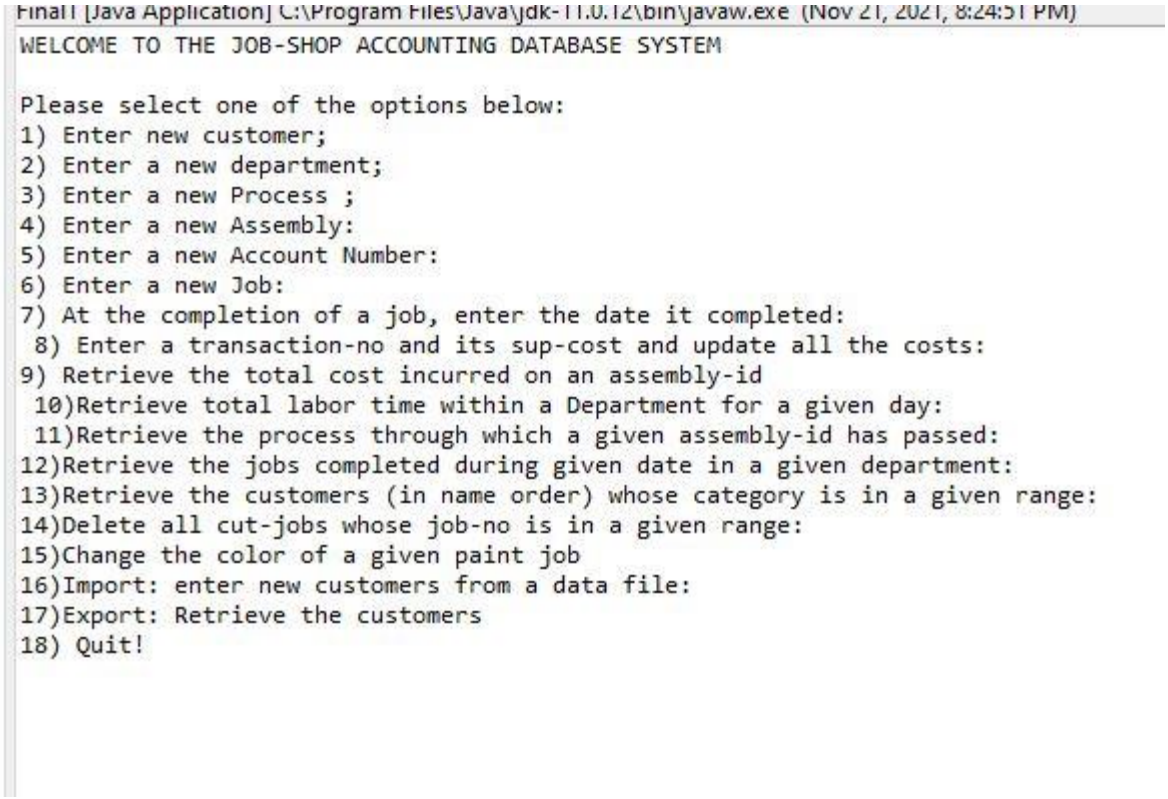
```
                sc.nextLine();
                // Determining the Processes supervised by the given Department:
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                            final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_10_1)) {
                            // Populate the query template with the data collected from the user
                            //System.out.print(cname);
                            statement1.setInt(1, department);
                            System.out.println("Dispatching the query 1...");
                            // Actually execute the populated query
                            ResultSet rs = statement1.executeQuery();
                            int psize = 0;
                            int jsize = 0;
                            int[] iarr = new int[77];
                            int[] jjj = new int[77];
                            int pindex = 0;
                            int jindex = 0;
                            while (rs.next()) {
                                int prcid = rs.getInt("processId");
                                iarr[pindex] = prcid;
                                pindex++;
                                psize++;
                            }
                            for (int i = 0; i < psize; i++)
                            {
                                System.out.println(iarr[i]);
                                try (
                                        final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_10_2)) {
                                        statement2.setInt(1, iarr[i]);
                                        System.out.println("Dispatching the query 10(2)...");
                                        ResultSet rs1 = statement2.executeQuery();
                                        int jobN = 0;
                                        while(rs1.next()) {
                                            int jobno = rs1.getInt("jobnomber");
                                            jjj[jindex] = jobno;
                                            jindex++;
                                        }
                                }
                            }
                            int[] rrr = new int[jindex];
                            int[] trtr = new int[jindex];
                            for(int k = 0; k < jindex; k++)
                            {
                                System.out.println(jjj[k]);
                                try (
                                        final PreparedStatement statement4 = connection.prepareStatement(QUERY_TEMPLATE_10_3)) {
                                        statement4.setInt(1, jjj[k]);
                                        System.out.println("Dispatching the query 10(c)...");
                                        ResultSet rs2 = statement4.executeQuery();

                                        while(rs2.next()) {
                                            trtr[k] = rs2.getInt("labortime");
                                            rrr[k] = rs2.getInt("dateofjobcompleted");
```

```
                                        rrr[k] = rs2.getInt("dateofjobcompleted");
                                    }
                                }
                            }
                            int totalLaborTime = 0;
                            for (int l = 0; l < jindex; l++) {
                                if(rrr[l] == comdate)
                                    totalLaborTime += trtr[l];
                            }
                            System.out.println("The total labor time for " + department + " departmen for jobs completed during the " +
                                    comdate +  " date is: " + totalLaborTime + " hours.");

                        }
                    }
                    break;
                case "13":
                    System.out.println("enter min for category:");
                    final int minimum = sc.nextInt();
                    sc.nextLine();
                    System.out.println("enter max for category::");
                    final int maximum = sc.nextInt();
                    sc.nextLine();
                    System.out.println("Connecting to the database...");
                    // Get the database connection, create statement and execute it right away, as no user input need be collected
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                        System.out.println("Dispatching the query...");
                        try (
                                final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_13)) {

                            statement2.setInt(1, minimum);
                            statement2.setInt(2, maximum);
                            System.out.println("Dispatching the query 13...");
                            ResultSet rs1 = statement2.executeQuery();
                            System.out.println("Contents of the Student table:");
                            System.out.println("name | address | Category");

                            // Unpack the tuples returned by the database and print them out to the user
                            while (rs1.next()) {

                                String name = rs1.getString("cname");
                                String address = rs1.getString("caddress");
                                int cat = rs1.getInt("category");
                                System.out.println(name + "\t" + address + "\t\t" + cat);
                                //resultSet.getString(2),
                                //resultSet.getString(3)

                            }
                        }

                    }
                    break;
                case "14":
                    System.out.println("minimum for job number:");
```

```
                System.out.println("minimum for job number:");
                final int minijobnum = sc.nextInt();
                sc.nextLine();
                System.out.println("maximum for job number:");
                final int maxjobnum = sc.nextInt();
                sc.nextLine();
                System.out.println("Connecting to the database...");
                // Get the database connection, create statement and execute it right away, as no user input need be collected
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                            final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_14_1)) {
                            // Populate the query template with the data collected from the user

                            statement1.setInt(1, minijobnum);
                            statement1.setInt(2, maxjobnum);
                            System.out.println("Dispatching the query 14(1)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement1.executeUpdate();
                            System.out.println(String.format("Done. %d rows deleted in cutjob table..", rows_inserted));
                        }

                    try (
                            final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_14_2)) {
                            // Populate the query template with the data collected from the user

                            statement2.setInt(1, minijobnum);
                            statement2.setInt(2, maxjobnum);
                            System.out.println("Dispatching the query 14(2)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement2.executeUpdate();
                            System.out.println(String.format("Done. %d rows deleted in assign table..", rows_inserted));
                        }
                    try (
                            final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_14_3)) {
                            // Populate the query template with the data collected from the user

                            statement3.setInt(1, minijobnum);
                            statement3.setInt(2, maxjobnum);
                            System.out.println("Dispatching the query 14(3)...");
                            // Actually execute the populated query
                            final int rows_inserted = statement3.executeUpdate();
                            System.out.println(String.format("Done. %d rows deleted in job table..", rows_inserted));
                        }
                }
                  break;
            case "15":
                System.out.println("enter job number that you want change color:");
                final int jobnumm = sc.nextInt();
                sc.nextLine();
                System.out.println("enter new color:");
                final String color = sc.nextLine();
                System.out.println("Connecting to the database...");
                // Get the database connection, create statement and execute it right away, as no user input need be collected
```

```
                    // Get the database connection, create statement and execute it right away, as no user input need be collected
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                        try (
                                final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_15)) {
                                // Populate the query template with the data collected from the user
                                statement1.setString(1, color);
                                statement1.setInt(2, jobnumm);
                                System.out.println("Dispatching the query 15...");
                                // Actually execute the populated query
                                final int rows_inserted = statement1.executeUpdate();
                                System.out.println(String.format("Done. %d rows updated in painjob table..", rows_inserted));
                            }
                    }
                    break;
                case "16":
                    System.out.println("Where is path of file you want to enter?:");
                    sc.nextLine();
                    final String file = sc.nextLine();
                    BufferedReader br;
                    String line = "";
                    String split = ",";
                try {
                    br = new BufferedReader(new FileReader(file));
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                        while ((line = br.readLine()) != null)
                        {
                            String[] customer = line.split(split);

                            try (
                                    final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_1)) {
                                    // Populate the query template with the data collected from the user
                                    statement1.setString(1, customer[0]);
                                    statement1.setString(2, customer[1]);
                                    statement1.setInt(3, Integer.parseInt(customer[2]));

                                    System.out.println("Dispatching the query 1...");

                                    final int rows_inserted = statement1.executeUpdate();
                                    System.out.println(String.format("Done. %d rows inserted in customer table..", rows_inserted));

                                }
                            }

                    br.close();
                    }
                } catch (FileNotFoundException e)
                {

                    System.out.println("not found!");
                    e.printStackTrace();
                    break;
                } catch (IOException e) {
                    System.out.println("eroooooor");
```

```
                System.out.println("eroooooor");
                e.printStackTrace();
                break;
            }
            break;
            case "17":
                System.out.println("enter min for category:");
                final int mincategory = sc.nextInt();
                sc.nextLine();
                System.out.println("enter max for category:");
                final int maxcategory = sc.nextInt();
                sc.nextLine();
                System.out.println("enter output file name");
                final String fileName = sc.nextLine();
                System.out.println("Connecting to the database...");
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    System.out.println("Dispatching the query...");
                    try (
                            final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_13)) {

                            statement2.setInt(1, mincategory);
                            statement2.setInt(2, maxcategory);
                            System.out.println("Dispatching the query 13...");
                            ResultSet tre = statement2.executeQuery();

                            try (PrintWriter rrr = new PrintWriter(fileName + ".csv")) {
                            while (tre.next()) {
                                StringBuilder sub = new StringBuilder();
                                    String name = tre.getString("cname");
                                    sub.append(name);
                                    sub.append(',');
                                    String address = tre.getString("caddress");
                                    sub.append(address);
                                    sub.append(',');
                                    int cat = tre.getInt("category");
                                    sub.append(cat);
                                    sub.append('\n');
                                    rrr.write(sub.toString());
                            }
                            }catch (FileNotFoundException e)

                            {
                                  System.out.println(e.getMessage());
                            }
                    }

                }
                break;
            case "18":
                System.out.println("Exiting! bye!");
                System.exit(0);
            default:
                System.out.println(String.format(
                            System.out.println(String.format(
                                "Unrecognized option: %s\n" +
                                "Please try again!",
                                option));
                            break;

                 }
        }
        sc.close();
    }
}

