Job-Shop Accounting System
FALL 2019 Individual Project

Zane Gray, zane.j.gray@ou.edu, 113248429

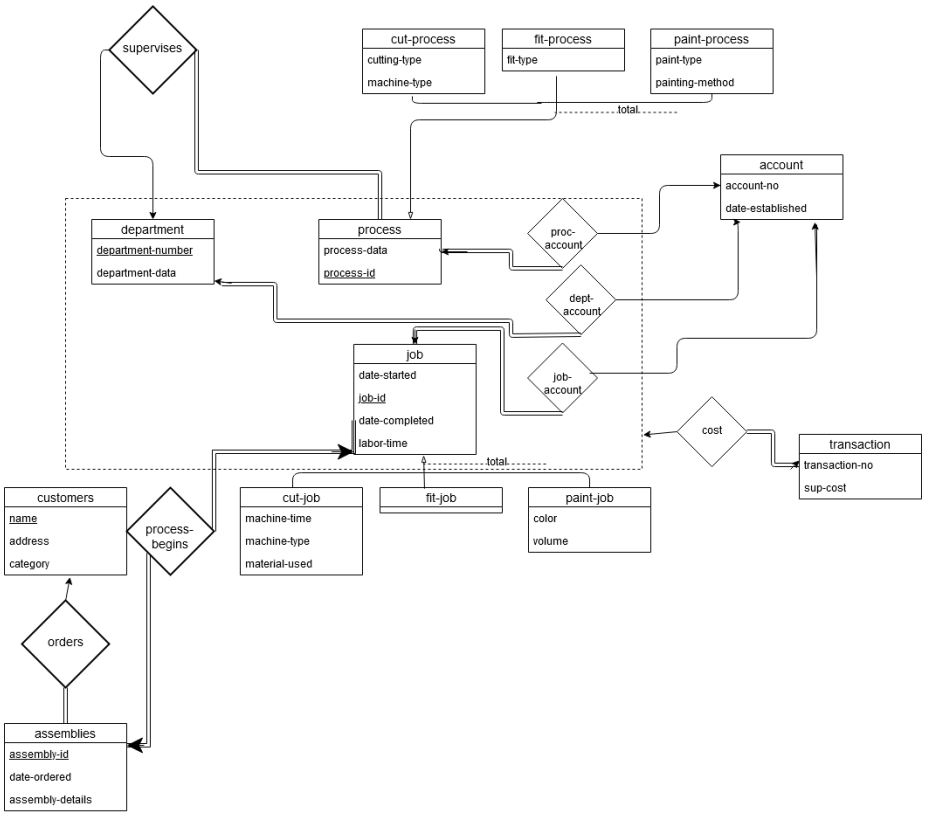
DR Le Gruenwald, CS 4513-001

November 19, 2019

Contents

1 Task 1.

1.1 ER Diagram



1.2 Relational Database Schema

department(department-number,department-data)

customer(name,address,category)

assemblies(assembly-id,name,date-ordered,assembly-details)

transaction(transaction-no,sup-cost,job-account-no,dept-account-no,proc-account-no)

job-account(account-no,date-established,process-id)

dept-account(account-no,date-established,department-number)

proc-account(account-no,date-established,process-id)

fit-job(job-id,date-started,date-completed,labor-time)

cut-job(machine-type,machine-time,material-used,job-id,date-started,date-completed,labor-time)

paint-job(job-id,date-started,date-completed,labor-time,color,volume)

fit-process(fit-process-id,process-data,fit-type,department-number)

cut-process(fit-process-id,process-data,cutting-type,machine-type,department-number)

paint-process(fit-process-id,process-data,paint -type,painting-method,department-number)

2 Task 2. Data Dictionary

2.1 Discussion of sstorage structures for tables

Table 1: customers

Name	Type	Size in Bytes	Constraints
name	NVARCHAR (128)	256	`PK_customer`
			`PRIMARY KEY (name)`
address	NVARCHAR (128)	256	`none`
category	TINYINT	1	`NOT NULL`
			`CHECK category <= 10 AND category>=1`

Table 2: assemblies

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL
			`PK_assembly PRIMARY KEY (id)`
customer_name	NVARCHAR (128)	256	`FK_customer_assembly`
			`FOREIGN KEY (customer_name)`
			`REFERENES`
			`customers (name)`
dateordered	DATETIME	8	`NOT NULL`
assemblydetails	NVARCHAR (128)	256	`none`

Table 3: departments

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL
			`PK_department PRIMARY KEY (id)`
department_data	NVARCHAR (128)	256	`none`

Table 4: cut_processes

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_cut_processes PRIMARY KEY (id)
department	INT	4	FK_cut_processes_departments FOREIGN KEY (department) REFERENES departments (id)
cuttingtype	NVARCHAR (128)	256	none
machinetype	NVARCHAR (128)	256	none

Table 5: paint_processes

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_paint_processes PRIMARY KEY (id)
department	INT	4	FK_paint_processes_departments FOREIGN KEY (department) REFERENES departments (id)
painttype	NVARCHAR (128)	256	none
paintingmethod	NVARCHAR (128)	256	none

Table 6: fit_processes

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_fit_processes PRIMARY KEY (id)
department	INT	4	FK_fit_processes_departments FOREIGN KEY (department) REFERENES departments (id)
fittype	NVARCHAR (128)	256	none

Table 7: cut_jobs

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_cut_jobs PRIMARY KEY (id)
assemblyid	INT	4	NOT NULL
processid	INT	4	NOT NULL
startdate	DATETIME	8	NOT NULL
enddate	DATETIME	8	none
cuttingtime	INT	4	none
labortime	INT	4	none
machinetype	NVARCHAR (128)	256	none
materialused	NVARCHAR (128)	256	none

Table 8: paint_jobs

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_paint_jobs PRIMARY KEY (id)
assemblyid	INT	4	NOT NULL
processid	INT	4	NOT NULL
startdate	DATETIME	8	NOT NULL
enddate	DATETIME	8	none
labortime	INT	4	none
paintvolume	NVARCHAR (128)	256	none
paintcolor	NVARCHAR (128)	256	none

Table 9: fit_jobs

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_fit_jobs PRIMARY KEY (id)
assemblyid	INT	4	NOT NULL
processid	INT	4	NOT NULL
startdate	DATETIME	8	NOT NULL
enddate	DATETIME	8	none
labortime	INT	4	none
fittype	NVARCHAR (128)	256	none

Table 10: assembly_accounts

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_assembly_account_number PRIMARY KEY (id)
assemblyid	INT	4	NOT NULL FK_assembly_account_assembly FOREIGN KEY (assemblyid) REFERENES assemblies (id)
dateestablished	DATETIME	8	NOT NULL

Table 11: department_accounts

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_department_account_number PRIMARY KEY (id)
departmentid	INT	4	NOT NULL FK_department_account_department FOREIGN KEY (assemblyid) REFERENES departments (id)
dateestablished	DATETIME	8	NOT NULL

Table 12: process_accounts

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_process_account_number PRIMARY KEY (id)
processid	INT	4	NOT NULL FK_process_account_process FOREIGN KEY (assemblyid) REFERENES processes (id)
dateestablished	DATETIME	8	NOT NULL

Table 13: transactions

Name	Type	Size in Bytes	Constraints
id	INT	4	NOT NULL PK_transactions PRIMARY KEY (id)
supcost	MONEY	8	none
assemblyaccountid	INT	4 FK_assembly_account_id	NOT NULL FOREIGN KEY (assemblyaccountid) REFERENES assembly_accounts (id)
processaccountid	INT	4	NOT NULL FK_process_account_id FOREIGN KEY (processaccountid) REFERENES process_accounts (id)
departmentaccountid	INT	4	NOT NULL FK_department_account_id FOREIGN KEY (departmentaccountid) REFERENES department_accounts (id)

