

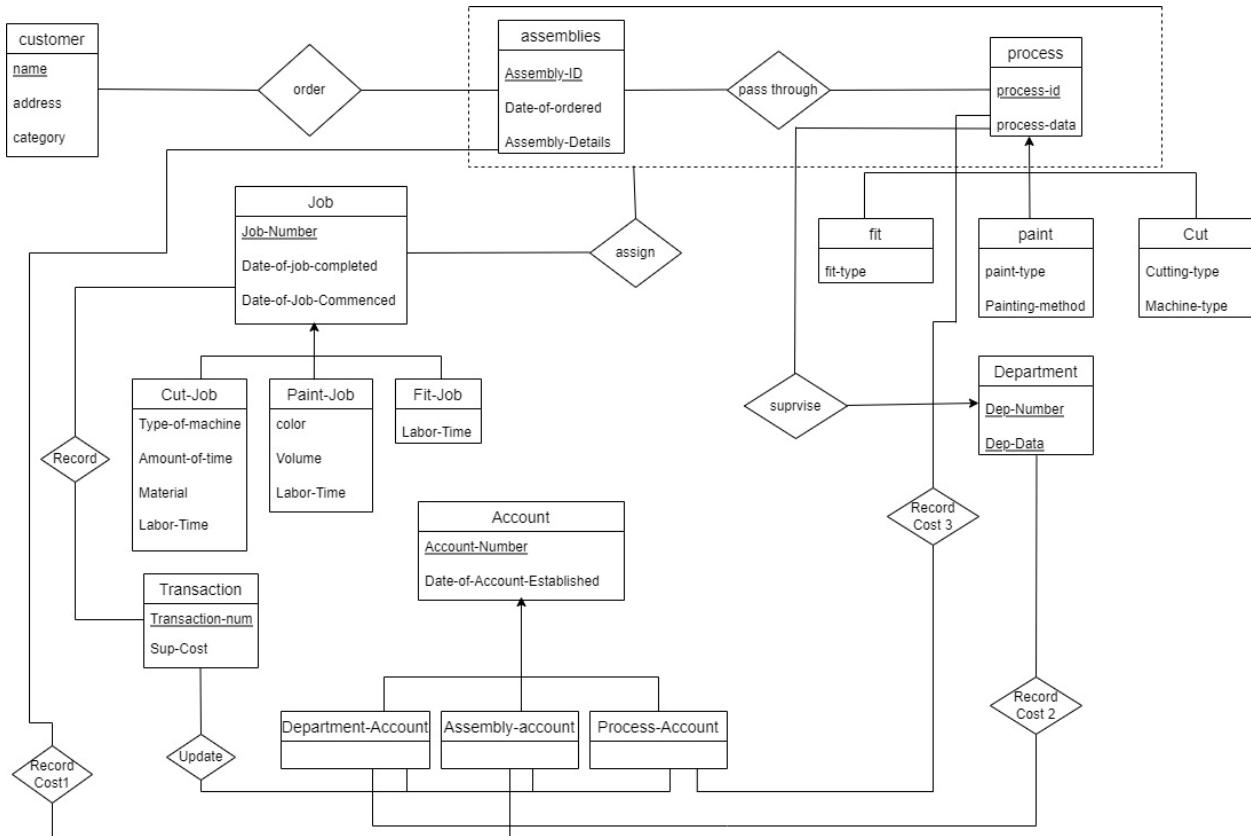
**Cs-4513-001-Fall 2021, Dr. Le Gruenwald,
Naeem Shahabi Sani, 113526767, shahabi@ou.edu,
INDIVIDUAL PROJECT: A JOB-SHOP ACCOUNTING SYSTEM**

Tasks Performed	Page Number
Task 1.	2-3
1.1. ER Diagram	2-2
1.2. Relational Database Schema	3-3
Task 2. Data Dictionary	4-7
Task 3.	7-11
3.1. Discussion of storage structures for tables	7-8
3.2. Discussion of storage structures for tables (Azure SQL Database)	9-11
Task 4. SQL statements and screenshots showing the creation of tables in Azure SQL Database	12-16
Task 5.	17-50
5.1. SQL statements (and Transact SQL stored procedures, if any)	17-27
Implementing all queries (1-15 and error checking)	
5.2. The Java source program and screenshots showing its successful compilation	28-50
Task 6. Java program Execution	51-90
6.1. Screenshots showing the testing of query 1	51-51
6.2. Screenshots showing the testing of query 2	52-53
6.3. Screenshots showing the testing of query 3	53-54
6.4. Screenshots showing the testing of query 4	55-56
6.5. Screenshots showing the testing of query 5	57-59
6.6. Screenshots showing the testing of query 6	60-61
6.7. Screenshots showing the testing of query 7	62-65
6.8. Screenshots showing the testing of query 8	65-65
6.9. Screenshots showing the testing of query 9	65-65
6.10. Screenshots showing the testing of query 10	66-68
6.11. Screenshots showing the testing of query 11	69-69
6.12. Screenshots showing the testing of query 12	70-72
6.13. Screenshots showing the testing of query 13	73-75
6.14. Screenshots showing the testing of query 14	76-79
6.15. Screenshots showing the testing of query 15	80-82
6.16. Screenshots showing the testing of the import and export options	83-86
6.17. Screenshots showing the testing of three types of errors	87-89
6.18. Screenshots showing the testing of the quit option	90-90

Tasks Performed	Page Number
Task 7. Web database application and its execution	91-97
7.1. Web database application source program and screenshots showing Its successful compilation	91-93
7.2. Screenshots showing the testing of the Web database application 9	94-97

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
<u>Name</u>	Varchar	32 Byte	Primary Key
Address	Varchar	32 Byte	
Category	Integer	4 Byte	1-10

<Assembly>

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

<Assign>

Field Name	Data Type	Size (Byte)	Constraints
<u>Assembly-ID</u>	INT	5	FK, Primary Key, Not Null
<u>Process-ID</u>	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
<u>Assembly-Id</u>	INT	5	FK, Primary Key, not null
<u>Process-ID</u>	INT	5	FK, Primary Key, not null

<Process>

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

<Order>

Field Name	Data Type	Size (Byte)	Constraints
<u>Assembly-Id</u>	INT	5	FK, Primary Key, not null
<u>CName</u>	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
<u>Job-Number</u>	INT	5	FK, Primary Key, Not null
Labor-Time	INT	5	

<Account>

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

<Department>

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

<Department-Account>

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

<Assembly-Account>

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

<Process-Account>

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

<Transaction>

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

<Record>

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

<Record cost2>

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

<Record Cost1>

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

<Record Cost3>

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

<Supervise>

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

<Update>

Field Name	Data Type	Size (Byte)c	Constraints
<u>Transaction-num</u>	INT	5	FK, Primary Key, Not null
<u>Account number</u>	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 Organization	File	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 retrieve 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 search	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 retrieve 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 blobs 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 passthruhg(
    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)
);
```

Messages

```
1:30:55 PM      Started executing query at Line 1
                  Commands completed successfully.
                  Total execution time: 00:00:00.095
```

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.

```
1  INSERT INTO process (processId,processData)
2  VALUES ('999', 'zzz')
3  INSERT INTO fit (fittype,processId)
4  VALUES ('besttt', '999')
5  INSERT INTO paint (painttype,paintingmethod,processId)
6  VALUES ('ttt', 'nnn', '999')
7
```

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.

```
1  INSERT INTO department (departmentnumber, departmentdata)
2  VALUES ('34', 'photo');
3
4
```

Messages

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

Query 4.

```
▶ Run □ Cancel ⚙ Disconnect ⚙ Change Connection cs-dsa-4513-sql-db ✓ | |  
1  INSERT into assembly (Assemblyid,dateordered,AssemblyDeatails)  
2  VALUES ('444', '11112011', 'try');  
3  INSERT into order1 (Assemblyid,cname)  
4  VALUES ('444', 'ferial')  
5  INSERT into passthruhg (Assemblyid,processId)  
6  VALUES ('444', '55');  
7
```

Messages

```
10:44:03 PM    Started executing query at Line 1  
                (1 row affected)  
                (1 row affected)  
                (1 row affected)  
Total execution time: 00:00:00.155
```

Query 6.