```


Final1 [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (Nov 21, 2021, 8:24:51 PM)
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
10)Retrieve total labor time within a Department for a given day:
11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
18) Quit!

Task 6. **Java program execution**
(Run the program created for Tasks 5 to test its correctness.)

6.1. Screenshots showing the testing of query 1

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) enter new customer;
2) Enter a new department;
3) Enter a new Process and its department together ;
4) Enter a new assembly with its customer-name and ....:
5) Create a new account and associate it with the process ...:
6) Enter a new job, given its job-no,...:
7) Enter the Completion date of a Job and Enter a new job, given its job-no,:
 8) Enter a transaction-no and its sup-cost ...:
10)Retrieve the total labor time within a department for a given day:
 13)Retrieve the customers (in name order) whose category is in a given range:
14)Delete all cut-jobs with a Job Number:
15)Change the color of a Paint Job:
16) Import: enter new customers from a data file: 20) Exit!
1
Please enter customer cname:
Naeem Sani
Please enter customer address:
Norman
Please enter customer category:
3
Connecting to the database...
NaeemDispatching the query...
Done. 1 rows inserted.
```

After 5 Run for query number 1.

Results Messages

	cname	caddress	category
1	Ali	bouston	9
2	ferial	oklahoma	2
3	jorj	newyork	7
4	mary	alberta	6
5	Naeem	Norman	3

6.2. Screenshots showing the testing of query 2

Query 2:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) enter new customer;
2) Enter a new department;
3) Enter a new Process and its department together ;
4) Enter a new assembly with its customer-name and ....:
5) Create a new account and associate it with the process ...:
6) Enter a new job, given its job-no,...:
7) Enter the Completion date of a Job and Enter a new job, given its job-no,:
 8) Enter a transaction-no and its sup-cost ...:
10)Retrieve the total labor time within a department for a given day:
 13)Retrieve the customers (in name order) whose category is in a given range:
14)Delete all cut-jobs with a Job Number:
15)Change the color of a Paint Job:
16) Import: enter new customers from a data file: 20) Exit!
2
Please enter department number
110
Please enter department data:
Mechanic
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

After 5 run for query 2

Results Messages

	departmentnumber	departmentdata
1	110	Mechanic
2	111	cs department
3	120	industry
4	130	electrical
5	134	art

6.3. Screenshots showing the testing of query 3

10 query for type 3.

Process table :

Results Messages

	processId	processData
1	10	trailer
2	12	very good
3	20	bad
4	33	very nice
5	45	test data
6	55	true
7	60	some
8	77	test
9	78	another test data
10	99	test
11	110	spring

Cut table:

Results Messages

	cuttingtype	machinetype	processId
1	fast	auto	33
2	chisel	lathe	99
3	hard	komatsu	110

Paint table:

Results Messages

	painttype	paintingmethod	processId
1	scrach	fast	12
2	good paint	easy	20
3	spray	automate	45
4	nice	color	60

Fit table:

Results Messages

	fittype	processId
1	best	10
2	good	55
3	close fit	77

Supervise table:

Results Messages

	processId	departmentnumber	departmentdata
1	10	110	mechanic
2	12	110	
3	20	110	mechanic
4	33	120	industry
5	45	111	cs department
6	55	134	art
7	60	130	electrical
8	77	111	cs department
9	99	111	cs department
10	110	111	cs department

6.4. Screenshots showing the testing of query 4

10 query for type 4.

```
4
 enter a new Assembly ID:
20
Please enter the Date of ordered of assembly:
11102019
Please enter the Assembly details:
good
Please enter the associated Customer Name:
Ali
Please enter the number of associated process IDs with this assembly
2
 enter %dth process ID:
60
 enter %dth process ID:
20
Dispatching the query 4(1)...
Done. 1 rows inserted in Assembly table.
Dispatching the query 4(2)...
Done. 1 rows inserted in order1 table.
Dispatching the query 4(3)...
Done. 1 rows inserted in passthrugh table.
Dispatching the query 4(3)...
Done. 1 rows inserted in passthrugh table.
```

Assembly Table :

Results Messages

	Assemblyid	dateofordered	AssemblyDeatails
1	10	11182021	trial
2	12	11102021	test assembly
3	14	11182021	trial
4	20	11102019	good
5	30	1012019	nice
6	40	1012010	very good
7	50	1022021	welldone
8	60	1042020	fast
9	70	4022020	best
10	90	1052020	slow

Order1 Table:

Results Messages

	Assemblyid	cname
1	14	ferial
2	20	Ali
3	30	Naeem
4	40	ferial
5	50	mary
6	60	Ali
7	70	naeem
8	90	Naeem

Pass-through Table:

Results Messages

	Assemblyid	processId
1	12	45
2	12	77
3	14	33
4	20	20
5	20	60
6	30	33
7	40	10
8	50	20
9	60	12
10	70	99
11	90	110

6.5. Screenshots showing the testing of query 5

10 query for type 5.

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
5
 enter a new Account number
30
 enter the date of account commenced please:
12021990
 enter the account type (1,2,3): 1.Departmentt Account, 2.Assembly Account, 3.Process Account:
2
Please enter the associated Assembly ID:
12
Connecting to the database...
Dispatching the query 5(a)...
Done. 1 rows inserted in account table.
Dispatching the query 5(3)...
Executing Statement 5(c)...
Done. 1 rows inserted in assemblyaccount table.
Dispatching the query 5(6)...
Executing Statement 5(6)...
Done. 1 rows inserted in record assembly cost table.
```

Account Table:

Results Messages

	accountnumber	dateofaccount
1	1	23
2	2	34
3	4	45
4	5	56
5	6	34
6	10	11202021
7	12	11102021
8	13	11102021
9	20	10202020
10	30	12021990
11	32	13
12	40	12122020
13	42	1012020
14	56	67
15	67	78
16	89	56
17	99	23
18	110	11112020

Process account table:

Results Messages

	accountnumber	sup_cost
1	10	0
2	42	0
3	110	0

Department account table:

Results Messages

	accountnumber	sup_cost
1	13	0
2	20	0
3	40	0

Assembly account table:

Results Messages

	accountnumber	sup_cost
1	2	0
2	6	0
3	30	0

6.6. Screenshots showing the testing of query 6

10 query for type 6.

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
6
Please enter new Job Number:
60
Please enter the date of job commenced:
03032007
Please enter the Process ID of to this job:
77
 enter the Assembly ID of process for this job:
50
Dispatching the query 6(1)...
Done. 1 rows inserted in Job table..
Dispatching the query 6(2)...
Done. 1 rows inserted in assign table..
```

Job Table

```
1   select * from job
```

Results Messages

	jobNomber	dateofjobcompleted	dateofjobcommenced	labortime	job_type
1	10	NULL	10012020	NULL	NULL
2	20	NULL	12122020	NULL	NULL
3	30	NULL	1012010	NULL	NULL
4	40	NULL	1012008	NULL	NULL
5	50	NULL	5052020	NULL	NULL
6	60	NULL	3032007	NULL	NULL
7	70	NULL	1042001	NULL	NULL
8	80	NULL	2032021	NULL	NULL
9	90	NULL	8082007	NULL	NULL
10	100	NULL	4052019	NULL	NULL

Assign Table:

```
1    select * from assign
```

Results Messages

	Assemblyid	processId	jobnomber
1	12	10	10
2	14	12	100
3	20	10	90
4	30	55	80
5	40	60	70
6	50	77	60
7	60	20	40
8	60	78	50
9	70	110	30
10	90	12	20

6.7. Screenshots showing the testing of query 7

10 query for type 7.

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
7
Please enter the Job Number which is in case 6:
20
Please enter the date the job completed:
15112023
Please enter type of Job(1,2,3): 1.Cut Job, 2. Paint Job , 3.Fit Job
2
Please enter the color:
black
Please enter the volume of paint :
6
Please enter the labor time in hours:
6
Connecting to the database...
Dispatching the query 7(1)...
Done. 1 rows inserted in Job table..
Dispatching the query 7(3)...
Done. 1 rows inserted in painjob table..
```