3 Task 3.

3.1 Discussion of storage structures for table

Table Name	Query No. and Type	Search Key	Query Frequency	Selected File Organization	Justifications
customers	1, insert		30/day	b+ tree indexed	because of frequent
	3, random search	name	40/day	on category	ranged search and
	13, range search	category	100/day		deletion
assemblies	3, insert		40/day	Extendable hash	because of
	5, random search	id	10/day	table on key	frequency of
	6, random search	id	50/day	id	random search
	9, random search	id	200/day		on the column
	11, random search	id	100/day		id
	12, random search	id	20/day		
departments	2, insertion		infrequent	extendable hashing	because of
	4, random search	id	infrequent	on column id	frequency of
	5, random search	id	50/day		random search
	10, random search	id	20/day		on the column
	11, random search	id	100/day		id
	12, random search	id	20/day		
cut_processes	4, insertion		infrequent	extendable hashing	because of
	5, random search	id	10/day	on column id	frequency of
	6, random search	id	50/day		random search
	11, random search	id	100/day		on the column id
paint_processes	4, insertion		infrequent	extendable hashing	because of
	5, random search	id	10/day	on column id	frequency of
	6, random search	id	50/day		random search
	11, random search	id	100/dayy		on the column id
fit_processes	4, insertion		infrequent	extendable hashing	because of
	5, random search	id	10/day	on column id	frequency of
	6, random search	id	50/day		random search
	11, random search	id	100/day		on the column id
cut_jobs	6, insertion		50/day		
	7, update (random search)	id	50/day	b-tree index	because of updates,
	10, random search	id	20/day	on column id	deletion and frequency of
	12, random search	enddate	20/day		random search
	14, deletion (range search)	id	1/month		on the column id
paint_jobs	6, insertion		50/day	extendable hashing	because of
	7, update (random search)	id	50/day	on column id	frequency of
	10, random search	id	20/day		random search
	12, random search	enddate	20/day		on the column id
	15, update (random search)	id	20/day		
fit_jobs	6, insertion		50/day	extendable hashing	because of
	7, update (random search)	id	50/day	on column id	frequency of
	10, random search	id	20/day		random search
	12, random search	enddate	20/day		on the column id
assembly_accounts	5, insertion		10/day	extendable hashing	because of the frequency
	8, random search	id	50/day	on the column id	of random search
	9, random search	id	200/day		on the column id
department_accounts	5, insertion		10/day	extendable hashing	because of the frequency
	8, random search	id	50/day	on the column id	of random search on that column
process_accounts	5, insertion		10/day	extendable hashing	because of the frequency
	8, random search	id	50/day	on the column id	of random search on that column
transactions	8, insert		50/day	10/day	extendable hashing
because of the frequency					
	9, random search	assemblyaccountid	200/day	on the column id	of random search on that column

3.2 Discussion of storage structures for tables (Azure SQL database)

Extendable hashing is not enabled on microsoft sql except for memory optimized tables.
So all hash tables will become b-trees.

4 Task 4. SQL statements and screenshots showing the creation of tables in Azure SQL database

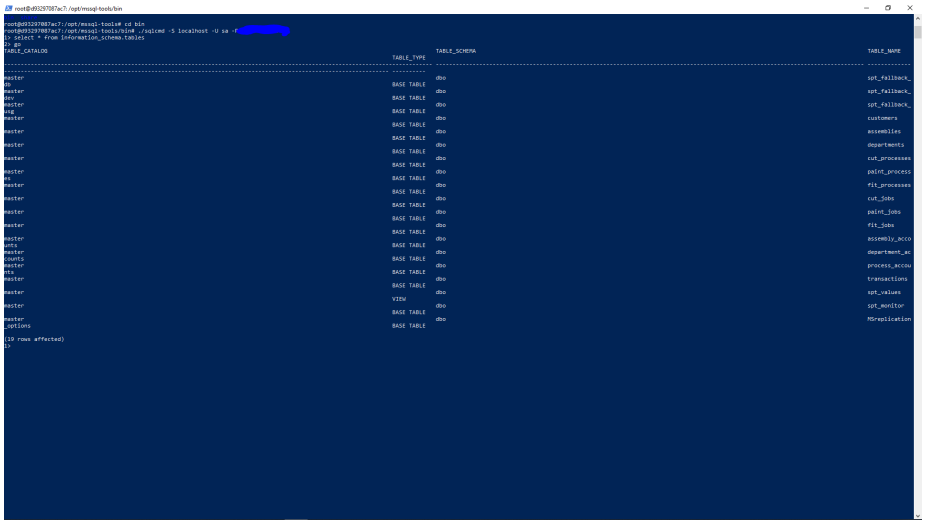
```
CREATE TABLE customers (
name NVARCHAR(128) NOT NULL,
address NVARCHAR(128),
category TINYINT NOT NULL
CHECK(category<=10 AND category >= 1),
CONSTRAINT PK_customer
PRIMARY KEY(name)
);
go
CREATE TABLE assemblies (
id INT NOT NULL IDENTITY,
customer_name NVARCHAR(128),
dateordered DATETIME NOT NULL,
assemblydetails NVARCHAR(128),
CONSTRAINT FK_customer_assembly
FOREIGN KEY (customer_name)
REFERENCES customers(name),
CONSTRAINT PK_assembly
PRIMARY KEY(id)
);
go
CREATE TABLE departments (
id INT NOT NULL IDENTITY,
departmentdata NVARCHAR(128),
CONSTRAINT PK_department
PRIMARY KEY(id)
);
CREATE TABLE cut_processes (
id INT NOT NULL ,
department INT,
cuttingtype NVARCHAR(128),
machinetype NVARCHAR(128),
CONSTRAINT PK_cut_processes
PRIMARY KEY (id),
CONSTRAINT FK_cut_processes_departments
FOREIGN KEY (department)
REFERENCES departments(id)
);
go
CREATE TABLE paint_processes (
id INT NOT NULL,
department INT,
```

```
painttype NVARCHAR(128),
paintingmethod NVARCHAR(128),
CONSTRAINT PK_paint_processes
PRIMARY KEY (id),
CONSTRAINT FK_paint_processes_departments
FOREIGN KEY (department)
REFERENCES departments(id)
);
go
CREATE TABLE fit_processes (
id INT NOT NULL,
department INT,
fittype NVARCHAR(128),
CONSTRAINT PK_fit_processes
PRIMARY KEY (id),
CONSTRAINT FK_fit_processes_departments
FOREIGN KEY (department)
REFERENCES departments(id)
);
go
CREATE TABLE cut_jobs (
id INT NOT NULL ,
assemblyid INT NOT NULL,
processid INT NOT NULL,
startdate DATETIME NOT NULL,
enddate DATETIME,
cuttingtime int,
machinetype NVARCHAR(128),
materialused NVARCHAR(128),
labortime int,
CONSTRAINT PK_cut_jobs
PRIMARY KEY (id),
);
go
CREATE TABLE paint_jobs (
id INT NOT NULL,
assemblyid INT NOT NULL,
processid INT NOT NULL,
startdate DATETIME NOT NULL,
enddate DATETIME,
paintcolor NVARCHAR(128),
paintvolume NVARCHAR(128),
labortime int,
CONSTRAINT PK_paint_jobs
PRIMARY KEY (id),
);
```

```
go
CREATE TABLE fit_jobs (
id INT NOT NULL,
assemblyid INT NOT NULL,
processid INT NOT NULL,
startdate DATETIME NOT NULL,
enddate DATETIME,
fittype NVARCHAR(128),
labortime int,
CONSTRAINT PK_fit_jobs
PRIMARY KEY (id),
);
go
CREATE TABLE assembly_accounts(
id INT NOT NULL,
dateestablished DATETIME NOT NULL,
assemblyid INT NOT NULL,
CONSTRAINT PK_assembly_account_number
PRIMARY KEY (id),
CONSTRAINT FK_assembly_account_assembly
FOREIGN KEY (assemblyid)
REFERENCES assemblies(id)
);
go
CREATE TABLE department_accounts(
id INT NOT NULL,
dateestablished DATETIME NOT NULL,
departmentid INT NOT NULL,
CONSTRAINT PK_department_account_number
PRIMARY KEY (id),
CONSTRAINT FK_department_account_department
FOREIGN KEY (departmentid)
REFERENCES departments(id)
);
go
CREATE TABLE process_accounts(
id INT NOT NULL,
dateestablished DATETIME NOT NULL,
processid INT NOT NULL,
CONSTRAINT PK_process_account_number
PRIMARY KEY (id)
);
go
CREATE TABLE transactions (
id INT NOT NULL,
supcost MONEY,
```

```
assemblyaccountid int,
processaccountid int,
departmentaccountid int
CONSTRAINT PK_transactions
PRIMARY KEY (id),
CONSTRAINT FK_assembly_account_id
FOREIGN KEY (assemblyaccountid)
REFERENCES assembly_accounts(id),
CONSTRAINT FK_process_account_id
FOREIGN KEY (processaccountid)
REFERENCES process_accounts(id),
CONSTRAINT FK_department_account_id
FOREIGN KEY (departmentaccountid)
REFERENCES department_accounts(id)
);
go
CREATE TRIGGER TR_fit_process_insert
ON fit_processes after insert
AS
if exists (
select id from inserted where id IN (
SELECT id FROM paint_processes
UNION SELECT id FROM cut_processes
) ) BEGIN ROLLBACK TRANSACTION END
go
CREATE TRIGGER TR_paint_process_insert
ON paint_processes after insert
AS
if exists (
select id from inserted where id IN (
SELECT id FROM fit_processes
UNION SELECT id FROM cut_processes
) ) BEGIN ROLLBACK TRANSACTION END
go
CREATE TRIGGER TR_cut_process_insert
ON cut_processes after insert
AS
if exists (
select id from inserted where id IN (
SELECT id FROM paint_processes
UNION SELECT id FROM fit_processes
) ) BEGIN ROLLBACK TRANSACTION END
go
CREATE TRIGGER TR_process_account_id_rel
ON process_accounts
AFTER INSERT
```

```
as
if exists(
select processid from inserted
WHERE PROCESSID NOT IN(
SELECT id FROM cut_processes
UNION SELECT id FROM paint_processes
UNION SELECT id FROM fit_processes
)
)BEGIN ROLLBACK TRANSACTION END;
```