```
1  INSERT into job (jobNumber, dateofjobcompleted, dateofjobcommenced, labortime)  
2  VALUES ('1000', '12121999', '12121990', '50')
```

Messages

```
10:34:15 PM    Started executing query at Line 1  
                (1 row affected)  
Total execution time: 00:00:00.072
```

Query 10.

```
Run Cancel Disconnect Change Connection | cs-dsa-4513-sql-db
1  SELECT DISTINCT processId
2  FROM supervise WHERE departmentnumber =  (111)
3  SELECT DISTINCT jobnomber
4  FROM assign WHERE processId = 10
5  SELECT labortime, dateofjobcompleted
6  FROM Job WHERE jobNumber =  (30)
```

Results Messages

	processId
1	45
2	77
3	99
4	110

	jobnomber

	labortime	dateofjobcompleted
1	45	12122015

Query 11.

```
1 select supervise.processId, departmentnumber  
2 from Assembly, process, supervise  
3 where supervise.processId = process.processId  
4 and Assembly.AssemblyId = 12  
5 order by dateordered 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.

```
1 select processId from supervise  
2 where departmentnumber = (110)  
3 select jobNumber, AssemblyId from assign  
4 where processId = (12)  
5 select job_type from job  
6 where jobnumber = (110) and dateofjobcompleted = (12122015)
```

Results		Messages
	processId	
1	10	
2	12	
3	20	

	jobNumber	AssemblyId

	job_type

Query 13.

```
▶ Run □ Cancel ⚙ Disconnect ⚙ Change Connection | cs-dsa-4513-sql-db      ▾  
1   SELECT * FROM customer WHERE category >= (5) and category < (4)
```

Results Messages

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

Query 14.

```
1   delete from cutjob where jobnomber > (8) and jobnomber < (100)  
2   delete from assign where jobnomber > (8) and jobnomber < (50)
```

Messages

```
10:15:52 PM    Started executing query at Line 1  
                  (2 rows affected)  
                  (1 row affected)  
Total execution time: 00:00:00.092
```

Query 15.

```
1 UPDATE painjob  
2 SET color = ('dark')  
3 WHERE jobnumber = 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.

```
1④ 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
```

```

10 import java.sql.Connection;]
11
12 public class Final1 {
13
14     // Database credentials
15     final static String HOSTNAME = "shah0037-sql-server.database.windows.net";
16     final static String DBNAME = "cs-dsa-4513-sql-db";
17     final static String USERNAME = "shah0037";
18     final static String PASSWORD = "Lenovoe460";
19
20     // Database connection string
21     final static String URL = String.format("jdbc:sqlserver://%;1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
22             HOSTNAME, DBNAME, USERNAME, PASSWORD);
23
24     // Query templates
25     final static String QUERY_TEMPLATE_1 = "INSERT INTO customer " +
26             "VALUES (?, ?, ?);"
27
28     final static String QUERY_TEMPLATE_2 = "INSERT INTO department " +
29             "VALUES (?, ?);"
30     final static String QUERY_TEMPLATE_3 = "INSERT INTO process " +
31             "VALUES (?, ?);"
32     final static String QUERY_TEMPLATE_3_1 = "INSERT INTO fit " +
33             "VALUES (?, ?);"
34     final static String QUERY_TEMPLATE_3_2 = "INSERT INTO paint " +
35             "VALUES (?, ?, ?);"
36     final static String QUERY_TEMPLATE_3_3 = "INSERT INTO cut " +
37             "VALUES (?, ?, ?);"
38     final static String QUERY_TEMPLATE_3_4 = "INSERT into supervise " +
39             "VALUES (?, ?, ?);"
40     final static String QUERY_TEMPLATE_4_1 = "INSERT into assembly " +
41             "VALUES (?, ?, ?);"
42     final static String QUERY_TEMPLATE_4_2 = "INSERT into order1 " +
43             "VALUES (?, ?);"
44     final static String QUERY_TEMPLATE_4_3 = "INSERT into passthru " +
45             "VALUES (?, ?);"
46     final static String QUERY_TEMPLATE_5_1 = "INSERT into account " +
47             "VALUES (?, ?);"
48     final static String QUERY_TEMPLATE_5_2 = "INSERT into departmentaccount " +
49             "VALUES (?, ?);"
50     final static String QUERY_TEMPLATE_5_3 = "INSERT into assemblyaccount " +
51             "VALUES (?, ?);"
52     final static String QUERY_TEMPLATE_5_4 = "INSERT into processaccount " +
53             "VALUES (?, ?);"
54     final static String QUERY_TEMPLATE_5_5 = "INSERT into record_dept_cost " +
55             "VALUES (?, ?);"
56     final static String QUERY_TEMPLATE_5_6 = "INSERT into record_assembly_cost " +
57             "VALUES (?, ?);"
58     final static String QUERY_TEMPLATE_5_7 = "INSERT into record_process_cost " +
59             "VALUES (?, ?);"
60     final static String QUERY_TEMPLATE_6_1 = "INSERT into job " +
61             "VALUES (?, ?, ?, ?, ?);"
62     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 jobNumber, Assemblyid from assign " +
118         "where processId = (?)";
119@     final static String QUERY_TEMPLATE_12_3 = "select job_type from job " +
120         "where jobnumber = (?) 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 jobnumber > (?) and jobnumber < (?) " ;
124     final static String QUERY_TEMPLATE_14_2 = "delete from assign Where jobnumber > (?) and jobnumber < (?) " ;
125     final static String QUERY_TEMPLATE_14_3 = "delete from job Where jobnumber > (?) and jobnumber < (?) " ;
126@     final static String QUERY_TEMPLATE_15 ="UPDATE painjob " +
127         "SET color = (?)"+ 
128         "Where jobnumber = (?);";
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

```

Task 5.2.

```
1⑩ 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
14     // Database credentials
15     final static String HOSTNAME = "shah0037-sql-server.database.windows.net";
16     final static String DBNAME = "cs-dsa-4513-sql-db";
17     final static String USERNAME = "shah0037";
18     final static String PASSWORD = "Lenovo@460#";
19
20     // Database connection string
21     final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=%s.database.windows.net;loginTimeout=30", 
22             HOSTNAME, DBNAME, USERNAME, PASSWORD);
23
24     // Query templates
25     final static String QUERY_TEMPLATE_1 = "INSERT INTO customer " +
26             "VALUES (?, ?, ?);";
27
28     final static String QUERY_TEMPLATE_2 = "INSERT INTO department " +
29             "VALUES (?, ?);";
30     final static String QUERY_TEMPLATE_3 = "INSERT INTO process " +
31             "VALUES (?, ?);";
32     final static String QUERY_TEMPLATE_3_1 = "INSERT INTO fit " +
33             "VALUES (?, ?);";
34     final static String QUERY_TEMPLATE_3_2 = "INSERT INTO paint " +
35             "VALUES (?, ?, ?);";
36     final static String QUERY_TEMPLATE_3_3 = "INSERT INTO cut " +
37             "VALUES (?, ?, ?);";
38     final static String QUERY_TEMPLATE_3_4 = "INSERT into supervise " +
39             "VALUES (?, ?, ?);";
40     final static String QUERY_TEMPLATE_4_1 = "INSERT into assembly " +
41             "VALUES (?, ?, ?);";
42     final static String QUERY_TEMPLATE_4_2 = "INSERT into order1 " +
43             "VALUES (?, ?);";
44     final static String QUERY_TEMPLATE_4_3 = "INSERT into passthrough " +
45             "VALUES (?, ?);";
46     final static String QUERY_TEMPLATE_5_1 = "INSERT into account " +
47             "VALUES (?, ?);";
48     final static String QUERY_TEMPLATE_5_2 = "INSERT into departmentaccount " +
49             "VALUES (?, ?);";
50     final static String QUERY_TEMPLATE_5_3 = "INSERT into assemblyaccount " +
51             "VALUES (?, ?);";
52     final static String QUERY_TEMPLATE_5_4 = "INSERT into processaccount " +
53             "VALUES (?, ?);";
54     final static String QUERY_TEMPLATE_5_5 = "INSERT into record_dept_cost " +
55             "VALUES (?, ?, ?);";
56     final static String QUERY_TEMPLATE_5_6 = "INSERT into record_assembly_cost " +
57             "VALUES (?, ?);";
58     final static String QUERY_TEMPLATE_5_7 = "INSERT into record_process_cost " +
59             "VALUES (?, ?);";
60     final static String QUERY_TEMPLATE_6_1 = "INSERT into job " +
61             "VALUES (?, ?, ?, ?);";
62     final static String QUERY_TEMPLATE_6_2 = "INSERT into assign " +
63             "VALUES (?, ?, ?, ?);";
```