Job table:

```
1   select * from job
```

Results Messages

	jobNomber	dateofjobcompleted	dateofjobcommenced	labortime	job_type
1	10	1012021	10012020	2	3
2	20	15112023	12122020	6	2
3	30	13042015	1012010	4	1
4	40	5052016	1012008	9	1
5	50	1012021	5052020	102	2
6	60	1012010	3032007	100	3
7	70	10102005	1042001	98	1
8	80	12122022	2032021	200	2
9	90	1012015	8082007	70	3
10	100	11102022	4052019	90	3

Cut job Table:

```
1   select * from cutjob
```

Results Messages

	typeofmachine	material	jobnomber	amountoftime	labortime
1	toyota	wood	30	7	4
2	hyundai	metal	40	9	9
3	komatsu	iron	70	80	98

Paintjob Table:

```
1   select * from painjob
```

Results | Messages

	color	volume	jobnomber	labortime
1	black	6	20	6
2	red	17	50	102
3	dark	70	80	200

Fit job table:

```
1   select * from fitjob
```

Results Messages

	jobnomber	labortime
1	10	2
2	60	100
3	90	70
4	100	90

6.8
6.9.

6.10. Screenshots showing the testing of query 10

3 query for type 10:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
10
Please enter the Department number that you want find the labor cost:
110
Please enter the date of completion for the jobs:
1012021
Dispatching the query 1...
10
Dispatching the query 10(2)...
12
Dispatching the query 10(2)...
20
Dispatching the query 10(2)...
10
Dispatching the query 10(c)...
90
Dispatching the query 10(c)...
20
Dispatching the query 10(c)...
100
Dispatching the query 10(c)...
40
Dispatching the query 10(c)...
The total labor time for 110 departmen for jobs completed during the 1012021 date is: 2 hours.
```

```
Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
10
Please enter the Department number that you want find the labor cost:
111
Please enter the date of completion for the jobs:
13042015
Dispatching the query 1...
45
Dispatching the query 10(2)...
77
Dispatching the query 10(2)...
99
Dispatching the query 10(2)...
110
Dispatching the query 10(2)...
60
Dispatching the query 10(c)...
30
Dispatching the query 10(c)...
The total labor time for 111 departmen for jobs completed during the 13042015 date is: 4 hours.
```

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
10
Please enter the Department number that you want find the labor cost:
111
Please enter the date of completion for the jobs:
1012010
Dispatching the query 1...
45
Dispatching the query 10(2)...
77
Dispatching the query 10(2)...
99
Dispatching the query 10(2)...
110
Dispatching the query 10(2)...
60
Dispatching the query 10(c)...
30
Dispatching the query 10(c)...
The total labor time for 111 departmen for jobs completed during the 1012010 date is: 100 hours.
```

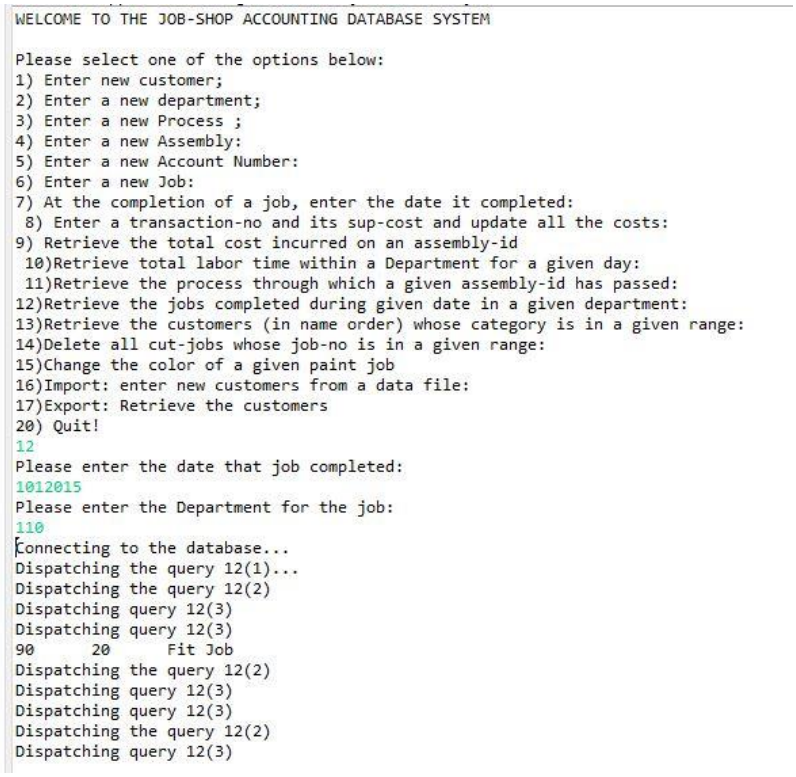
6.11. Screenshots showing the testing of query 11

3 query for type 11:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
11
Please enter the assembly ID which has passed so far :
10
Dispatching the query 11...
Process ID  Department Number
10      110
12      110
20      110
33      120
45      111
55      134
60      130
77      111
99      111
110     111
```

6.12. Screenshots showing the testing of query 12

3 query for type 12:



```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
12
Please enter the date that job completed:
1012015
Please enter the Department for the job:
110
Connecting to the database...
Dispatching the query 12(1)...
Dispatching the query 12(2)
Dispatching query 12(3)
Dispatching query 12(3)
90      20      Fit Job
Dispatching the query 12(2)
Dispatching query 12(3)
Dispatching query 12(3)
Dispatching the query 12(2)
Dispatching query 12(3)
```

```
Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
12
Please enter the date that job completed:
1012010
Please enter the Department for the job:
111
Connecting to the database...
Dispatching the query 12(1)...
Dispatching the query 12(2)
Dispatching the query 12(2)
Dispatching query 12(3)
60      50      Fit Job
Dispatching the query 12(2)
Dispatching the query 12(2)
Dispatching query 12(3)
```

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
12
Please enter the date that job completed:
11102022
Please enter the Department for the job:
110
Connecting to the database...
Dispatching the query 12(1)...
Dispatching the query 12(2)
Dispatching query 12(3)
Dispatching query 12(3)
Dispatching the query 12(2)
Dispatching query 12(3)
100     14      Fit Job
Dispatching query 12(3)
Dispatching the query 12(2)
Dispatching query 12(3)
```

6.13. Screenshots showing the testing of query 13

3 query for type 13:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
13
enter min for category:
3
enter max for category::
8
Connecting to the database...
Dispatching the query...
Dispatching the query 13...
Contents of the Student table:
name | address | Category
jorj    newyork         7
mary    alberta         6
Naeem   Norman          3
saeed   azizi           3
```

Run 2:

```
Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
13
enter min for category:
1
enter max for category::
10
Connecting to the database...
Dispatching the query...
Dispatching the query 13...
Contents of the Student table:
name | address | Category
Ali     bouston         9
ferial  oklahoma                2
Hamon   abq             9
jorj    newyork         7
mary    alberta         6
Naeem   Norman          3
saeed   azizi           3
```

Run 3:

```
Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
13
enter min for category:
2
enter max for category::
3
Connecting to the database...
Dispatching the query...
Dispatching the query 13...
Contents of the Student table:
name | address | Category
ferial  oklahoma               2
```

6.14. Screenshots showing the testing of query 14

3 query for 14:

This is a cut job table before run:

```
1   select * from cutjob
```

Results Messages

	typeofmachine	material	jobnomber	amountoftime	labortime
1	kia	good	10	2	3
2	komatsu	tech	20	67	34
3	jambo	nice	30	56	45
4	kmz	good	40	32	890
5	hard	best	50	65	87

Run 1:

```
20) Quit!
14
minimum for job number:
8
maximum for job number:
20
Connecting to the database...
Dispatching the query 14(1)...
Done. 1 rows deleted in cutjob table..
Dispatching the query 14(2)...
Done. 1 rows deleted in assign table..
Dispatching the query 14(3)...
Done. 1 rows deleted in job table..
```

```
1    select * from cutjob
```

Results Messages

	typeofmachine	material	jobnomber	amountoftime	labortime
1	komatsu	tech	20	67	34
2	jambo	nice	30	56	45
3	kmz	good	40	32	890
4	hard	best	50	65	87

Run 2:

```
14
minimum for job number:
10
maximum for job number:
30
Connecting to the database...
Dispatching the query 14(1)...
Done. 1 rows deleted in cutjob table..
Dispatching the query 14(2)...
Done. 1 rows deleted in assign table..
Dispatching the query 14(3)...
Done. 1 rows deleted in job table..
```

```
1    select * from cutjob
```

Results Messages

	typeofmachine	material	jobnomber	amountoftime	labortime
1	jambo	nice	30	56	45
2	kmz	good	40	32	890
3	hard	best	50	65	87

Run 3:

```
14
minimum for job number:
30
maximum for job number:
50
Connecting to the database...
Dispatching the query 14(1)...
Done. 1 rows deleted in cutjob table..
Dispatching the query 14(2)...
Done. 1 rows deleted in assign table..
Dispatching the query 14(3)...
Done. 1 rows deleted in job table..
```

```
1   select * from cutjob
```

Results Messages

	typeofmachine	material	jobnomber	amountoftime	labortime
1	jambo	nice	30	56	45
2	hard	best	50	65	87

6.15. Screenshots showing the testing of query 15

3 query for 15:
This is paintjob table before run.