5 Task 5. The Java source program and screenshots showing its successful compilation

```
//Zane Gray
//November 14 2019
//individual project question 5
//cs 4513 001 fall 2019
import java.io.*;
import java.text.*;
import java.lang.String.*;
import java.util.*;
import java.sql.*;
import java.util.*;
import microsoft.sql.*;
import java.sql.*;

//boilerplate class shamelessly taken from the example files.
public class q5 {
	static Scanner in;
	static String hostName;
	static String dbName;
	static String user;
	static String password;
	static String url;
	static Connection connection;
	static String schema;
	static Statement statement;

	public static void main(final String[] args) throws
	    ↪ SQLException {
		in = new Scanner(System.in);
		try {
			Class.forName("com.microsoft.sqlserver.jdbc.
			    ↪ SQLServerDriver");
		} catch (final ClassNotFoundException e) {
			System.out.println("Sql␣Server␣driver␣not␣
			    ↪ found");
			e.printStackTrace();
			System.out.println(e);
			return;
		}
		// Connect to database
		hostName = "127.0.0.1";
		dbName = "master";
```

```
user = "sa";
password = "Jbermine41611";
url = "jdbc:sqlserver://localhost:1433;user=sa;
    ↪ password=Jbermine41611";
connection = DriverManager.getConnection(url);
schema = connection.getSchema();
statement = connection.createStatement();
// main menu
do {
        switch (iterateMenu()) {
        case 1:
                choiceOne();
                break;// new customer
        case 2:
                choiceTwo();
                break;// new dept
        case 3:
                choiceThree();
                break;// new assembly
        case 4:
                choiceFour();
                break;// new process
        case 5:
                choiceFive();
                break;// new account
        case 6:
                choiceSix();
                break;// new job
        case 7:
                choiceSeven();
                break;// update job
        case 8:
                choiceEight();
                break;// new transaction
        case 9:
                choiceNine();
                break;// get cost of assembly
        case 10:
                choiceTen();
                break;// get labor time by
                    ↪ department and date
        case 11:
                choiceEleven();
                break;// get complete processes by
                    ↪ assembly
        case 12:
```

```
                        choiceTwelve();
                        break;// get complete jobs by date
                            ↪ and department
                case 13:
                        choiceThirteen();
                        break;// get customers by category
                case 14:
                        choiceFourteen();
                        break;// delete cut-jobs by range
                case 15:
                        choiceFifteen();
                        break;// change paint-job number
                case 16:
                        choiceSixteen();
                        break;// load customers from file
                case 17:
                        choiceSeventeen();
                        break;// save customers to file
                case 18:
                        return;
                default:
                        System.out.println("Invalid␣choice")
                            ↪ ;
                }
        } while (true);
}

// Present the user with the main menu and get their
    ↪ selection
public static int iterateMenu() {
        System.out.println("WELCOME␣TO␣THE␣JOB-SHOP␣
            ↪ ACCOUNTING␣DATABASE␣SYSTEM");
        System.out.println("(1)␣Enter␣a␣new␣customer");
        System.out.println("(2)␣Enter␣a␣new␣department");
        System.out.println(
                        "(3)␣Enter␣a␣new␣assembly␣with␣its␣
                            ↪ customer-name,assembly-details
                            ↪ ,␣assembly-id␣and␣date-ordered
                            ↪ ");
        System.out.println(
                        "(4)␣Enter␣a␣new␣process-id␣and␣its␣
                            ↪ department␣together␣with␣its␣
                            ↪ type␣and␣information␣relevant␣
                            ↪ to␣the␣type");
        System.out.println(
                        "(5)␣Create␣a␣new␣account␣and␣
```

```
                    ↪ associate␣it␣with␣the␣process,
                    ↪ ␣assembly,␣or␣department␣to␣
                    ↪ which␣it␣is␣applicable");
System.out
                .println("(6)␣Enter␣a␣new␣job,␣given
                    ↪ ␣its␣job-no,␣assembly-id,␣
                    ↪ process-id,␣and␣date␣the␣job␣
                    ↪ commenced");
System.out.println(
                "(7)␣At␣the␣completion␣of␣a␣job,␣
                    ↪ enter␣the␣date␣it␣completed␣
                    ↪ and␣the␣information␣relevant␣
                    ↪ to␣the␣type␣of␣job");
System.out.println(
                "(8)␣Enter␣a␣transaction-no␣and␣its␣
                    ↪ sup-cost␣and␣update␣all␣the␣
                    ↪ costs␣(details)␣of␣the␣
                    ↪ affected␣accounts␣by␣adding␣
                    ↪ sup-cost␣to␣their␣current␣
                    ↪ values␣of␣details");
System.out.println("(9)␣Retrieve␣the␣cost␣incurred␣
    ↪ on␣an␣assembly-id");
System.out.println(
                "(10)␣Retrieve␣the␣total␣labor␣time␣
                    ↪ within␣a␣department␣for␣jobs␣
                    ↪ completed␣in␣the␣department␣
                    ↪ during␣a␣given␣date");
System.out.println(
                "(11)␣Retrieve␣the␣processes␣through
                    ↪ ␣which␣a␣given␣assembly-id␣has
                    ↪ ␣passed␣so␣far␣(in␣
                    ↪ datecommenced␣order)␣and␣the␣
                    ↪ department␣responsible␣for␣
                    ↪ each␣process");
System.out.println(
                "(12)␣Retrieve␣the␣jobs␣(together␣
                    ↪ with␣their␣type␣information␣
                    ↪ and␣assembly-id)␣completed␣
                    ↪ during␣a␣given␣date␣in␣a␣given
                    ↪ ␣department");
System.out.println("(13)␣Retrieve␣the␣customers␣(in
    ↪ ␣name␣order)␣whose␣category␣is␣in␣a␣given␣
    ↪ range");
System.out.println("(14)␣Delete␣all␣cut-jobs␣whose␣
    ↪ job-no␣is␣in␣a␣given␣range");
System.out.println("(15)␣Change␣the␣color␣of␣a␣
```

```
            ↪ given␣paint␣job");
        System.out.println("(16)␣Import:␣enter␣new␣
            ↪ customers␣from␣a␣data␣file␣until␣the␣file␣is
            ↪ ␣empty");
        System.out.println(
                    "(17)␣Export:␣Retrieve␣the␣customers
                        ↪ ␣(in␣name␣order)␣whose␣
                        ↪ category␣is␣in␣a␣given␣range␣
                        ↪ and␣output␣them␣to␣a␣data␣file
                        ↪ ␣instead␣of␣screen");
        System.out.println("(18)␣Quit");
        int i = -1; // error in case of invalid input
        try {
                i = in.nextInt();
                in.nextLine();
        } catch (final Exception e) {
                System.out.println(e);
        }
        return (i);
}

// Enter a new customer given name, address and category
static void choiceOne() {
        System.out.print("Enter␣customer␣name:\t");
        final String name = in.nextLine().replaceAll("'", "
            ↪ '");// escape quotes
        System.out.print("\nEnter␣customer␣address:\t");
        final String address = in.nextLine().replaceAll("'"
            ↪ , "'");
        System.out.print("\nEnter␣customer␣category:\t");
        final String category = in.nextLine().replaceAll("'
            ↪ ", "'");
        System.out.println("");
        final String query = "INSERT␣INTO␣customers␣VALUES␣
            ↪ ('" + name + "','" + address + "'," +
            ↪ category + ");";
        try {
                statement.execute(query);

        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
```

```
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// Enter a new department given data
static void choiceTwo() {
        System.out.print("Enter␣department␣data:\t");
        final String data = in.nextLine().replaceAll("'", "
            ↪ ''");// escape quotes;
        System.out.println("");
        final String query = "INSERT␣INTO␣departments␣
            ↪ VALUES␣('" + data + "');";
        try {
                statement.execute(query);

        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// Enter a new assembly given customer name, date
// ordered and assembly details
static void choiceThree() {
        System.out.print("Enter␣customer␣name␣for␣this␣
            ↪ assembly:\t");
        final String name = in.nextLine().replaceAll("'", "
            ↪ ''");// escape quotes;
        System.out.println("\nEnter␣assembly␣date␣and␣time␣
            ↪ ordered");
        System.out.print(
                        "yyyymmdd␣hh:mm:ss␣{am|pm};␣" + "eg␣
                            ↪ 20100202␣10:10:10␣am␣for␣
                            ↪ february␣2nd␣2010␣at␣10:10:10␣
                            ↪ am):\t");
        final String date = in.nextLine().replaceAll("'", "
            ↪ ''");// escape quotes;
        System.out.print("\nEnter␣assembly␣details␣:\t");
        final String details = in.nextLine().replaceAll("'"
```

```
            ↪ , "''");// escape quotes;
        System.out.println("");
        final String query = "INSERT␣INTO␣assemblies␣VALUES
            ↪ ␣('" + name + "','" + date + "','" + details
            ↪  + "');";
        try {
                statement.execute(query);