```

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 jobNumber, Assemblyid from assign " +
118         "where processId = (?)";
119@     final static String QUERY_TEMPLATE_12_3 = "select job_type from job " +
120         "where jobnumber = (?) 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 jobnumber > (?) and jobnumber < (?) " ;
124     final static String QUERY_TEMPLATE_14_2 = "delete from assign Where jobnumber > (?) and jobnumber < (?) " ;
125     final static String QUERY_TEMPLATE_14_3 = "delete from job Where jobnumber > (?) and jobnumber < (?) " ;
126@     final static String QUERY_TEMPLATE_15 ="UPDATE painjob " +
127         "SET color = (?)"+ 
128         "Where jobnumber = (?);";
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     sc.nextLine();
167     System.out.println("Please enter customer address:");
168     // Preceding nextInt, nextFloat, etc. do not consume new line characters from the user input.
169     // We call nextLine to consume that newline character, so that subsequent nextLine doesn't return nothing.
170     //sc.nextLine();
171     final String caddress = sc.nextLine(); // Read in user input of student First Name (white-spaces allowed).
172
173     System.out.println("Please enter customer category:");
174     // No need to call nextLine extra time here, because the preceding nextLine consumed the newline character.
175     final float category = sc.nextFloat(); // Read in user input of student Last Name (white-spaces allowed).
176
177     System.out.println("Connecting to the database...");
178     // Get a database connection and prepare a query statement
179     try (final Connection connection = DriverManager.getConnection(URL)) {
180         try {
181             final PreparedStatement statement = connection.prepareStatement(QUERY_TEMPLATE_1));
182             // Populate the query template with the data collected from the user
183             System.out.print(cname);
184             statement.setString(1, cname);
185             statement.setString(2, caddress);
186             statement.setFloat(3, category);
187
188             System.out.println("Dispatching the query...");
189             // Actually execute the populated query
190             final int rows_inserted = statement.executeUpdate();
191             System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
192         }
193     }
194     break;
195 case "2":
196     System.out.println("Please enter department number");
197     final int departmentnumber = sc.nextInt(); // Read in the user input of student ID
198
199     //sc.nextLine();
200     System.out.println("Please enter department data:");
201     // Preceding nextInt, nextFloat, etc. do not consume new line characters from the user input.
202     // We call nextLine to consume that newline character, so that subsequent nextLine doesn't return nothing.
203     sc.nextLine();
204     final String departmentdata = sc.nextLine(); // Read in user input of student First Name (white-spaces allowed).
205
206
207     System.out.println("Connecting to the database...");
208     // Get a database connection and prepare a query statement
209     try (final Connection connection = DriverManager.getConnection(URL)) {
210         try {
211             final PreparedStatement statement = connection.prepareStatement(QUERY_TEMPLATE_2));
212             // Populate the query template with the data collected from the user
213
214             statement.setInt(1, departmentnumber);
215             statement.setString(2, departmentdata);
216
217
218

```

```

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

```

```

270         statement2.setInt(2, processId);
271         statement2.setString(1, fittype);
272
273         System.out.println("Dispatching the query...");
274         // Actually execute the populated query
275         final int rows_inserted = statement2.executeUpdate();
276         System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
277     }
278 }
280 }
281
282 }
283
284 else if (processmodel == 2) {
285     //information for Paint Process
286     System.out.println("Please enter Paint type:");
287     final String painttype = sc.nextLine();
288     //sc.nextLine();
289     System.out.println("Please enter Paint method:");
290     final String paintingmethod = sc.nextLine();
291     //sc.nextLine();
292
293     System.out.println("Connecting to the database...");
294     // Get a database connection and prepare a query statement
295     try (final Connection connection = DriverManager.getConnection(URL)) {
296
297         try {
298             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_3));
299             // Populate the query template with the data collected from the user
300
301             statement1.setInt(1, processId);
302             statement1.setString(2, processData);
303
304             System.out.println("Dispatching the query...");
305             // Actually execute the populated query
306             final int rows_inserted = statement1.executeUpdate();
307             System.out.println(String.format("Done. %d rows inserted in the process table.", rows_inserted));
308         }
309     try {
310         final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_3_2));
311         // Populate the query template with the data collected from the user
312
313         statement2.setInt(3, processId);
314         statement2.setString(1, painttype);
315         statement2.setString(2, paintingmethod);
316         System.out.println("Dispatching the query...");
317         // Actually execute the populated query
318         final int rows_inserted = statement2.executeUpdate();
319         System.out.println("Executing Statement 3(2)...");
320         System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
321     }
322 }
```

```

322
323
324
325
326
327
328
329
330     }
331     else {
332         // information for Cutting
333         System.out.println("Please enter the Cutting Type:");
334         final String cuttingType = sc.nextLine();
335         System.out.println("Please enter the Machine Type:");
336         final String machineType = sc.nextLine();
337         System.out.println("Connecting to the database...");
338         // Get a database connection and prepare a query statement
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         statement4.setInt(1, processId);
375         statement4.setInt(2, deptnum);
376         statement4.setString(3, deptData);
377
378         System.out.println("Dispatching the query 3(4)...");
379         // Actually execute the populated query
380         final int rows_inserted = statement4.executeUpdate();
381         System.out.println(String.format("Done. %d rows inserted in supervise table.", rows_inserted));
382
383     }
384 }
385 break;
386 case "4":
387     System.out.println(" enter a new Assembly ID:");
388     final int assemblyid = sc.nextInt();
389     sc.nextLine();
390     System.out.println("Please enter the Date of ordered of assembly:");
391     final int dateof_ordered = sc.nextInt();
392     sc.nextLine();
393     System.out.println("Please enter the Assembly details:");
394     final String assemblyDetails = sc.nextLine();
395     System.out.println("Please enter the associated Customer Name:");
396     final String assembly_customername = sc.nextLine();
397     System.out.println("Please enter the number of associated process IDs with this assembly");
398     final int numassemblies = sc.nextInt();
399     sc.nextLine();
400     int[] assemblyrep = new int[numassemblies];
401     for (int j = 0; j < numassemblies; j++) {
402     {
403         System.out.println(" enter %dth process ID:");
404         assemblyrep[j] = sc.nextInt();
405         sc.nextLine();
406     }
407
408 try (final Connection connection = DriverManager.getConnection(URL)) {
409     try {
410         final PreparedStatement statement4 = connection.prepareStatement(QUERY_TEMPLATE_4_1)) {
411         // Populate the query template with the data collected from the user
412
413         statement4.setInt(1, assemblyid);
414         statement4.setInt(2, dateof_ordered);
415         statement4.setString(3, assemblyDetails);
416
417         System.out.println("Dispatching the query 4(1)...");
418         // Actually execute the populated query
419         final int rows_inserted = statement4.executeUpdate();
420         System.out.println(String.format("Done. %d rows inserted in Assembly table.", rows_inserted));
421
422     }
423     try {
424         final PreparedStatement statement5 = connection.prepareStatement(QUERY_TEMPLATE_4_2)) {
425         // Populate the query template with the data collected from the user
426

```

```

426 |     statement5.setInt(1, assemblyid);
427 |     statement5.setString(2, assembly_customername);
428 |
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 passthru table.", rows_inserted));
449             }
450         }
451     }
452     break;
453 }
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
479             }
480         }
481     }

```

```
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     }
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);
```