```
1    select * from painjob
```

Results | Messages

	color	volume	jobnomber	labortime
1	red	67	60	90

Run 1:

```
15
enter job number that you want change color:
60
enter new color:
blue
Connecting to the database...
Dispatching the query 15...
Done. 1 rows updated in painjob table..
```

```
1   select * from painjob
```

Results Messages

	color	volume	jobnomber	labortime
1	blue	67	60	90

Run 2:

```
15
enter job number that you want change color:
60
enter new color:
yellow
Connecting to the database...
Dispatching the query 15...
Done. 1 rows updated in painjob table..
```

```
1   select * from painjob
```

Results Messages

	color	volume	jobnomber	labortime
1	yellow	67	60	90

Run 3:

```
15
enter job number that you want change color:
60
enter new color:
black
Connecting to the database...
Dispatching the query 15...
Done. 1 rows updated in painjob table..
```

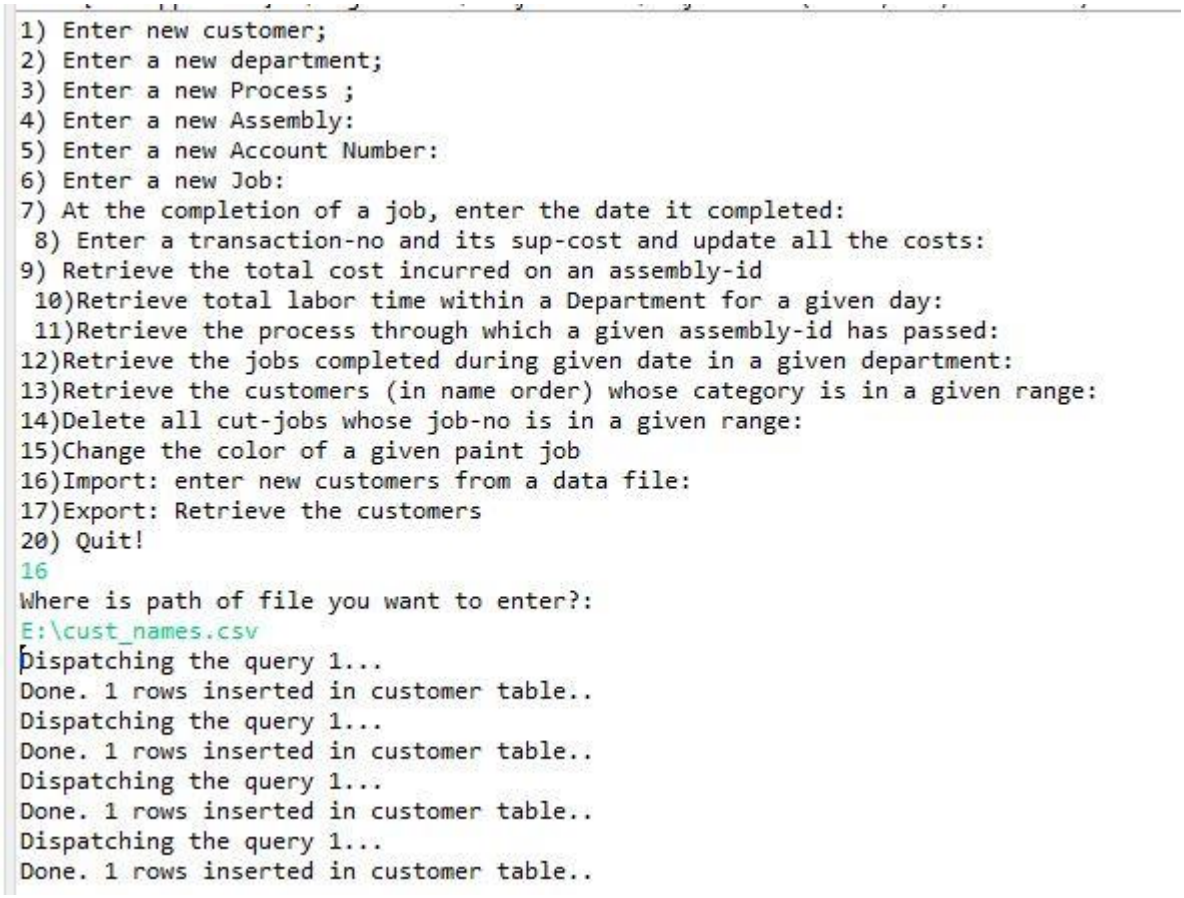
```
1   select * from painjob
```

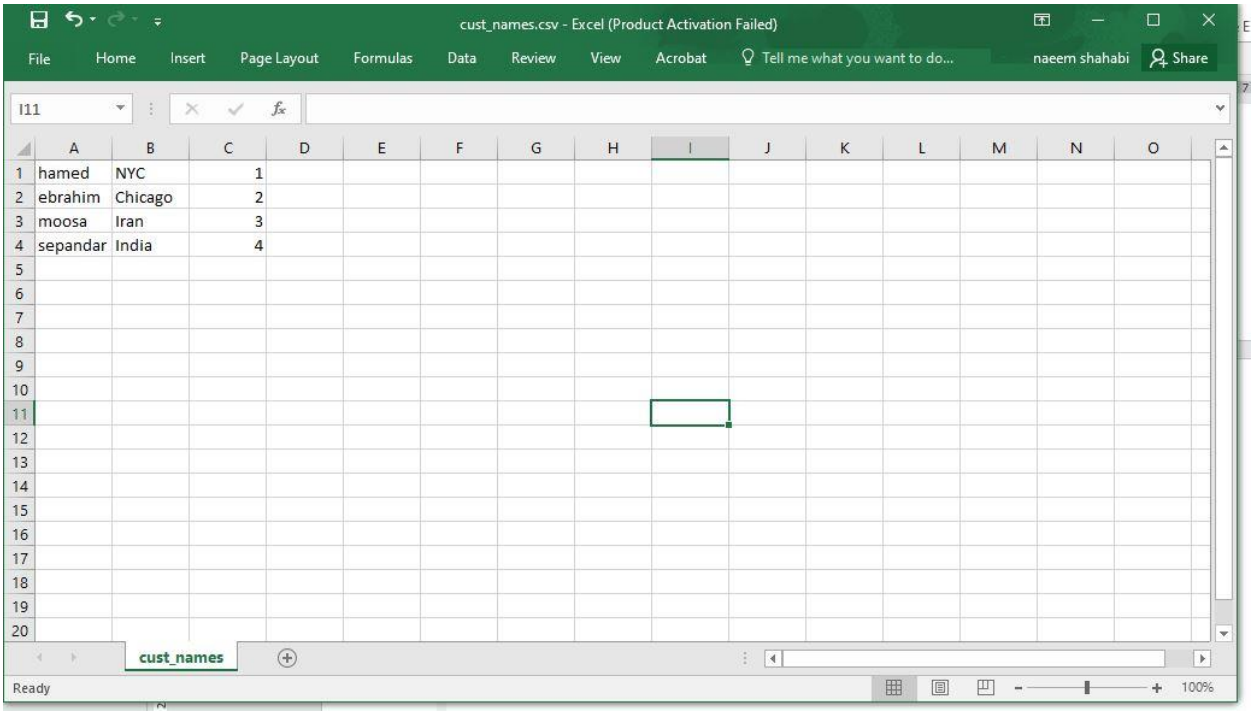
Results | Messages

	color	volume	jobnomber	labortime
1	black	67	60	90

6.16. Screenshots showing the testing of query 16

1 run for query 16:




cust_names.csv - Excel (Product Activation Failed)
File Home Insert Page Layout Formulas Data Review View Acrobat Tell me what you want to do... naeem shahabi Share
I11
A B C D E F G H I J K L M N O
1 hamed NYC 1
2 ebrahim Chicago 2
3 moosa Iran 3
4 sepandar India 4
cust_names
Ready
100%

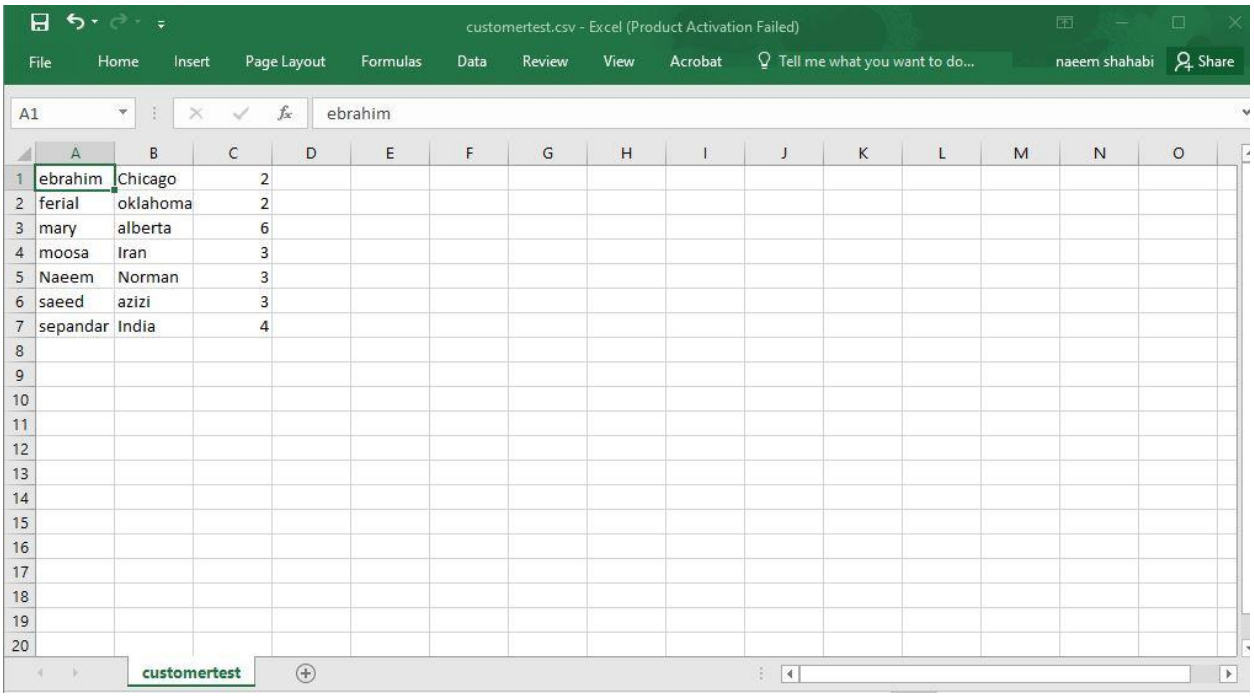
```
1   select * from customer
```

Results Messages

	cname	caddress	category
1	Ali	bouston	9
2	ebrahim	Chicago	2
3	ferial	oklahoma	2
4	hamed	NYC	1
5	Hamon	abq	9
6	hesam	boston	8
7	jorj	newyork	7
8	mary	alberta	6
9	moosa	Iran	3
10	Naeem	Norman	3
11	saeed	azizi	3
12	sepandar	India	4

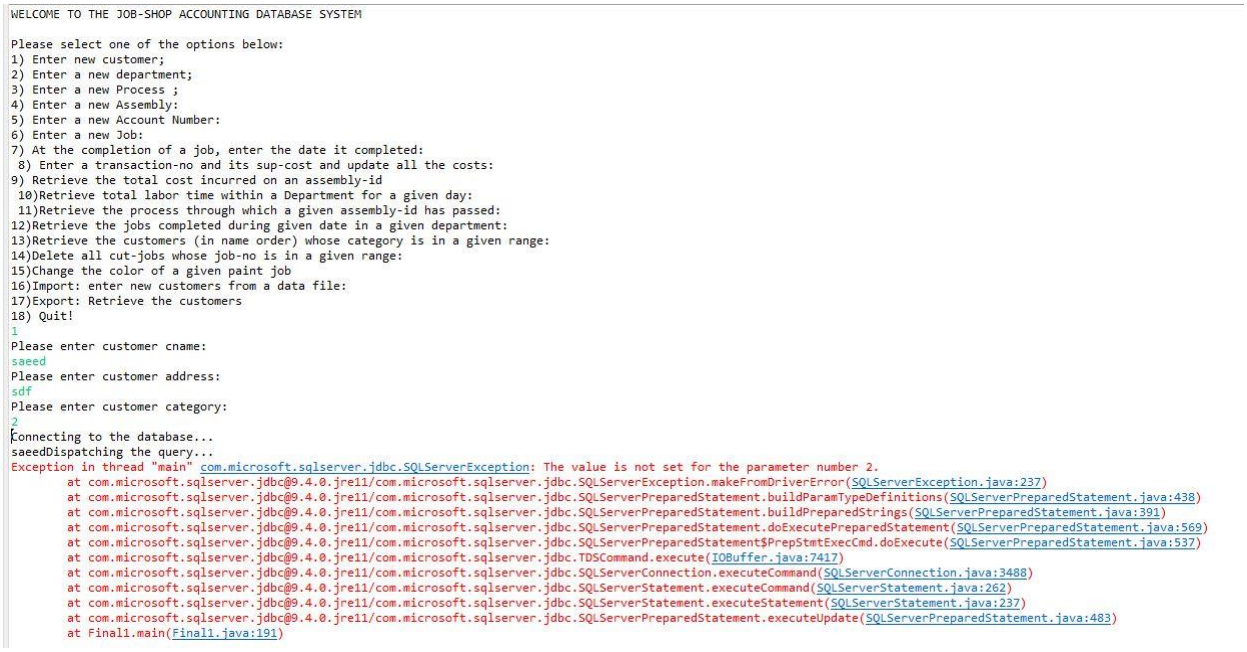
1 run for query 17:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
20) Quit!
17
enter min for category:
2
enter max for category:
7
enter output file name
customertest
Connecting to the database...
Dispatching the query...
Dispatching the query 13...
```