        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// Enter a new process-id and its department together with
// its type and information relevant to the type
static void choiceFour() {
        // data for any type of process
        System.out.print("Enter␣a␣new␣process␣id:\t");
        final String id = in.nextLine().replaceAll("'", "''
            ↪ ");// escape quotes;
        System.out.print("\nEnter␣the␣department␣id␣of␣the␣
            ↪ new␣process");
        final String dept = in.nextLine().replaceAll("'", "
            ↪ ''");// escape quotes;
        final String values = "␣VALUES␣(" + id + "," + dept
            ↪  + ",'";// will be appended to a
            ↪ stringbuilder
        // once the table name is known
        System.out.println("\nEnter␣the␣process␣type");
        System.out.print("(must␣be␣one␣of␣'Cut',␣'Fit'␣or␣'
            ↪ Paint'):\t");
        final String proctype = in.nextLine();
        // get the appropriate table for the process type
        final StringBuilder query = new StringBuilder("
            ↪ INSERT␣INTO␣");
        switch (proctype.toLowerCase().charAt(0)) {
        // all info relevant to a fit process
        case 'f':
                query.append("␣fit_processes␣");
```

```
        query.append(values);
        System.out.print("\nEnter␣the␣fit␣type");
        query.append(in.nextLine().replaceAll("'", "
            ↪ ''"));// escape quotes
        break;
// all info relevant to a cut process
case 'c':
        query.append("␣cut_processes␣");
        query.append(values);
        System.out.print("\nEnter␣the␣cutting␣type")
            ↪ ;
        query.append(in.nextLine().replaceAll("'", "
            ↪ ''") + "','");
        System.out.print("\nEnter␣the␣machine␣type")
            ↪ ;
        query.append(in.nextLine().replaceAll("'", "
            ↪ ''"));
        break;
// all info relevant to a paint process
case 'p':
        query.append("␣paint_processes␣");
        query.append(values);
        System.out.print("\nEnter␣the␣paint␣type");
        query.append(in.nextLine().replaceAll("'", "
            ↪ ''") + "','");
        System.out.print("\nEnter␣the␣painting␣
            ↪ method");
        query.append(in.nextLine().replaceAll("'", "
            ↪ ''"));
        break;
default:
        System.out.println(proctype + "␣is␣not␣a␣
            ↪ valid␣process␣type");
        return;
}
query.append("');");
System.out.println("");
try {
        statement.execute(query.toString());

} catch (final SQLException e) {
        System.out.println("A␣SQL␣Error␣has␣occoured
            ↪ :");
        System.out.println(e);
        System.out.print("(state:");
        System.out.print(e.getSQLState() + ";");
```

```
                System.out.print("error code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// Create a new account and associate it with the process,
    ↪  assembly, or
// department to which it is applicable
static void choiceFive() {
        System.out.println("Enter the account type");
        System.out.print("Must be one of Process, Assembly
            ↪ or Department:\t");
        final String accountType = in.nextLine().replaceAll
            ↪ ("'", "''");// escape quotes
        System.out.print("\nEnter the accounts unique
            ↪ number:\t");
        final String accountNumber = in.nextLine().
            ↪ replaceAll("'", "''");
        System.out.print("\nEnter the Id associated with
            ↪ the" + accountType);
        final String acctId = in.nextLine().replaceAll("'",
            ↪  "''");
        // choose the relevant table
        String query;
        switch (accountType.toLowerCase().charAt(0)) {
        case 'p':
                query = ("INSERT INTO process_accounts
                    ↪ VALUES(" + accountNumber + ",
                    ↪ CURRENT_TIMESTAMP," + acctId + ");");
                break;
        case 'a':
                query = ("INSERT INTO assembly_accounts
                    ↪ VALUES(" + accountNumber + ",
                    ↪ CURRENT_TIMESTAMP," + acctId + ");");
                break;
        case 'd':
                query = ("INSERT INTO department_accounts
                    ↪ VALUES(" + accountNumber + ",
                    ↪ CURRENT_TIMESTAMP," + acctId + ");");
                break;
        default:
                System.out.println(accountType + " is not a
                    ↪ valid account type");
                return;
        }
```

```
        System.out.println("");
        try {
                statement.execute(query.toString());

        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }

}

// Enter a new job, given its job-no, assembly-id, process
    ↪ -id, and date the job
// commenced
static void choiceSix() {
        System.out.println("\nEnter␣the␣job␣type");
        System.out.print("(must␣be␣one␣of␣'Cut',␣'Fit'␣or␣'
            ↪ Paint'):\t");
        final String proctype = in.nextLine();
        // get the appropriate table for the job type
        final StringBuilder query = new StringBuilder("
            ↪ INSERT␣INTO␣");
        switch (proctype.toLowerCase().charAt(0)) {

        case 'f':
                query.append("␣fit_jobs␣(id,␣assemblyid,␣
                    ↪ processid,␣startdate)␣VALUES␣");
                break;

        case 'c':
                query.append("␣cut_jobs␣(id,␣assemblyid,␣
                    ↪ processid,␣startdate)␣VALUES␣");
                break;

        case 'p':
                query.append("␣paint_jobs␣(id,␣assemblyid,␣
                    ↪ processid,␣startdate)␣VALUES␣");
                break;
        default:
                System.out.println(proctype + "␣is␣not␣a␣
```

```
                ↪ valid␣job␣type");
            return;
        }
        System.out.print("\nEnter␣a␣unique␣job␣Id:\t");
        query.append("␣(" + in.nextLine().replaceAll("'", "
            ↪ ''"));
        System.out.print("\nEnter␣an␣assembly␣Id:\t");
        query.append("," + in.nextLine().replaceAll("'", "
            ↪ ''"));
        System.out.print("\nEnter␣a␣process␣Id:\t");
        query.append("," + in.nextLine().replaceAll("'", "
            ↪ ''") + ",CURRENT_TIMESTAMP)");

        System.out.println("");
        try {
            statement.execute(query.toString());
        } catch (final SQLException e) {
            System.out.println("A␣SQL␣Error␣has␣occoured
                ↪ :");
            System.out.println(e);
            System.out.print("(state:");
            System.out.print(e.getSQLState() + ";");
            System.out.print("error␣code:");
            System.out.print(Integer.toString(e.
                ↪ getErrorCode()) + ")");
        }

}

// At the completion of a job, enter the date it completed
    ↪  and the
// information relevant to the type of job
static void choiceSeven() {

        System.out.print("Enter␣the␣job␣Id");
        // make sure that the job exists and find the
            ↪ appropriate table
        final String id = in.nextLine().replaceAll("'", "''
            ↪ ");
        try {

            final ResultSet rs = statement.executeQuery(
                ↪ "␣SELECT␣jobtype␣FROM␣" + "␣(␣select␣
                ↪ 'cut_jobs'␣AS␣jobtype␣"
                        + "␣FROM␣cut_jobs␣" + "␣where
                            ↪ ␣id␣=␣" + id + "␣UNION
```

```
                                        ↪ ␣" + "␣SELECT␣'
                                        ↪ paint_jobs'␣" + "␣AS␣
                                        ↪ jobtype␣"
                                    + "␣FROM␣paint_jobs␣" + "␣
                                        ↪ where␣id␣=␣" + id + "␣
                                        ↪ UNION␣select␣'fit_jobs
                                        ↪ '␣" + "␣AS␣jobtype␣"
                                    + "␣FROM␣fit_jobs␣␣" + "␣
                                        ↪ where␣id␣=␣" + id +")␣
                                        ↪ AS␣jobtypes");
                        int count = 0;
String table = "error";
                        while (rs.next()) {
                                table = rs.getString(1);
                                ++count; // should be exactly 1
                        }
                        if (count != 1) {

                                System.out.println("Invalid␣account␣
                                    ↪ id");
                                return;
                        }
                        System.out.print("\nEnter␣the␣end␣date:\t");
                        final String enddate = in.nextLine().
                            ↪ replaceAll("'", "''");// escape
                            ↪ quotes
                        System.out.print("\nEnter␣the␣labor␣time:\t"
                            ↪ );
                        final String labortime = in.nextLine().
                            ↪ replaceAll("'", "''");
                        final StringBuilder query = new
                            ↪ StringBuilder("UPDATE␣" + table + "␣"
                            ↪  + "SET␣enddate␣=␣CAST('" + enddate
                                        + "'␣AS␣DATETIME)," + "
                                            ↪ labortime␣=␣" +
                                            ↪ labortime);
                        switch (table.charAt(0)) {
                        case 'c':
                                System.out.print("\nEnter␣the␣
                                    ↪ material␣used:\t");