```
530 statement1.setInt(1, accountnumb);
531 statement1.setInt(2, accountdate);
532
533 System.out.println("Dispatching the query 5(a)...");
534 // Actually execute the populated query
535 final int rows_inserted = statement1.executeUpdate();
536 System.out.println(String.format("Done. %d rows inserted in account table.", rows_inserted));
537
538 }
539 try {
540     final PreparedStatement statement9 = connection.prepareStatement(QUERY_TEMPLATE_5_3)) {
541         // Populate the query template with the data collected from the user
542
543         statement9.setInt(1, accountnumb);
544         statement9.setInt(2, 0);
545         System.out.println("Dispatching the query 5(3)...");
546         // Actually execute the populated query
547         final int rows_inserted = statement9.executeUpdate();
548         System.out.println("Executing Statement 5(c)...");
549         System.out.println(String.format("Done. %d rows inserted in assemblyaccount table.", rows_inserted));
550
551     }
552     try {
553         final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_5_6)) {
554             // Populate the query template with the data collected from the user
555             //System.out.print(cname);
556             statement3.setInt(1, accountnumb);
557             statement3.setInt(2, AssemblyID);
558             System.out.println("Dispatching the query 5(6)...");
559             // Actually execute the populated query
560             final int rows_inserted = statement3.executeUpdate();
561             System.out.println("Executing Statement 5(6)...");
562             System.out.println(String.format("Done. %d rows inserted in record assembly cost table.", rows_inserted));
563
564     }
565
566     }
567 }
568
569 }
570 else {
571     // information for Cut Process
572     System.out.println(" enter the associated process ID:");
573     final int processID = sc.nextInt();
574     sc.nextLine();
575
576     System.out.println("Connecting to the database...");
577     // Get a database connection and prepare a query statement
578     try (final Connection connection = DriverManager.getConnection(URL)) {
579
580         try {
581             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_5_1)) {
582                 // Populate the query template with the data collected from the user
583                 // ...
584             }
585         }
586     }
587 }
```

```

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();
635
636 }

```

```

634     final int processID = sc.nextInt();
635     sc.nextLine();
636
637     System.out.println(" enter the Assembly ID of process for this job:");
638     final int assemblyID = sc.nextInt();
639     sc.nextLine();
640
641     try (final Connection connection = DriverManager.getConnection(URL)) {
642         try {
643             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_6_1));
644             // Populate the query template with the data collected from the user
645
646             statement1.setInt(1, jobNum);
647             statement1.setInt(3, datejobCommenced);
648             statement1.setNull(2, Types.NULL);
649             statement1.setNull(4, Types.NULL);
650             statement1.setNull(5, Types.NULL);
651             System.out.println("Dispatching the query 6(1)...");
652             // Actually execute the populated query
653             final int rows_inserted = statement1.executeUpdate();
654             System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));
655
656         } catch (SQLException e) {
657             e.printStackTrace();
658         }
659         finally {
660             statement1.close();
661         }
662         try {
663             final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_6_2));
664             // Populate the query template with the data collected from the user
665
666             statement2.setInt(1, assemblyID);
667             statement2.setInt(3, jobNum);
668             statement2.setInt(2, processID);
669             System.out.println("Dispatching the query 6(2)...");
670             // Actually execute the populated query
671             final int rows_inserted = statement2.executeUpdate();
672             System.out.println(String.format("Done. %d rows inserted in assign table..", rows_inserted));
673         } catch (SQLException e) {
674             e.printStackTrace();
675         }
676         finally {
677             statement2.close();
678         }
679     }
680
681     case "6":
682         System.out.println("Please enter the Job Number which is in case 6:");
683         final int jobnum = sc.nextInt();
684         sc.nextLine();
685
686         System.out.println("Please enter the date the job completed:");
687         final int datejobCompleted = sc.nextInt();
688         sc.nextLine();
689
690         System.out.println("Please enter type of Job(1,2,3): 1.Cut Job, 2. Paint Job , 3.Fit Job");
691         final int type = sc.nextInt();
692         sc.nextLine();
693
694         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         try (
718             final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_2)) {
719             // Populate the query template with the data collected from the user
720             //System.out.print(cname);
721             statement3.setString(1, machinetype);
722             statement3.setString(2, material);
723             statement3.setInt(3, jobnum);
724             statement3.setInt(4, time);
725             statement3.setInt(5, laborTime);
726             System.out.println("Dispatching the query 6(3)...");
727             // Actually execute the populated query
728             final int rows_inserted = statement3.executeUpdate();
729             System.out.println(String.format("Done. %d rows inserted in cutjob table..", rows_inserted));
730         }
731     }
732 }
733 else if (type == 2)
734 {
735     System.out.println("Please enter the color:");
736     final String color = sc.nextLine();
737
738

```

```

738     final String color = sc.nextLine();
739
740     System.out.println("Please enter the volume of paint :");
741     final int volume = sc.nextInt();
742     sc.nextLine();
743
744     System.out.println("Please enter the labor time in hours:");
745     final int laborTime = sc.nextInt();
746     sc.nextLine();
747
748     System.out.println("Connecting to the database...");
749     // Get a database connection and prepare a query statement
750     try (final Connection connection = DriverManager.getConnection(URL)) {
751         try {
752             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_7_1);
753             // Populate the query template with the data collected from the user
754             //System.out.print(cname);
755             statement1.setInt(1, datejobCompleted);
756             statement1.setInt(2, laborTime);
757             statement1.setInt(3, type);
758             statement1.setInt(4, jobnum);
759             System.out.println("Dispatching the query 7(1)...");
760             // Actually execute the populated query
761             final int rows_inserted = statement1.executeUpdate();
762             System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));
763         }
764     }
765
766     try {
767         final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_3);
768         // Populate the query template with the data collected from the user
769         //System.out.print(cname);
770         statement3.setString(1, color);
771         statement3.setInt(2, volume);
772         statement3.setInt(3, jobnum);
773         statement3.setInt(4, laborTime);
774
775         System.out.println("Dispatching the query 7(3)...");
776         // Actually execute the populated query
777         final int rows_inserted = statement3.executeUpdate();
778         System.out.println(String.format("Done. %d rows inserted in painjob table..", rows_inserted));
779     }
780 }
781 }
782
783 else
784 {
785     System.out.println("Please enter the labor time in hours:");
786     final int laborTime = sc.nextInt();
787     sc.nextLine();
788     System.out.println("Connecting to the database...");
789     // Get a database connection and prepare a query statement
790     try (final Connection connection = DriverManager.getConnection(URL)) {
791

```

```

790     try (final Connection connection = DriverManager.getConnection(URL)) {
791         try {
792             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_7_1));
793             // Populate the query template with the data collected from the user
794
795             statement1.setInt(1, datejobCompleted);
796             statement1.setInt(2, laborTime);
797             statement1.setInt(3, type);
798             statement1.setInt(4, jobnum);
799             System.out.println("Dispatching the query 7(1)...");
800             // Actually execute the populated query
801             final int rows_inserted = statement1.executeUpdate();
802             System.out.println(String.format("Done. %d rows inserted in Job table..", rows_inserted));
803
804         }
805
806         try {
807             final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_7_4));
808             // Populate the query template with the data collected from the user
809             //System.out.print(cname);
810
811             statement3.setInt(1, jobnum);
812             statement3.setInt(2, laborTime);
813
814             System.out.println("Dispatching the query 7(4)...");
815             // Actually execute the populated query
816             final int rows_inserted = statement3.executeUpdate();
817             System.out.println(String.format("Done. %d rows inserted in fitjob table..", rows_inserted));
818
819         }
820     }
821     break;
822
823 case "11":
824     System.out.println("Please enter the assembly ID which has passed so far :");
825     final int aID = sc.nextInt();
826     sc.nextLine();
827
828     try (final Connection connection = DriverManager.getConnection(URL)) {
829         try {
830             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_11));
831             statement1.setInt(1, aID);
832             System.out.println("Dispatching the query 11...");
833             ResultSet rs1 = statement1.executeQuery();
834             System.out.println("Process ID Department Number");
835             while(rs1.next()) {
836                 System.out.println(rs1.getInt(1) + "\t" + rs1.getInt(2));
837             }
838         }
839     }
840     break;
841
842 case "12":
843     System.out.println("Please enter the date that job completed:");
844

```