6.17. Screenshots showing the testing of query 17

Error checking.



```
<terminated> Final1 [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (Nov 21, 2021, 8:08:24 PM – 8:08:30 PM)
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter new customer;
2) Enter a new department;
3) Enter a new Process ;
4) Enter a new Assembly:
5) Enter a new Account Number:
6) Enter a new Job:
7) At the completion of a job, enter the date it completed:
 8) Enter a transaction-no and its sup-cost and update all the costs:
9) Retrieve the total cost incurred on an assembly-id
 10)Retrieve total labor time within a Department for a given day:
 11)Retrieve the process through which a given assembly-id has passed:
12)Retrieve the jobs completed during 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:
17)Export: Retrieve the customers
18) Quit!
1
Please enter customer cname:
saeed
Please enter customer address:
sabz
Please enter customer category:
4
Connecting to the database...
saeedException in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The index 4 is out of range.
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDriverError(SQLServerException.java:237)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.setterGetParam(SQLServerPreparedStatement.java:1131)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.setValue(SQLServerPreparedStatement.java:1145)
        at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.setFloat(SQLServerPreparedStatement.java:1391)
        at Final1.main(Final1.java:188)
```

Also errors in Azure SQL:

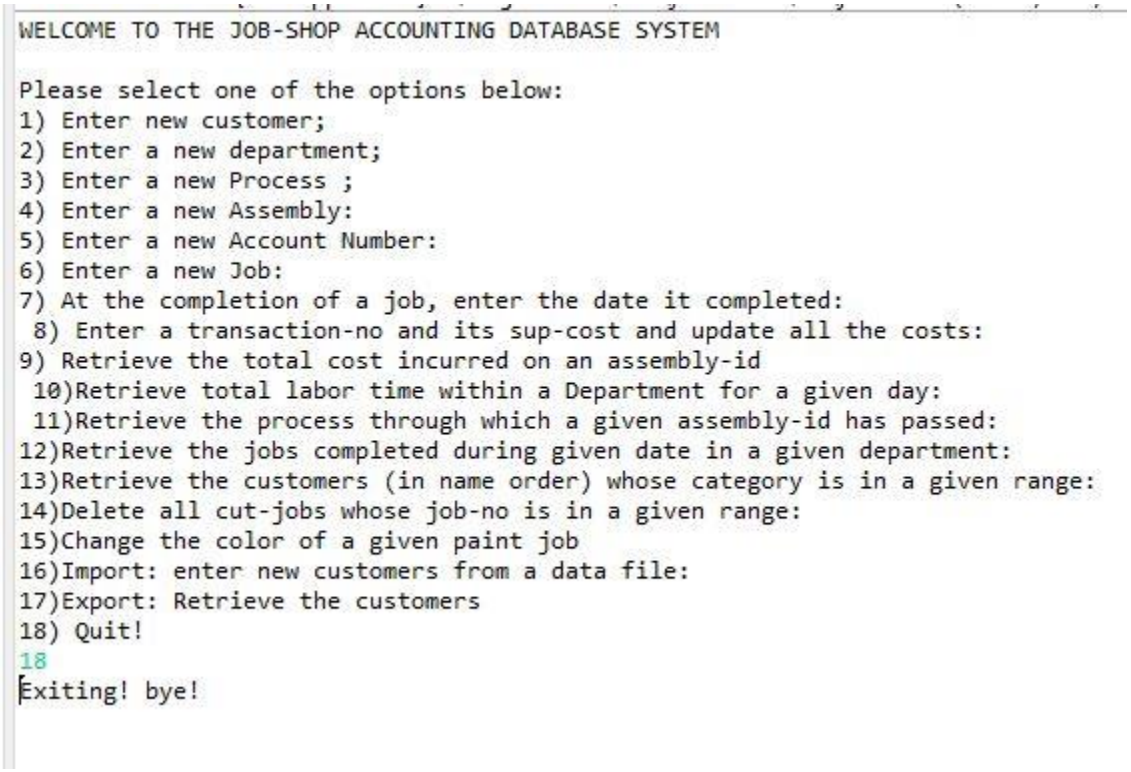
Messages

10:44:25 PM Started executing query at Line 1

Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK__Assembly__0334DF33D8C6AAA7'. Cannot insert duplicate key in object 'dbo.Assembly'. The duplicate key value is (444).
The statement has been terminated.
Msg 2627, Level 14, State 1, Line 3
Violation of PRIMARY KEY constraint 'PK__order1__DB438BE21804372C'. Cannot insert duplicate key in object 'dbo.order1'. The duplicate key value is (444, ferial).
The statement has been terminated.
Msg 2627, Level 14, State 1, Line 5
Violation of PRIMARY KEY constraint 'PK__passthru__33284181573B538A'. Cannot insert duplicate key in object 'dbo.passthrugh'. The duplicate key value is (444, 55).
The statement has been terminated.
Total execution time: 00:00:00.339