                                query.append(",materialused␣=␣'" +
                                    ↪ in.nextLine().replaceAll("'",
                                    ↪ "''") + "';");
                                break;
                        case 'p':
                                System.out.print("\nEnter␣the␣color␣
```

```
                        ↪ of␣paint:\t");
                    query.append(",paintcolor='" + in.
                        ↪ nextLine().replaceAll("'", "''
                        ↪ "));
                    System.out.print("\nEnter␣the␣volume
                        ↪ ␣of␣paint");
                    query.append("',paintvolume='" + in.
                        ↪ nextLine().replaceAll("'", "''
                        ↪ ") + "';");
                    break;
                default:
                    query.append(";");
                }

                statement.execute(query.toString());
            } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
            }

    }

    /*
     * Enter a transaction-no and its sup-cost and update all
         ↪ the costs (details) of
     * the affected accounts by adding sup-cost to their
         ↪ current values of details
     */
    static void choiceEight() {
            System.out.print("\nEnter␣a␣unique␣transaction-no:\
                ↪ t");
            final String transNo = in.nextLine();
            System.out.print("\nEnter␣the␣sup-cost␣of␣the␣
                ↪ transaction:\t");
            final String supCost = in.nextLine();
            System.out.print("\nEnter␣the␣process␣account␣
                ↪ number:\t");
            final String procAcct = in.nextLine();
            System.out.print("\nEnter␣the␣assembly␣account␣
                ↪ number:\t");
```

```
        final String assNo = in.nextLine();
        System.out.print("\nEnter␣the␣department␣account␣
            ↪ number:\t");
        final String depNo = in.nextLine();
        final String query = "INSERT␣INTO␣transactions␣
            ↪ VALUES␣(" + transNo + "," + supCost + "," +
            ↪ assNo + ","
                    + procAcct + "," + depNo + ");";

        System.out.println("");
        try {
                statement.execute(query.toString());
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// Retrieve the cost incurred on an assembly-id
static void choiceNine() {
        System.out.print("\nEnter␣an␣assembly␣number:\t");
        final String query = "SELECT␣SUM(supcost)␣" + "FROM
            ↪ ␣transactions␣t␣" + "INNER␣JOIN␣
            ↪ assembly_accounts␣aa␣"
                    + "ON␣t.assemblyaccountid␣=␣aa.id␣"
                        ↪ + "INNER␣JOIN␣assemblies␣a␣" +
                        ↪  "ON␣aa.assemblyid=a.id␣"
                    + "WHERE␣a.id␣=␣" + in.nextLine().
                        ↪ replaceAll("'", "''");//
                        ↪ escape quotes

        System.out.println("");
        try {
                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                rs.next();
                System.out.println("Total␣cost:\t␣" + rs.
                    ↪ getString(1));
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
```

```
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
                return;
        }
}

/*
 * Retrieve the total labor time within a department for
     ↪ jobs completed in the
 * department during a given date
 */
static void choiceTen() {
        System.out.print("\nEnter␣the␣department␣id:\t");
        final String depId = in.nextLine();
        System.out.print("\nEnter␣the␣date␣completed");
        System.out.print("yyyymmdd␣;␣" + "eg␣20100202␣for␣
            ↪ february␣2nd␣2010␣):\t");
        final String date = in.nextLine();
        final String query = "␣␣␣␣␣␣␣␣␣␣SELECT␣SUM(t)␣" + "
            ↪ FROM␣(SELECT␣j.labortime␣t,␣p.department␣d,␣
            ↪ j.enddate␣e␣"
                        + "␣FROM␣cut_jobs␣j␣INNER␣JOIN␣
                            ↪ cut_processes␣p␣ON␣j.processid
                            ↪ =p.id␣"
                        + "␣UNION␣SELECT␣j.labortime␣t,p.
                            ↪ department␣d,␣j.enddate␣e␣"
                        + "␣FROM␣paint_jobs␣j␣INNER␣JOIN␣
                            ↪ paint_processes␣p␣ON␣j.
                            ↪ processid=p.id␣"
                        + "␣UNION␣SELECT␣j.labortime␣t,␣p.
                            ↪ department␣d,␣␣j.enddate␣e␣"
                        + "␣FROM␣fit_jobs␣j␣INNER␣JOIN␣
                            ↪ fit_processes␣p␣ON␣j.processid
                            ↪ =p.id)␣AS␣a" + "␣WHERE(␣a.d="
                            ↪ + depId
                        + "␣AND␣CAST(␣'"+date+"'␣AS␣DATE)␣=␣
                            ↪ CAST␣(a.e␣AS␣DATE));";
        try {
                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                rs.next();
```

```
                System.out.println("Total␣labor␣time␣:" + rs
                    ↪ .getString(1));
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
                return;
        }
}

/*
 * Retrieve the processes through which a given assembly-
     ↪ id has passed so far
 * (in datecommenced order) and the department responsible
     ↪  for each process
 */
static void choiceEleven() {
        System.out.print("\nEnter␣the␣assembly-id\t");
        final String query = "␣WITH␣jobs␣AS␣(␣" + "␣␣SELECT
            ↪ ␣id,␣assemblyid,␣processid,␣startdate,␣
            ↪ enddate␣␣"
                        + "␣FROM␣cut_jobs␣UNION␣" + "␣
                            ↪ ␣␣␣␣␣␣␣SELECT␣id,␣assemblyid,␣
                            ↪ processid,␣startdate,␣enddate␣
                            ↪ ␣"
                        + "␣FROM␣paint_jobs␣UNION␣" + "␣
                            ↪ ␣␣␣␣␣SELECT␣id,␣assemblyid,␣
                            ↪ processid,␣startdate,␣enddate␣
                            ↪ ␣"
                        + "␣FROM␣fit_jobs␣␣" + "␣),␣
                            ↪ processes␣AS␣(␣" + "␣␣␣SELECT␣
                            ↪ id,␣department␣FROM␣
                            ↪ cut_processes␣UNION␣"
                        + "␣␣␣␣SELECT␣id,␣department␣FROM␣
                            ↪ paint_processes␣UNION␣␣" + "␣
                            ↪ ␣␣␣␣␣␣SELECT␣id,␣department␣
                            ↪ FROM␣fit_processes␣"
                        + "␣)␣SELECT␣jobs.processid,␣
                            ↪ processes.department␣"
                        + "␣FROM␣jobs␣INNER␣JOIN␣processes␣
                            ↪ ON␣jobs.processid␣=␣processes.
```

```
                        ↪ id␣"
                    + "␣WHERE␣jobs.enddate␣NOT␣NULL␣AND␣
                        ↪ jobs.assemblyid␣=␣" +in.
                        ↪ nextLine()+ "␣␣␣ORDER␣BY␣jobs.
                        ↪ startdate␣" ;
        try {

                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                while (rs.next()) {
                        System.out.printf("PROCESSId:\t%s|␣
                            ↪ DEPARTMENT:\t%s", rs.getString
                            ↪ (1), rs.getString(2));
                }
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
                return;
        }
}

/*
 * Retrieve the jobs (together with their type information
     ↪  and assembly-id)
 * completed during a given date in a given department
 */
static void choiceTwelve() {
        System.out.print("\nEnter␣the␣department␣id:\t");
        final String depId = in.nextLine();
        System.out.print("\nEnter␣the␣date␣completed");
        System.out.print("yyyymmdd␣;␣" + "eg␣20100202␣for␣
            ↪ february␣2nd␣2010␣):\t");
        final String date = in.nextLine();
        final String query = "␣SELECT␣jobtype,␣assemblyid,␣
            ↪ processid␣FROM␣(␣" + "␣␣␣␣select␣'cutjob'␣␣"
                    + "␣AS␣jobtype,␣assemblyid,␣
                        ↪ processid,␣enddate␣␣" + "␣FROM
                        ↪ ␣cut_jobs␣UNION␣" + "␣␣␣␣
                        ↪ select␣'paintjob'␣␣"
                    + "␣AS␣jobtype,␣assemblyid,␣
```

```
                    ↪ processid␣,␣enddate␣␣" + "␣
                    ↪ FROM␣paint_jobs␣UNION␣" + "␣
                    ↪ ␣␣␣␣␣␣␣␣select␣'fitjob'␣"
                + "␣␣AS␣jobtype,␣assemblyid,␣
                    ↪ processid␣,␣enddate␣␣" + "␣
                    ↪ FROM␣fit_jobs␣␣"
                + "␣)␣as␣a␣WHERE␣CAST(enddate␣AS␣
                    ↪ DATE)␣=␣CAST(␣'" + date + "'␣␣
                    ↪ AS␣DATE)␣␣" + "␣AND␣processid␣
                    ↪ IN␣(␣"
                + "␣␣␣␣SELECT␣id␣FROM␣cut_processes␣
                    ↪ WHERE␣department=␣" + depId +
                    ↪ "␣␣UNION␣"
                + "␣␣␣␣SELECT␣id␣FROM␣
                    ↪ paint_processes␣WHERE␣
                    ↪ department=␣" + depId + "␣␣
                    ↪ UNION␣"
                + "␣␣␣␣SELECT␣id␣FROM␣fit_processes␣
                    ↪ WHERE␣department=␣" + depId +
                    ↪ "␣␣);␣" ;
        try {

                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                while (rs.next()) {
                        System.out.printf("Job␣type:\t%s|␣id
                            ↪ :\t%s|␣processid:\t%s", rs.
                            ↪ getString(1), rs.getString(2),
                                    rs.getString(3));
                }
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
                return;
        }
}

/*
 * Retrieve the customers (in name order) whose category
     ↪ is in a given range
```