```

842     System.out.println("Please enter the date that job completed:");
843     final int jobcomp = sc.nextInt();
844     sc.nextLine();
845     System.out.println("Please enter the Department for the job:");
846     final int depjob = sc.nextInt();
847     sc.nextLine();
848     System.out.println("Connecting to the database...");
849     try (final Connection connection = DriverManager.getConnection(URL)) {
850         try (final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_12_1)) {
851             statement1.setInt(1, depjob);
852             System.out.println("Dispatching the query 12(1)...");
853             ResultSet rs1 = statement1.executeQuery();
854             while(rs1.next()) {
855                 int processidd = rs1.getInt(1);
856                 try (final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_12_2)) {
857                     statement2.setInt(1, processidd);
858                     System.out.println("Dispatching the query 12(2)");
859                     ResultSet rs2 = statement2.executeQuery();
860                     while(rs2.next()) {
861                         int jobid = rs2.getInt(1);
862                         int assemblyID = rs2.getInt(2);
863                         try (final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_12_3)) {
864                             statement3.setInt(1, jobid);
865                             statement3.setInt(2, jobcomp);
866                             System.out.println("Dispatching query 12(3)");
867                             ResultSet rs3 = statement3.executeQuery();
868                             while(rs3.next()) {
869                                 int Type1 = rs3.getInt(1);
870                                 if(Type1 == 1) {
871                                     System.out.println(jobid + "\t" + assemblyID + "\t" + "Cut Job");
872                                 }
873                                 else if(Type1 == 2) {
874                                     System.out.println(jobid + "\t" + assemblyID + "\t" + "Paint Job");
875                                 }
876                                 else
877                                     System.out.println(jobid + "\t" + assemblyID + "\t" + "Fit Job");
878                             }
879                         }
880                     }
881                 }
882             }
883         }
884     }
885     break;
886 case "10":
887     System.out.println("Please enter the Department number that you want find the labor cost:");
888     final int department = sc.nextInt();
889     sc.nextLine();
890     System.out.println("Please enter the date of completion for the jobs:");
891     final int comdate = sc.nextInt();
892     sc.nextLine();
893     ...
894 ---
```

```

894     sc.nextLine();
895     // Determining the Processes supervised by the given Department:
896     try (final Connection connection = DriverManager.getConnection(URL)) {
897         try {
898             final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_10_1);
899             // Populate the query template with the data collected from the user
900             //System.out.print(sname);
901             statement1.setInt(1, department);
902             System.out.println("Dispatching the query 1...");
903             // Actually execute the populated query
904             ResultSet rs = statement1.executeQuery();
905             int psize = 0;
906             int jsize = 0;
907             int[] iarr = new int[77];
908             int[] jjj = new int[77];
909             int pindex = 0;
910             int jindex = 0;
911             while (rs.next()) {
912                 int prcid = rs.getInt("processId");
913                 iarr[pindex] = prcid;
914                 pindex++;
915                 psizes++;
916             }
917             for (int i = 0; i < psizes; i++) {
918                 {
919                     System.out.println(iarr[i]);
920                     try {
921                         final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_10_2);
922                         statement2.setInt(1, iarr[i]);
923                         System.out.println("Dispatching the query 10(2)...");
924                         ResultSet rsl = statement2.executeQuery();
925                         int jobN = 0;
926                         while(rsl.next()) {
927                             int jobno = rsl.getInt("jobnomber");
928                             jjj[jindex] = jobno;
929                             jindex++;
930                         }
931                     }
932                 }
933                 int[] rrr = new int[jindex];
934                 int[] trrr = new int[jindex];
935                 for(int k = 0; k < jindex; k++) {
936                     System.out.println(jjj[k]);
937                     try {
938                         final PreparedStatement statement4 = connection.prepareStatement(QUERY_TEMPLATE_10_3);
939                         statement4.setInt(1, jjj[k]);
940                         System.out.println("Dispatching the query 10(c)...");
941                         ResultSet rs2 = statement4.executeQuery();
942                         while(rs2.next()) {
943                             trrr[k] = rs2.getInt("labortime");
944                             rrr[k] = rs2.getInt("dateofjobcompleted");
945                         }
946                     }
947                 }
948             }
949         }
950     }

```

```

946                               rrr[k] = rs2.getInt("dateofjobcompleted");
947
948                           }
949                         }
950                         int totallaborTime = 0;
951                         for (int l = 0; l < jindex; l++) {
952                             if(rrr[l] == comdate)
953                                 totallaborTime += trtr[l];
954                         }
955                         System.out.println("The total labor time for " + department + " department for jobs completed during the " +
956                         comdate + " date is: " + totallaborTime + " hours.");
957                         }
958                     }
959                     break;
960             case "13":
961                 System.out.println("enter min for category:");
962                 final int minimum = sc.nextInt();
963                 sc.nextLine();
964                 System.out.println("enter max for category::");
965                 final int maximum = sc.nextInt();
966                 sc.nextLine();
967                 System.out.println("Connecting to the database...");
968                 // Get the database connection, create statement and execute it right away, as no user input need be collected
969                 try (final Connection connection = DriverManager.getConnection(URL)) {
970                     System.out.println("Dispatching the query...");
971                     try (
972                         final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_13)) {
973
974                         statement2.setInt(1, minimum);
975                         statement2.setInt(2, maximum);
976                         System.out.println("Dispatching the query 13...");
977                         ResultSet rs1 = statement2.executeQuery();
978                         System.out.println("Contents of the Student table:");
979                         System.out.println("name | address | Category");
980
981                         // Unpack the tuples returned by the database and print them out to the user
982                         while (rs1.next()) {
983
984                             String name = rs1.getString("cname");
985                             String address = rs1.getString("caddress");
986                             int cat = rs1.getInt("category");
987                             System.out.println(name + "\t" + address + "\t\t" + cat);
988                             //resultSet.getString(2),
989                             //resultSet.getString(3)
990
991                         }
992                     }
993                 }
994                 break;
995             case "14":
996                 System.out.println("minimum for job number:");
997
998

```

```

998     System.out.println("minimum for job number:");
999     final int minijobnum = sc.nextInt();
1000    sc.nextLine();
1001    System.out.println("maximum for job number:");
1002    final int maxjobnum = sc.nextInt();
1003    sc.nextLine();
1004    System.out.println("Connecting to the database...");
1005    // Get the database connection, create statement and execute it right away, as no user input need be collected
1006    try (final Connection connection = DriverManager.getConnection(URL)) {
1007        try {
1008            final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_14_1)) {
1009                // Populate the query template with the data collected from the user
1010
1011                statement1.setInt(1, minijobnum);
1012                statement1.setInt(2, maxjobnum);
1013                System.out.println("Dispatching the query 14(1)...");
1014                // Actually execute the populated query
1015                final int rows_inserted = statement1.executeUpdate();
1016                System.out.println(String.format("Done. %d rows deleted in cutjob table..", rows_inserted));
1017            }
1018
1019            try {
1020                final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_14_2)) {
1021                    // Populate the query template with the data collected from the user
1022
1023                    statement2.setInt(1, minijobnum);
1024                    statement2.setInt(2, maxjobnum);
1025                    System.out.println("Dispatching the query 14(2)...");
1026                    // Actually execute the populated query
1027                    final int rows_inserted = statement2.executeUpdate();
1028                    System.out.println(String.format("Done. %d rows deleted in assign table..", rows_inserted));
1029            }
1030
1031            try {
1032                final PreparedStatement statement3 = connection.prepareStatement(QUERY_TEMPLATE_14_3)) {
1033                    // Populate the query template with the data collected from the user
1034
1035                    statement3.setInt(1, minijobnum);
1036                    statement3.setInt(2, maxjobnum);
1037                    System.out.println("Dispatching the query 14(3)...");
1038                    // Actually execute the populated query
1039                    final int rows_inserted = statement3.executeUpdate();
1040                    System.out.println(String.format("Done. %d rows deleted in job table..", rows_inserted));
1041            }
1042        }
1043        break;
1044    case "15":
1045        System.out.println("enter job number that you want change color:");
1046        final int jobnumm = sc.nextInt();
1047        sc.nextLine();
1048        System.out.println("enter new color:");
1049        final String color = sc.nextLine();
1050        System.out.println("Connecting to the database...");
1051        // Get the database connection, create statement and execute it right away, as no user input need be collected

```