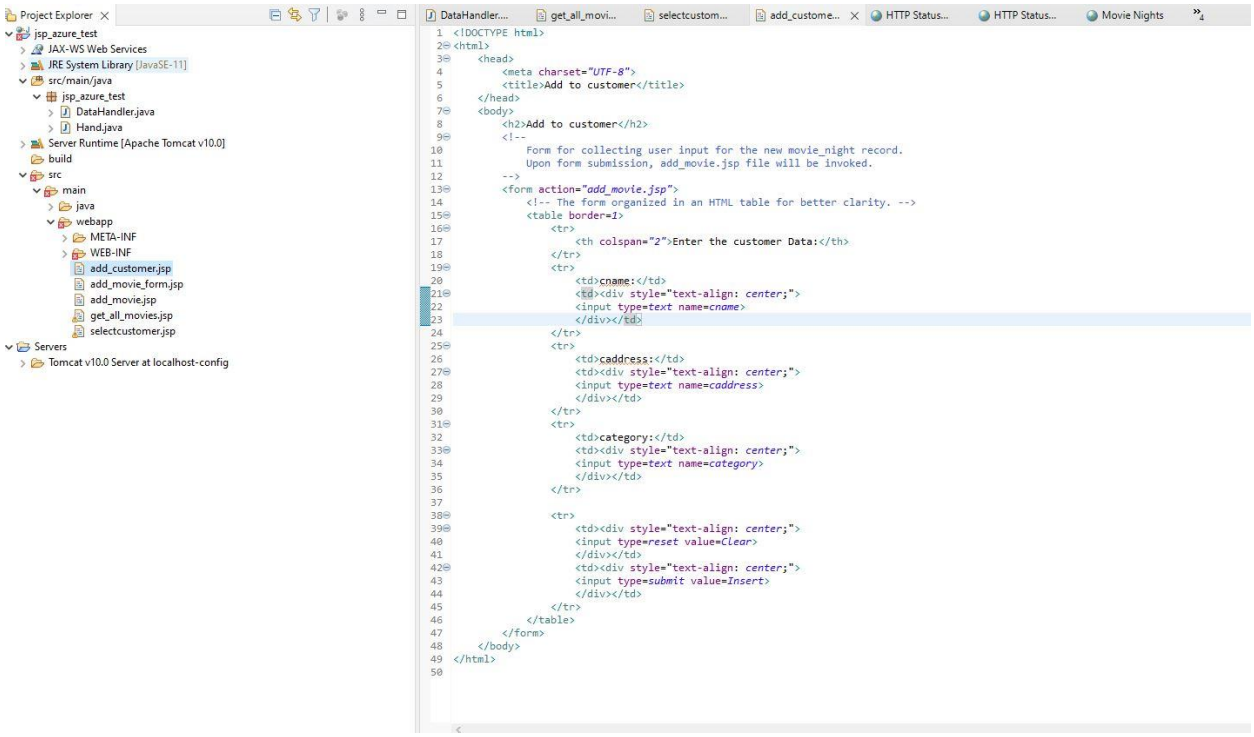
6.18. Screenshots showing the testing of query 18

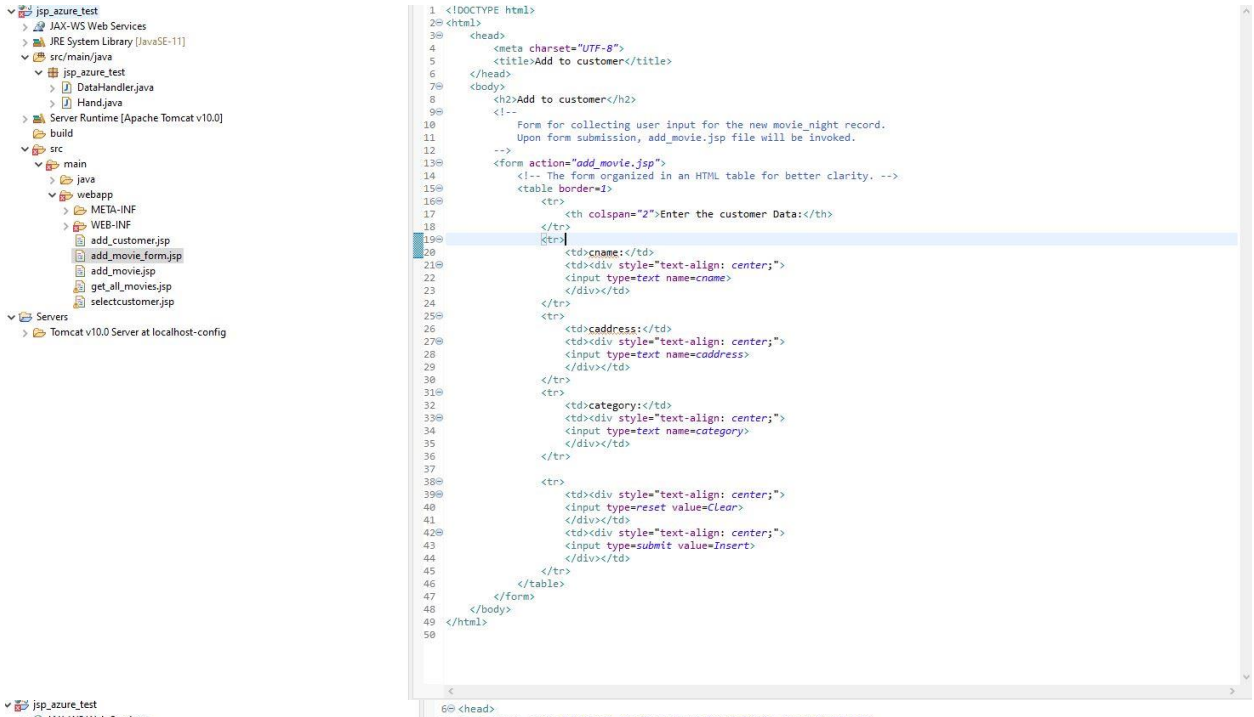
Run query 18:



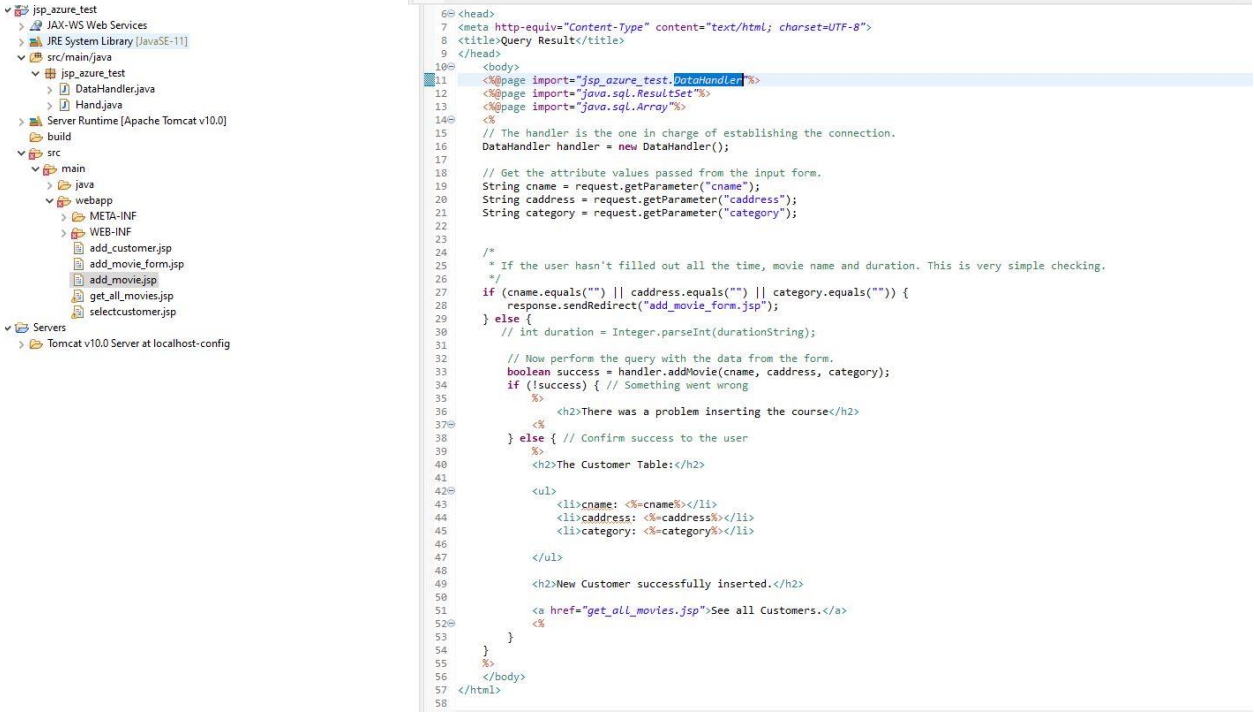
Task 7. Web database application and its execution

7.1. Web database application source program and screenshots

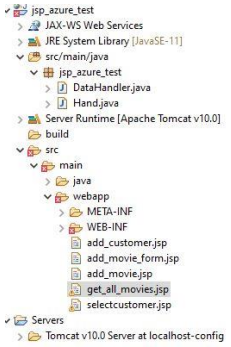




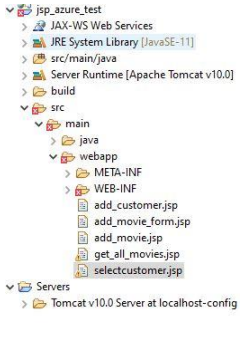

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Add to customer</title>
    </head>
    <body>
        <h2>Add to customer</h2>
        <!--
            Form for collecting user input for the new movie_night record.
            Upon form submission, add_movie.jsp file will be invoked.
        -->
        <form action="add_movie.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>cname:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=cname>
                    </div></td>
                </tr>
                <tr>
                    <td>caddress:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=caddress>
                    </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>
```




```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="jsp_azure_test.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