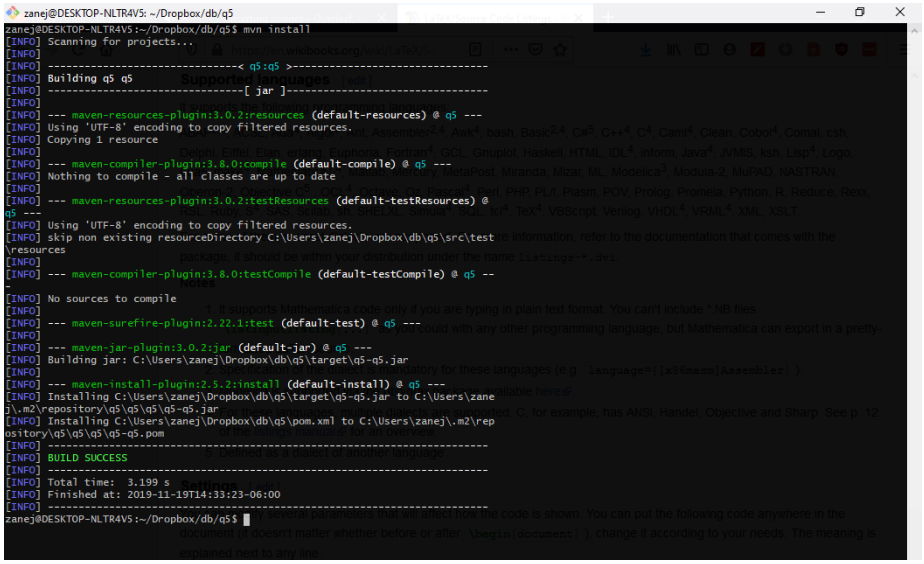
```
 */
static void choiceThirteen() {
        System.out.print("\nEnter␣the␣customer␣category␣
            ↪ lower␣bound:\t");
        final String lowerbound = in.nextLine();
        System.out.print("\nEnter␣the␣customer␣category␣
            ↪ upper␣bound:\t");
        final String upperbound = in.nextLine();
        final String query = "SELECT␣*␣FROM␣customers␣WHERE
            ↪ ␣category␣BETWEEN␣" + lowerbound + "␣AND␣" +
            ↪  upperbound
                        + "ORDER␣BY␣name";
        try {

                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                while (rs.next()) {
                        System.out.printf("NAME:\t%s|␣
                            ↪ ADDRESS:\t%s|␣CATEGORY:\t%s",
                            ↪ rs.getString(1), rs.getString
                            ↪ (2),
                                        rs.getString(3));
                }
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
                return;
        }
}

/*
 * Delete all cut-jobs whose job-no is in a given range
 */
static void choiceFourteen() {
        System.out.print("\nEnter␣the␣job-no␣lower␣bound:\t
            ↪ ");
        final String lowerbound = in.nextLine();
        System.out.print("\nEnter␣the␣job-no␣upper␣bound:\t
            ↪ ");
        final String upperbound = in.nextLine();
```

```
        final String query = "DELETE␣FROM␣cut_jobs␣WHERE␣id
            ↪ ␣BETWEEN␣" + lowerbound + "␣AND␣" +
            ↪ upperbound + ";";
        try {
                statement.execute(query.toString());
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

static void choiceFifteen() {
        System.out.print("\nEnter␣the␣paint␣job␣id:\t");
        final String id = in.nextLine();
        System.out.print("\nEnter␣the␣new␣color:\t");
        final String color = in.nextLine().replaceAll("'",
            ↪ "''");// escape quotes
        final String query = "UPDATE␣paint_jobs␣SET␣
            ↪ paintcolor␣=␣'" + color + "'␣WHERE␣id=" + id
            ↪  + ";";
        try {
                statement.execute(query.toString());
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// load customers from file
// (format is a value list)
static void choiceSixteen() {
        File file;
        String record;
        System.out.print("\nEnter␣the␣filename:\t");
```

```
file = new File(in.nextLine());
// try to read the file
// (format is a value list)
BufferedReader br;
try {
       br = new BufferedReader(new FileReader(file)
           ↪ );
} catch (final FileNotFoundException e) {
       System.out.print("file␣not␣found");
       return;
}
// try to build the "values " clause in the sql
    ↪ statement
final StringBuilder query = new StringBuilder("
    ↪ INSERT␣INTO␣customers␣VALUES␣(");
try {
       // no comma before the first record
       if ((record = br.readLine()) == null) {
              System.out.println("Empty␣file");
              return;
       } else {
              query.append(record);
       }
       query.append(")");
       // build the "VALUES" clause in the sql
           ↪ statement
       while ((record = br.readLine()) != null) {
              query.append(",(" + record + ")");
       }
       query.append(";");
} catch (final IOException e) {
       System.out.println("an␣IO␣error␣has␣occoured
           ↪ ");
       return;
} finally {
       try {
              br.close();
       } catch (Exception e) {
              System.out.println("an␣IO␣error␣has␣
                  ↪ occoured");
              return;
       }
}
// insert
try {
       statement.execute(query.toString());
```

```
        } catch (final SQLException e) {
                System.out.println("A␣SQL␣Error␣has␣occoured
                    ↪ :");
                System.out.println(e);
                System.out.print("(state:");
                System.out.print(e.getSQLState() + ";");
                System.out.print("error␣code:");
                System.out.print(Integer.toString(e.
                    ↪ getErrorCode()) + ")");
        }
}

// save customers to file where customers in a given range
// (format is a value list)
static void choiceSeventeen() {
        PrintWriter out;
        System.out.print("\nEnter␣the␣customer␣category␣
            ↪ lower␣bound:\t");
        final String lowerbound = in.nextLine();
        System.out.print("\nEnter␣the␣customer␣category␣
            ↪ upper␣bound:\t");
        final String upperbound = in.nextLine();
        System.out.print("\nEnter␣the␣filename");
        final String filename = in.nextLine();
        final String query = "SELECT␣*␣FROM␣customers␣WHERE
            ↪ ␣category␣BETWEEN␣" + lowerbound + "␣AND␣" +
            ↪  upperbound
                        + "ORDER␣BY␣name";
        // try to open a file for writing
        // (format is a value list)
        try {
                out = new PrintWriter(filename);
        } catch (final Exception e) {
                System.out.println("Could␣not␣write␣to␣file␣
                    ↪ " + filename);
                return;
        }

        try {

                final ResultSet rs = statement.executeQuery(
                    ↪ query);
                while (rs.next()) {
                        out.printf("'%s','%s',%s\n", rs.
                            ↪ getString(1).replaceAll("'", "
                            ↪ ''"), // escape quotes
```

```
                                        rs.getString(2).
                                            ↪ replaceAll("'",
                                            ↪  "''"), rs.
                                            ↪ getString(3));
                        }
                        out.close();
                    } catch (final SQLException e) {
                        out.close();
                        System.out.println("A␣SQL␣Error␣has␣occoured
                            ↪ :");
                        System.out.println(e);
                        System.out.print("(state:");
                        System.out.print(e.getSQLState() + ";");
                        System.out.print("error␣code:");
                        System.out.print(Integer.toString(e.
                            ↪ getErrorCode()) + ")");
                        return;
                    }
                }
}
```