```

1050 // Get the database connection, create statement and execute it right away, as no user input need be collected
1051
1052 try (final Connection connection = DriverManager.getConnection(URL)) {
1053     try {
1054         final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_15);
1055         // Populate the query template with the data collected from the user
1056         statement1.setString(1, color);
1057         statement1.setInt(2, jobnumm);
1058         System.out.println("Dispatching the query 15...");
1059         // Actually execute the populated query
1060         final int rows_inserted = statement1.executeUpdate();
1061         System.out.println(String.format("Done. %d rows updated in painjob table..", rows_inserted));
1062     }
1063     break;
1064 case "16":
1065     System.out.println("Where is path of file you want to enter?:");
1066     sc.nextLine();
1067     final String file = sc.nextLine();
1068     BufferedReader br;
1069     String line = "";
1070     String split = ",";
1071     try {
1072         br = new BufferedReader(new FileReader(file));
1073         try (final Connection connection = DriverManager.getConnection(URL)) {
1074             while ((line = br.readLine()) != null)
1075             {
1076                 String[] customer = line.split(split);
1077
1078                 try {
1079                     final PreparedStatement statement1 = connection.prepareStatement(QUERY_TEMPLATE_1);
1080                     // Populate the query template with the data collected from the user
1081                     statement1.setString(1, customer[0]);
1082                     statement1.setString(2, customer[1]);
1083                     statement1.setInt(3, Integer.parseInt(customer[2]));
1084
1085                     System.out.println("Dispatching the query 1...");
1086
1087                     final int rows_inserted = statement1.executeUpdate();
1088                     System.out.println(String.format("Done. %d rows inserted in customer table..", rows_inserted));
1089
1090                 }
1091             }
1092             br.close();
1093         }
1094     } catch (FileNotFoundException e)
1095     {
1096
1097         System.out.println("not found!");
1098         e.printStackTrace();
1099         break;
1100     } catch (IOException e) {
1101         System.out.println("erooooooooor");
1102     }

```

```

1102     System.out.println("eroooooor");
1103     e.printStackTrace();
1104     break;
1105 }
1106 break;
1107 case "17":
1108     System.out.println("enter min for category:");
1109     final int mincategory = sc.nextInt();
1110     sc.nextLine();
1111     System.out.println("enter max for category:");
1112     final int maxcategory = sc.nextInt();
1113     sc.nextLine();
1114     System.out.println("enter output file name");
1115     final String fileName = sc.nextLine();
1116     System.out.println("Connecting to the database...");
1117     try (final Connection connection = DriverManager.getConnection(URL)) {
1118         System.out.println("Dispatching the query...");
1119         try {
1120             final PreparedStatement statement2 = connection.prepareStatement(QUERY_TEMPLATE_13)) {
1121
1122                 statement2.setInt(1, mincategory);
1123                 statement2.setInt(2, maxcategory);
1124                 System.out.println("Dispatching the query 13...");
1125                 ResultSet tre = statement2.executeQuery();
1126
1127                 try (PrintWriter rrr = new PrintWriter(fileName + ".csv")) {
1128                     while (tre.next()) {
1129                         StringBuilder sub = new StringBuilder();
1130                         String name = tre.getString("cname");
1131                         sub.append(name);
1132                         sub.append(',');
1133                         String address = tre.getString("caddress");
1134                         sub.append(address);
1135                         sub.append(',');
1136                         int cat = tre.getInt("category");
1137                         sub.append(cat);
1138                         sub.append('\n');
1139                         rrr.write(sub.toString());
1140                     }
1141                 }catch (FileNotFoundException e)
1142                 {
1143                     System.out.println(e.getMessage());
1144                 }
1145             }
1146         }
1147     }
1148 }
1149 break;
1150 case "18":
1151     System.out.println("Exiting! bye!");
1152     System.exit(0);
1153 default:
1154     System.out.println(String.format(
1155             System.out.println(String.format(
1156                 "Unrecognized option: %s\n" +
1157                 "Please try again!",
1158                 option)));
1159     break;
1160 }
1161 }
1162 sc.close();
1163 }
1164 }
1165

```

Final [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 passthrough table.
Dispatching the query 4(3)...
Done. 1 rows inserted in passthrough table.
```

Assembly Table :

	Assemblyid	dateordered	AssemblyDetails
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:

	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:

	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
```

	jobNumber	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
```

	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
```

	jobNumber	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
```

	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
```

	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

	jobnumber	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 department 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 department 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 department 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:

The screenshot shows a database interface with a query editor and a results grid.

Query Editor (Top):

```
1  select * from cutjob
```

Results Grid (Bottom):

	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	jobnumber	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 painjob table before run.

The screenshot shows a database interface with a query editor at the top containing the SQL command: `1 select * from painjob`. Below the editor is a results grid titled "Results". The grid has four columns: color, volume, jobnomber, and labortime. There is one row of data: color is red, volume is 67, jobnomber is 60, and labortime is 90.

	color	volume	jobnomber	labortime
1	red	67	60	90

Run 1:

The screenshot shows a command-line session. The user runs query 15 and is prompted to enter a job number to change its color. They enter 60 and then provide a new color, blue. The system connects to the database, dispatches the query, and informs the user that 1 row was updated in the painjob table.

```
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	jobnumber	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	jobnumber	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	jobnumber	labortime
1	black	67	60	90

6.16. Screenshots showing the testing of query 16

1 run for query 16:

```
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!
16
Where is path of file you want to enter?:
E:\cust_names.csv
Dispatching the query 1...
Done. 1 rows inserted in customer table..
Dispatching the query 1...
Done. 1 rows inserted in customer table..
Dispatching the query 1...
Done. 1 rows inserted in customer table..
Dispatching the query 1...
Done. 1 rows inserted in customer table..
```

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

	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											
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															

cust_names

Ready

```
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...
```

The screenshot shows an Excel spreadsheet titled "customertest.csv - Excel (Product Activation Failed)". The spreadsheet contains a single sheet named "customertest". The data is organized in a table with columns A through O. Row 1 contains the column headers. Rows 2 through 7 contain the data entries. The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ebrahim	Chicago	2												
2	ferial	oklahoma	2												
3	mary	alberta	6												
4	moosa	Iran	3												
5	Naeem	Norman	3												
6	saeed	azizi	3												
7	separandar	India	4												
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															

6.17. Screenshots showing the testing of query 17

Error checking.

```
18) Quit!
0
Please enter new Job Number:
30
Please enter the date of job commenced:
123
Please enter the Process ID of to this job:
12
    enter the Assembly ID of process for this job:
22
Dispatching the query 6(1)...Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PKHKEY constraint "PK_Job". Cannot insert duplicate key in object 'dbo.Job'. The duplicate key value is (30).
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.getLexResult(SQLServerStatement.java:1662)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:615)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$RepStmExecCmd.doExecute(SQLServerPreparedStatement.java:537)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(IOBuffer.java:3488)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerStatement.java:262)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:237)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.java:483)
at final.main(Final.java:854)
```

```
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:
saed
Please enter customer address:
sdf
Please enter customer category:
2
Connecting to the database...
saedDispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The value is not set for the parameter number 2.
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.buildParamTypeDefinitions(SQLServerPreparedStatement.java:438)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.buildPreparedStrings(SQLServerPreparedStatement.java:391)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:569)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$RepStmExecCmd.doExecute(SQLServerPreparedStatement.java:537)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7417)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3488)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:262)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:237)
at com.microsoft.sqlserver.jdbc@9.4.0.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.java:483)
at final.main(Final.java:191)
```

```

terminated> Final1 [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (Nov 21, 2021, 00:02:49 PM - 00:03:00 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...
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:

```

▶ Run □ Cancel ⚙ Disconnect ⓘ Change Connection cs-dsa-4513-sql-db ▾ | Explain ⚑ Enable SQLCMD ⓘ Export as Notebook
1 delete from job where jobnumber > (8) and jobnumber < (100)
2 delete from cutjob where jobnumber > (8) and jobnumber < (100)
3 delete from assign where jobnumber > (8) and jobnumber < (50)

```

Messages