    // Get the attribute values passed from the input form.
    String cname = request.getParameter("cname");
    String caddress = request.getParameter("caddress");
    String category = request.getParameter("category");


    /*
     * If the user hasn't filled out all the time, movie name and duration. This is very simple checking.
     */
    if (cname.equals("") || caddress.equals("") || category.equals("")) {
        response.sendRedirect("add_movie_form.jsp");
    } else {
      // int duration = Integer.parseInt(durationString);

       // Now perform the query with the data from the form.
       boolean success = handler.addMovie(cname, caddress, category);
       if (!success) { // Something went wrong
           %>
               <h2>There was a problem inserting the course</h2>
           <%
       } else { // Confirm success to the user
           %>
           <h2>The Customer Table:</h2>

           <ul>
               <li>cname: <%=cname%></li>
               <li>caddress: <%=caddress%></li>
               <li>category: <%=category%></li>

           </ul>

           <h2>New Customer successfully inserted.</h2>

           <a href="get_all_movies.jsp">See all Customers.</a>
           <%
       }
    }
    %>
    </body>
</html>
```



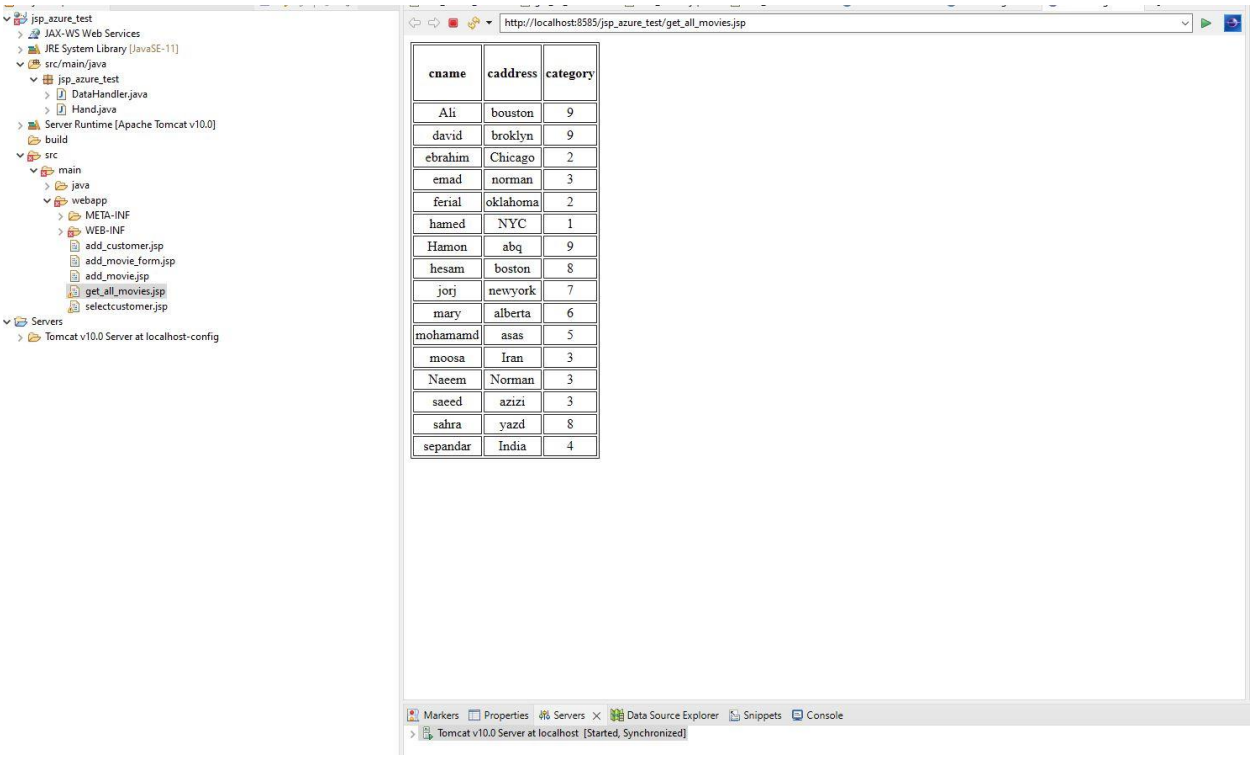

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Movie Nights</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.DataHandler"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get all the movies from the database
            final DataHandler handler = new DataHandler();
            final ResultSet movies = handler.getAllMovies();
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>cname</h4>
              </td>
              <td align="center">
                <h4>caddress</h4>
              </td>
              <td align="center">
                <h4>category</h4>
              </td>


            </tr>
            <%
               while(movies.next()) { // For each movie_night record returned...
                    // Extract the attribute values for every row returned
                    final String cname = movies.getString("cname");
                    final String caddress = movies.getString("caddress");
                    final String category = movies.getString("category");


                    out.println("<tr>"); // Start printing out the new table row
                    out.println( // Print each attribute value
                         "<td align=\"center\">" + cname +
                         "</td><td align=\"center\"> " + caddress +
                         "</td><td align=\"center\"> " + category +
                      "</td>");
                    out.println("</tr>");
                }
                %>
            </table>
    </body>
</html>
```




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Movie Nights</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.Hand"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get all the movies from the database
            final Hand handler = new Hand();
            final ResultSet movies = handler.selectcustomer();
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>cname</h4>
              </td>
              <td align="center">
                <h4>caddress</h4>
              </td>
              <td align="center">
                <h4>category</h4>
              </td>


            </tr>
            <%
               while(movies.next()) { // For each movie_night record returned...
                    // Extract the attribute values for every row returned
                    final String cname = movies.getString("cname");
                    final String caddress = movies.getString("caddress");
                    final String category = movies.getString("category");