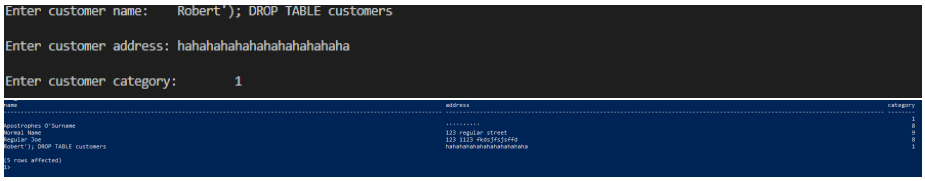


6 Task 6. Java Program Execution

6.1 screenshots showing the testing of query 1

```
PS C:\Users\zanej\Dropbox\db> & 'C:\Users\zanej\.vscode\extensions\vscjava.vscode-java-debug-0.23.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-13.0.1\bin\j
ess=localhost:63671' '-Dfile.encoding=UTF-8' '@C:\Users\zanej\AppData\Local\Temp\cp_9lcealupn3ucs44c29b5hz7xg.argfile' 'q5'
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
1
Enter customer name:    Normal Name

Enter customer address: 123 regular street

Enter customer category:        9

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
1
Enter customer name:    Regular Joe

Enter customer address: 123 1123 fkdsjfsjsffd

Enter customer category:        8
```