```

10:17:16 PM Started executing_query_at_Line_3
Msg 547, Level 16, State 0, Line 1
The DELETE statement conflicted with the REFERENCE constraint "FK__assign__jobnombe__2FCF1ABA". The conflict occurred in database "cs-dsa-4513-sql-db", table "dbo.assign", column 'jobnumber'.
The statement has been terminated.
(0 rows affected)
(0 rows affected)
Total execution time: 00:00:00.087

```

```
▶ Run □ Cancel ⌘ Change Connection cs-dsa-4513-sql-db ▾ | ⚖ Explain ⚙ Enable SQLCMD ↗ Export as Notebook
1  INSERT into assembly (AssemblyId,dateordered,AssemblyDetails)
2  VALUES ('444','11112011', 'try');
3  INSERT into order1 (AssemblyId,cname)
4  VALUES ('444', 'ferial')
5  INSERT into passthru (AssemblyId,processId)
6  VALUES (444, 55);
7
```

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_03340F33D8C6AAA7'. 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_DB4388E21804372C'. 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.passthru'. 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:

```
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!
18
Exiting! bye!
```

Task 7. Web database application and its execution

7.1. Web database application source program and screenshots

The screenshot shows a Java-based web application project structure and its corresponding code in a code editor. The project structure includes JAX-WS Web Services, IRE System Library, src/main/java (containing DataHandler.java and HandJava), Server Runtime (Apache Tomcat v10.0), build, src (main, java, webapp, META-INF, WEB-INF, containing add_customer.jsp, add_movie_form.jsp, add_movie.jsp, get_all_movies.jsp, selectcustomer.jsp), and Servers (Tomcat v10.0 Server at localhost-config). The code editor displays the JSP file 'add_customer.jsp' with line numbers 1 through 50. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <title>Add to customer</title>
6     </head>
7     <body>
8         <h2>Add to customer</h2>
9         <!--
10            Form for collecting user input for the new movie_night record.
11            Upon form submission, add_movie.jsp file will be invoked.
12        -->
13         <form action="add_movie.jsp">
14             <!-- The form organized in an HTML table for better clarity. -->
15             <table border=1>
16                 <tr>
17                     <th colspan="2">Enter the customer Data:</th>
18                 </tr>
19                 <tr>
20                     <td>cname:</td>
21                     <td><div style="text-align: center;">
22                         <input type="text" name="cname">
23                     </div></td>
24                 </tr>
25                 <tr>
26                     <td>caddress:</td>
27                     <td><div style="text-align: center;">
28                         <input type="text" name="caddress">
29                     </div></td>
30                 </tr>
31                 <tr>
32                     <td>category:</td>
33                     <td><div style="text-align: center;">
34                         <input type="text" name="category">
35                     </div></td>
36                 </tr>
37                 <tr>
38                     <td><div style="text-align: center;">
39                         <input type="reset" value="Clear">
40                     </div></td>
41                     <td><div style="text-align: center;">
42                         <input type="submit" value="Insert">
43                     </div></td>
44                 </tr>
45             </table>
46         </form>
47     </body>
48 </html>
```

```

v jsp_azure_test
> JAX-WS Web Services
> JRE System Library [JavaSE-11]
v src/main/java
  jsp_azure_test
    > DataHandler.java
    > Hand.java
> Server Runtime [Apache Tomcat v10.0]
  build
v src
  main
    > java
      webapp
        > META-INF
        > WEB-INF
          add_customer.jsp
          add_movie_form.jsp
          add_movie.jsp
          get_all_movies.jsp
          selectcustomer.jsp
v Servers
  Tomcat v10.0 Server at localhost-config

```

```

1_ <!DOCTYPE html>
2@ <html>
3@   <head>
4@     <meta charset="UTF-8">
5@     <title>Add to customer</title>
6@   </head>
7@   <body>
8@     <h2>Add to customer</h2>
9@     <!--
10@       Form for collecting user input for the new movie_night record.
11@       Upon form submission, add_movie.jsp file will be invoked.
12@     -->
13@     <form action="add_movie.jsp">
14@       <!-- The form organized in an HTML table for better clarity. -->
15@       <table border=1>
16@         <tr>
17@           <th colspan="2">Enter the customer Data:</th>
18@         </tr>
19@         <tr>
20@           <td>cname:</td>
21@           <td><div style="text-align: center;">
22@             <input type="text name=cname">
23@           </div></td>
24@         </tr>
25@         <tr>
26@           <td>caddress:</td>
27@           <td><div style="text-align: center;">
28@             <input type="text name=caddress">
29@           </div></td>
30@         </tr>
31@         <tr>
32@           <td>category:</td>
33@           <td><div style="text-align: center;">
34@             <input type="text name=category">
35@           </div></td>
36@         </tr>
37@         <tr>
38@           <td><div style="text-align: center;">
39@             <input type="reset value=Clear">
40@           </div></td>
41@           <td><div style="text-align: center;">
42@             <input type="submit value=Insert">
43@           </div></td>
44@         </tr>
45@       </table>
46@     </form>
47@   </body>
48@ </html>
50

```

```

v jsp_azure_test
> JAX-WS Web Services
> JRE System Library [JavaSE-11]
v src/main/java
  jsp_azure_test
    > DataHandler.java
    > Hand.java
> Server Runtime [Apache Tomcat v10.0]
  build
v src
  main
    > java
      webapp
        > META-INF
        > WEB-INF
          add_customer.jsp
          add_movie_form.jsp
          add_movie.jsp
          get_all_movies.jsp
          selectcustomer.jsp
v Servers
  Tomcat v10.0 Server at localhost-config

```

```

6@ <head>
7@   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8@   <title>Query Result</title>
9@ </head>
10@ <body>
11@   <%@page import="jsp_azure_test.DataHandler"%>
12@   <%@page import="java.sql.ResultSet"%>
13@   <%@page import="java.sql.Array"%>
14@   %
15@   // The handler is the one in charge of establishing the connection.
16@   DataHandler handler = new DataHandler();
17@   %
18@   // Get the attribute values passed from the input form.
19@   String cname = request.getParameter("cname");
20@   String caddress = request.getParameter("caddress");
21@   String category = request.getParameter("category");
22@   %
23@   /*
24@    * If the user hasn't filled out all the time, movie name and duration. This is very simple checking.
25@    */
26@   if (cname.equals("") || caddress.equals("") || category.equals("")) {
27@     response.sendRedirect("add_movie_form.jsp");
28@   } else {
29@     // int duration = Integer.parseInt(durationString);
30@     %
31@     // Now perform the query with the data from the form.
32@     boolean success = handler.addMovie(cname, caddress, category);
33@     if (!success) { // Something went wrong
34@       %
35@       <h2>There was a problem inserting the course</h2>
36@     } else { // Confirm success to the user
37@       %
38@       <h2>The Customer Table:</h2>
39@       %
40@       <ul>
41@         <li>cname: <%cname%></li>
42@         <li>caddress: <%caddress%></li>
43@         <li>category: <%category%></li>
44@       </ul>
45@       %
46@       <h2>New Customer successfully inserted.</h2>
47@       %
48@       <a href="get_all_movies.jsp">See all Customers.</a>
49@       %
50@     }
51@   }
52@   %
53@ }
54@ %
55@ </body>
56@ </html>
58