                    out.println("<tr>"); // Start printing out the new table row
                    out.println( // Print each attribute value
                         "<td align=\"center\">" + cname +
                         "</td><td align=\"center\"> " + caddress +
                         "</td><td align=\"center\"> " + category +
                      "</td>");
                    out.println("</tr>");
                }
                %>
            </table>
```

7.2 Screenshots showing the testing of the Web database application

Get_all_movies.jsp: Show all the customers in customer table.

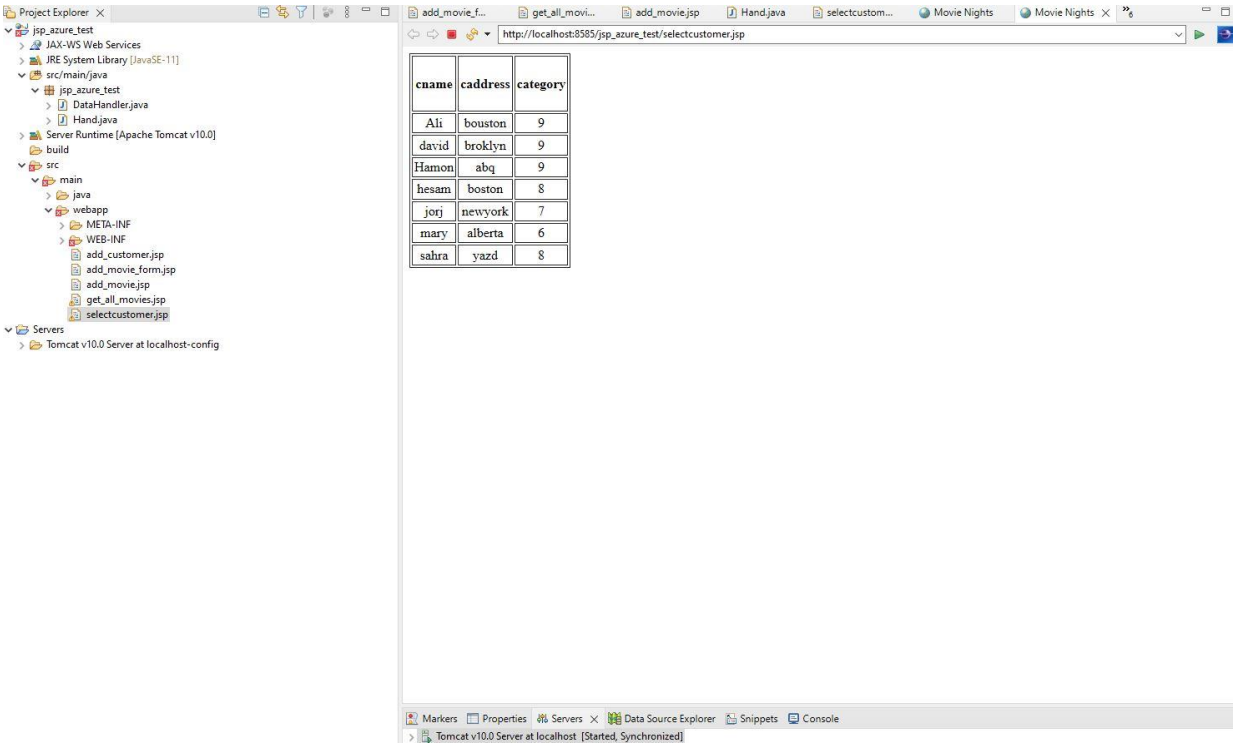


cname	caddress	category
Ali	bouston	9
david	broklyn	9
ebrahim	Chicago	2
emad	norman	3
ferial	oklahoma	2
hamed	NYC	1
Hamon	abq	9
hesam	boston	8
jorj	newyork	7
mary	alberta	6
mohamamd	asas	5
moosa	Iran	3
Naeem	Norman	3
saeed	azizi	3
sahra	yazd	8
sepandar	India	4

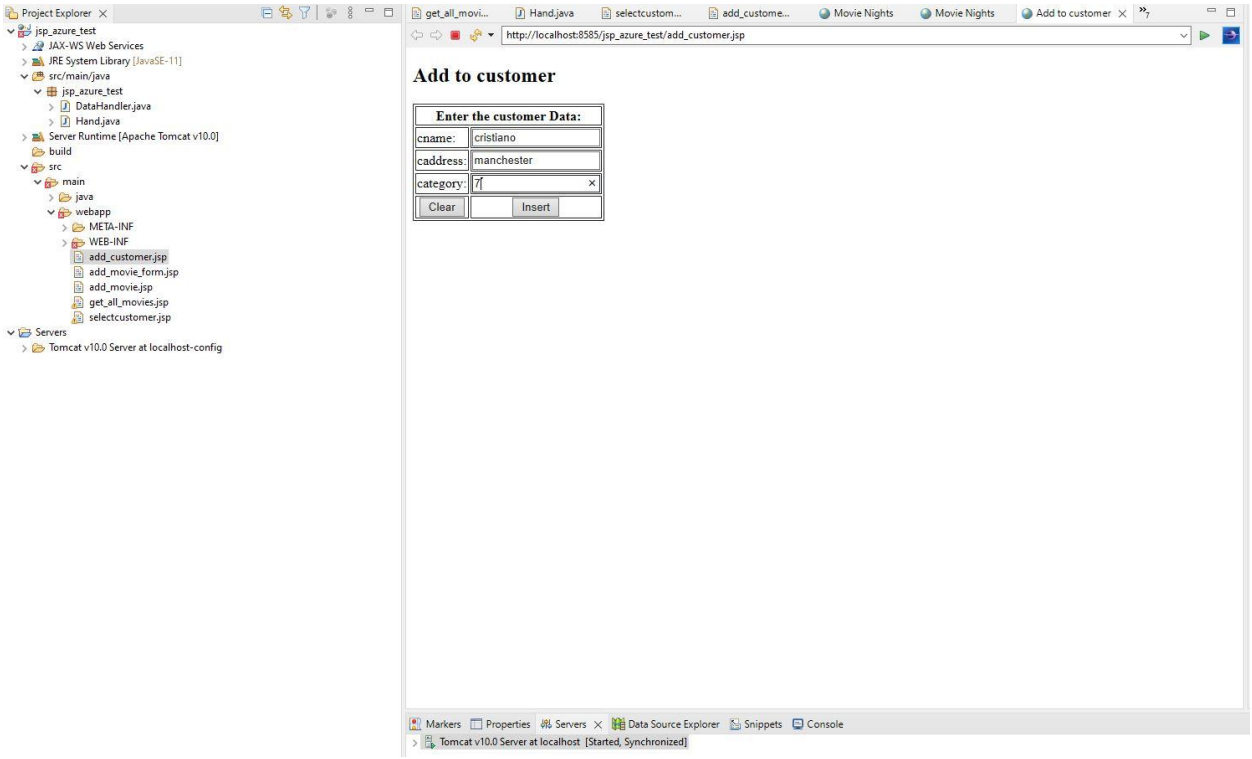
.

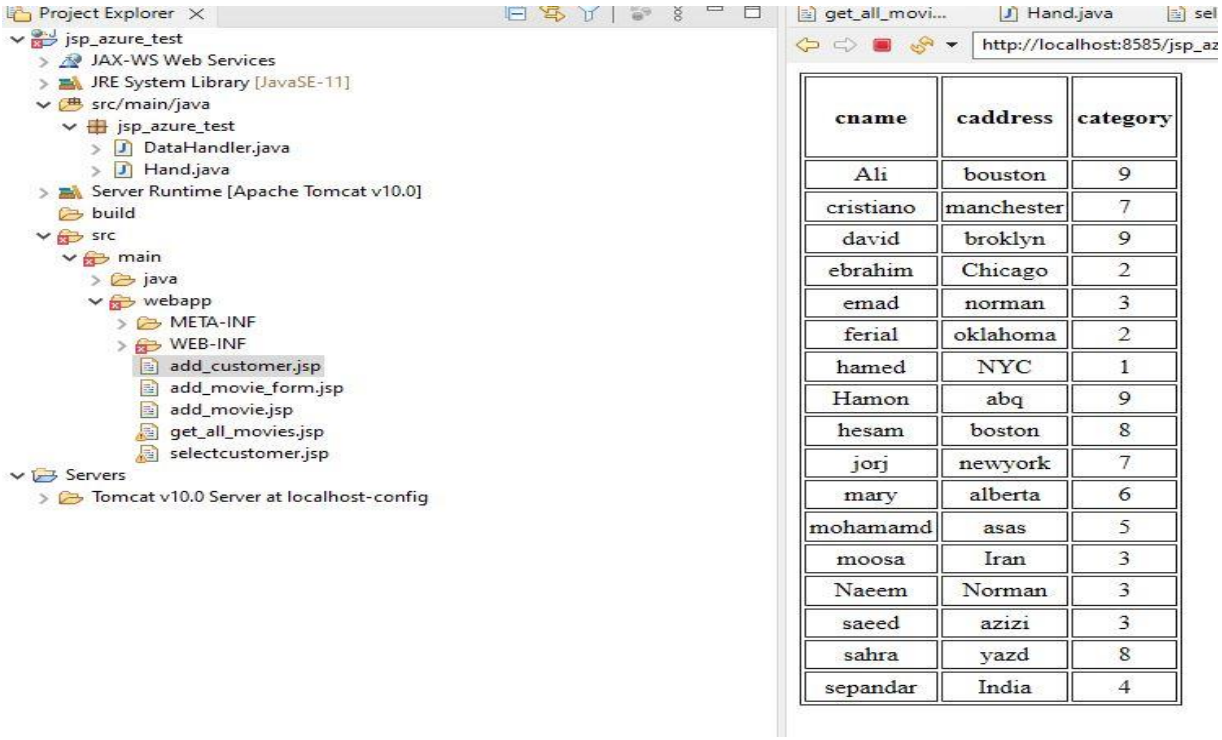
Run the program in this order : first query 13 then query 1 and then 13 again.

First run 13 for period between 5, 10.



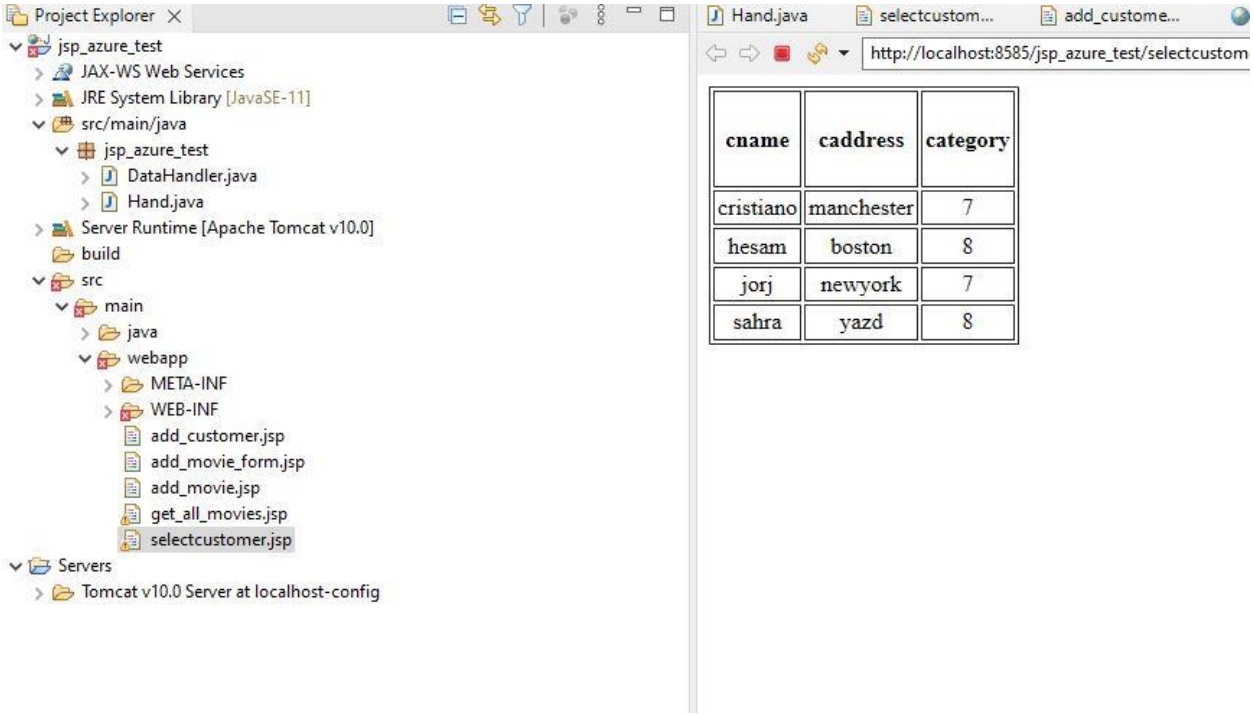
Run 1 and add new customer with category 7





cname	caddress	category
Ali	bouston	9
cristiano	manchester	7
david	broklyn	9
ebrahim	Chicago	2
emad	norman	3
ferial	oklahoma	2
hamed	NYC	1
Hamon	abq	9
hesam	boston	8
jorj	newyork	7
mary	alberta	6
mohamamd	asas	5
moosa	Iran	3
Naeem	Norman	3
saeed	azizi	3
sahra	yazd	8
sepandar	India	4

Run Query 13 again for period between 6 and 9.



cname	caddress	category
cristiano	manchester	7
hesam	boston	8
jorj	newyork	7
sahra	yazd	8

As you can see the result of query 1 change the result of query 13.