```

jsp_azure_test

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4<%@ html>
5<%@ head>
6   <meta charset="UTF-8">
7   <title>Movie Nights</title>
8 </head>
9<%@ body>
10  <%@page import="jsp_azure_test.DataHandler"%>
11  <%@page import="java.sql.ResultSet"%>
12<%>
13  // We instantiate the data handler here, and get all the movies from the database
14  final DataHandler handler = new DataHandler();
15  final ResultSet movies = handler.getAllMovies();
16<%>
17  <!-- The table for displaying all the movie records -->
18<%@ table cellspacing="2" cellpadding="2" border="1">
19<%@ tr <!-- The table headers row -->
20<%@ td align="center">
21   <h4>name</h4>
22<%@ td align="center">
23   <h4>address</h4>
24<%@ td align="center">
25   <h4>category</h4>
26<%@ td>
27<%@ /td>
28<%@ /tr>
29<%>
30<%>
31  while(movies.next()) { // For each movie_night record returned...
32    // Extract the attribute values for every row returned
33    final String cname = movies.getString("cname");
34    final String caddress = movies.getString("caddress");
35    final String category = movies.getString("category");
36<%>
37
38    out.println("<tr>"); // Start printing out the new table row
39    out.println( // Print each attribute value
40      "<td align='center'>" + cname +
41      "</td><td align='center'>" + caddress +
42      "</td><td align='center'>" + category +
43      "</td>");
44    out.println("</tr>");
45  }
46<%>
47<%>
48<%@ /table>
49<%@ /body>
50<%@ html>
51<%>
```

jsp_azure_test

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4<%@ html>
5<%@ head>
6   <meta charset="UTF-8">
7   <title>Movie Nights</title>
8 </head>
9<%@ body>
10  <%@page import="jsp_azure_test.Hand"%>
11  <%@page import="java.sql.ResultSet"%>
12<%>
13  // We instantiate the data handler here, and get all the movies from the database
14  final Hand handler = new Hand();
15  final ResultSet movies = handler.selectcustomer();
16<%>
17  <!-- The table for displaying all the movie records -->
18<%@ table cellspacing="2" cellpadding="2" border="1">
19<%@ tr <!-- The table headers row -->
20<%@ td align="center">
21   <h4>name</h4>
22<%@ td align="center">
23   <h4>address</h4>
24<%@ td align="center">
25   <h4>category</h4>
26<%@ td>
27<%@ /td>
28<%@ /tr>
29<%>
30<%>
31  while(movies.next()) { // For each movie_night record returned...
32    // Extract the attribute values for every row returned
33    final String cname = movies.getString("cname");
34    final String caddress = movies.getString("caddress");
35    final String category = movies.getString("category");
36<%>
37
38    out.println("<tr>"); // Start printing out the new table row
39    out.println( // Print each attribute value
40      "<td align='center'>" + cname +
41      "</td><td align='center'>" + caddress +
42      "</td><td align='center'>" + category +
43      "</td>");
44    out.println("</tr>");
45  }
46<%>
47<%>
48<%@ /table>
49<%@ /body>
50<%@ html>
51<%>
```

7.2 Screenshots showing the testing of the Web database application

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

The screenshot shows the Eclipse IDE interface. On the left, the 'Project Explorer' view displays the project structure for 'jsp_azure_test'. It includes 'JAX-WS Web Services', 'JRE System Library [JavaSE-11]', 'src/main/java' (containing 'jsp_azure_test' package with 'DataHandler.java' and 'Hand.java'), 'src' (containing 'main' with 'java' and 'webapp' folders), 'Servers' (showing 'Tomcat v10.0 Server at localhost-config' is started and synchronized), and 'Build Path' (with 'Apache Tomcat v10.0'). The 'webapp/WEB-INF' folder under 'src/main/webapp' contains several JSP files: 'add_customer.jsp', 'add_movie_form.jsp', 'add_movie.jsp', 'get_all_movies.jsp', and 'selectcustomer.jsp'. On the right, a browser window shows the output of the 'get_all_movies.jsp' page. The page displays a table with three columns: 'cname', 'address', and 'category'. The data is as follows:

cname	address	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
mohamand	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.

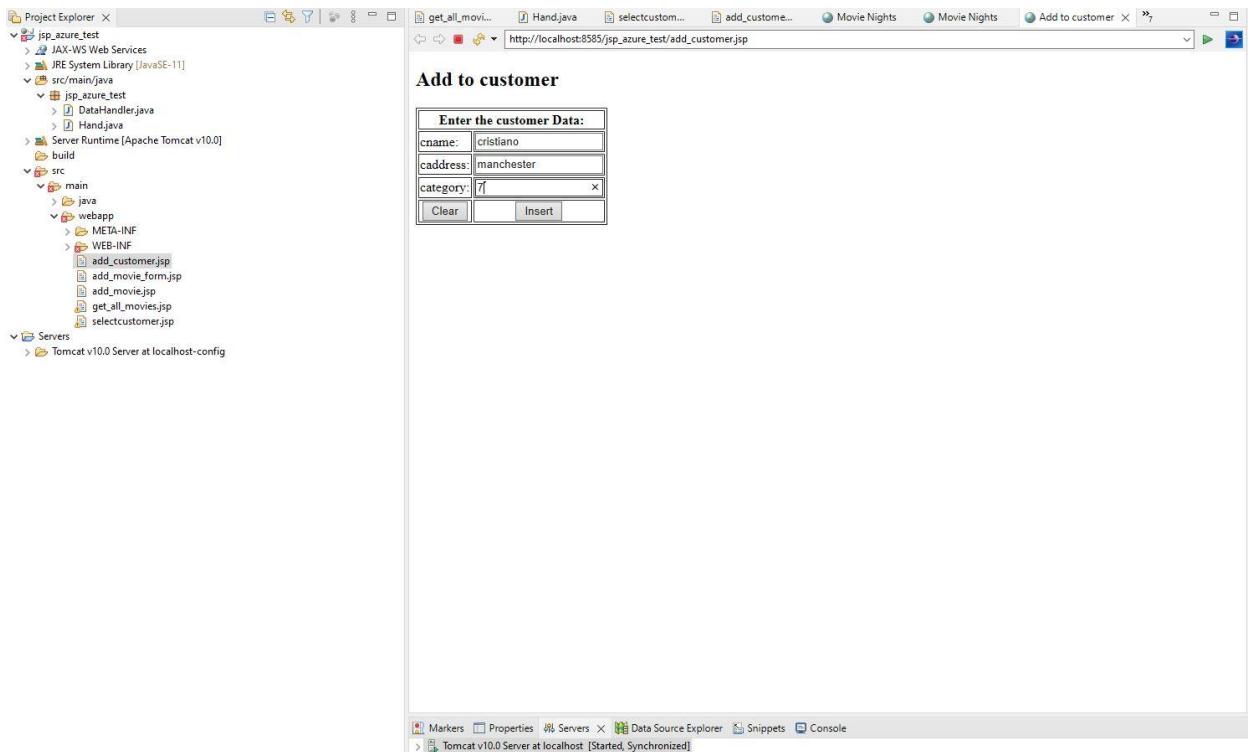
The screenshot displays the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "jsp_azure_test". It includes a "src/main/java" folder containing "jsp_azure_test" with files "DataHandler.java" and "Hand.java", and a "Server Runtime [Apache Tomcat v10.0]" entry.
- Browser:** Displays the URL http://localhost:8585/jsp_azure_test/selectcustomer.jsp. The page content is a table with the following data:

cname	caddress	category
Ali	bouston	9
david	broklyn	9
Hamon	abq	9
hesam	boston	8
jorj	newyork	7
mary	alberta	6
sahra	yazd	8

Bottom Status Bar: Shows tabs for Markers, Properties, Servers, Data Source Explorer, Snippets, and Console. The Servers tab indicates "Tomcat v10.0 Server at localhost [Started, Synchronized]."

Run 1 and add new customer with category 7



The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer view displays a Java project named "jsp_azure_test". It contains JAX-WS Web Services, a JRE System Library [JavaSE-11], a src/main/java folder with a jsp_azure_test package containing DataHandler.java and Hand.java, a Server Runtime [Apache Tomcat v10.0] build configuration, and a src folder with main and webapp subfolders. The webapp folder contains META-INF and WEB-INF subfolders with several JSP files: add_customer.jsp, add_movie_form.jsp, add_movie.jsp, get_all_movies.jsp, and selectcustomer.jsp. On the right, a browser window shows the URL http://localhost:8585/jsp_azure_test. The page displays a table with columns: cname, caddress, and category. The data is as follows:

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	yazzd	8
sepandar	India	4

Run Query 13 again for period between 6 and 9.

The screenshot shows the Eclipse IDE interface. The Project Explorer view is identical to the previous one, displaying the "jsp_azure_test" project structure. On the right, a browser window shows the URL http://localhost:8585/jsp_azure_test/selectcustomer.jsp. The page displays a table with columns: cname, caddress, and category. The data is as follows:

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

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