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
1
Enter customer name:    Apostrophes O'Surname

Enter customer address: ''''''''''

Enter customer category:        8

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
1
Enter customer name:

Enter customer address:

Enter customer category:        1
```


Enter customer name: Robert'); DROP TABLE customers
Enter customer address: hahahahahahahahahahaha
Enter customer category: 1
name address category
apostrophes O'Surname 1
Normal Name 123 regular street 9
Regular Joe 123 1123 4kdnjfejs4f2 8
Robert'); DROP TABLE customers hahahahahahahahahahaha 1
(5 rows affected)

6.2 screenshots showing the testing of query 2

```
2
Enter department data:  jfjjfjfjfjfjj

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
2
Enter department data:  `

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
2
Enter department data:  ()
```

```
2
Enter department data:  huehuehuehuehuehue

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
2
Enter department data:  ffffffffffffffffff

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
2
Enter department data:  fffffffffffeoejojojoejoeoej
```

```
1> select * from departments
2> go
id          departmentdata
----------- ----------------------------------------------------------------------------------------------------------------------------------------------------------------
          1 jfjjfjfjfjfjj
          2 '
          3 ()
          4 huehuehuehuehuehue
          5 fffffffffffffffffff
          6 fffffffffffeoejojojoejoeoej
          7 Real words.
          8 'hjfifi \n \n'
          9 2
         10 department data:

(10 rows affected)
```

6.3 screenshots showing the testing of query 3

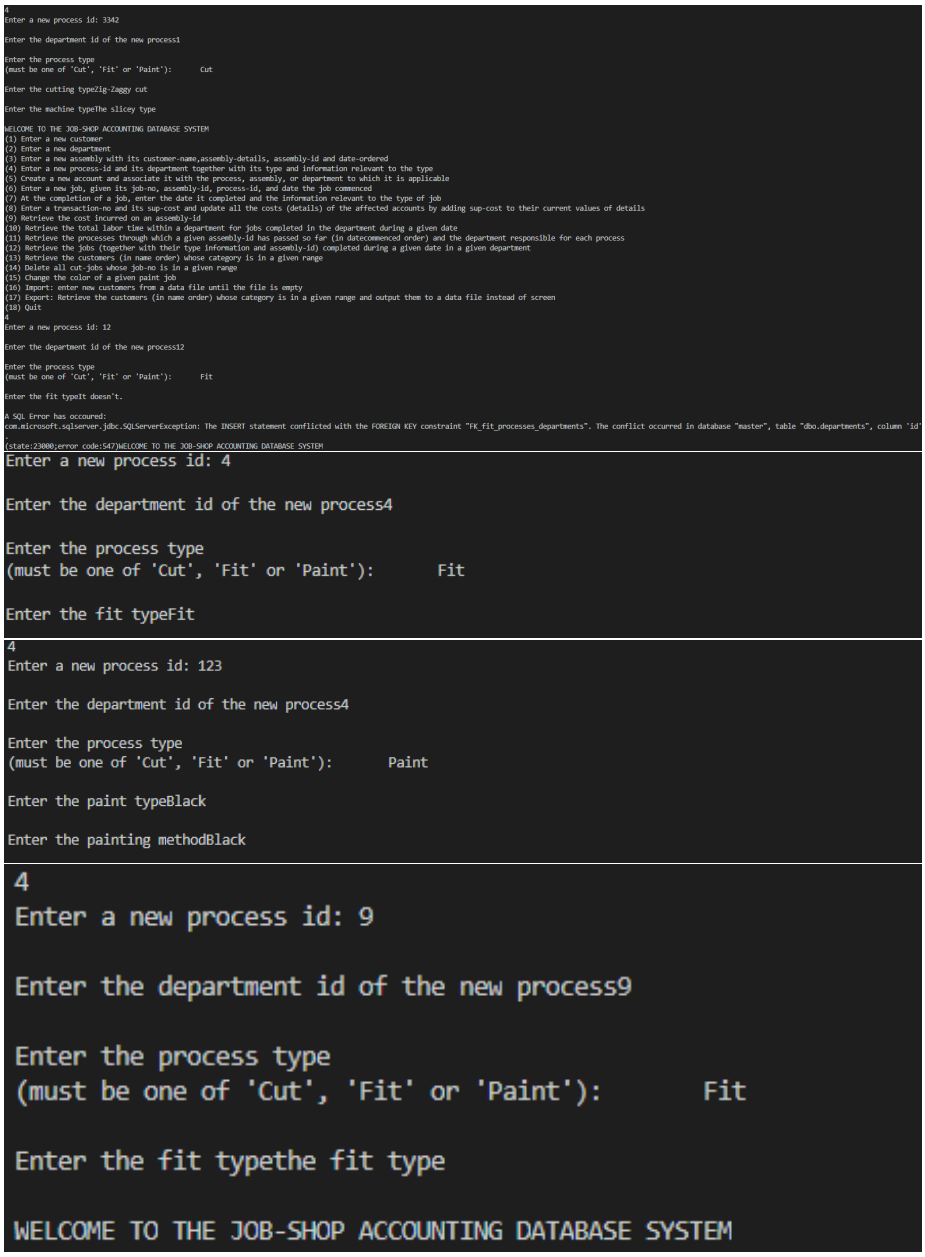
```
Enter customer name for this assembly:

Enter assembly date and time ordered
yyyymmdd hh:mm:ss {am|pm}; eg 20100202 10:10:10 am for february 2nd 2010 at 10:10:10 am):       20100202 10:10:12 am

Enter assembly details :        Assembly details

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
3
Enter customer name for this assembly:

Enter assembly date and time ordered
yyyymmdd hh:mm:ss {am|pm}; eg 20100202 10:10:10 am for february 2nd 2010 at 10:10:10 am):       17760704 04:04:04 pm

Enter assembly details :        jkfjdkjdffddfkdfkkjkfdkjdfjk
```

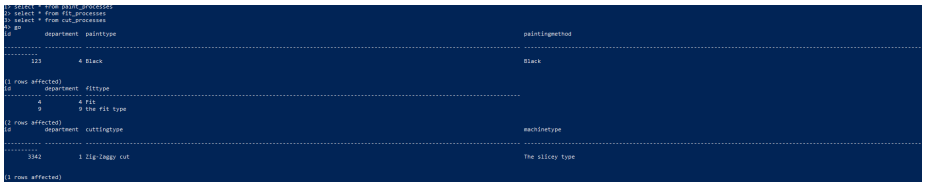
```
Enter customer name for this assembly:  Normal Name

Enter assembly date and time ordered
yyyymmdd hh:mm:ss {am|pm}; eg 20100202 10:10:10 am for february 2nd 2010 at 10:10:10 am):       20201010 00:00:00 pm

Enter assembly details :        Normal assembly

A SQL Error has occoured:
com.microsoft.sqlserver.jdbc.SQLServerException: The conversion of a varchar data type to a datetime data type resulted in an out-of-range value.
(state:S0003;error code:242)WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
3
Enter customer name for this assembly:  Nots Mc'Customer

Enter assembly date and time ordered
yyyymmdd hh:mm:ss {am|pm}; eg 20100202 10:10:10 am for february 2nd 2010 at 10:10:10 am):       20101001 01:02:03 pm

Enter assembly details :        not gonna be assembled

A SQL Error has occoured:
com.microsoft.sqlserver.jdbc.SQLServerException: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_customer_assembly". The conflict occurred in database "master", table "dbo.customers", column 'name'.
(state:23000;error code:547)WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
```

```
Enter customer name for this assembly:  Regular Joe

Enter assembly date and time ordered
yyyymmdd hh:mm:ss {am|pm}; eg 20100202 10:10:10 am for february 2nd 2010 at 10:10:10 am):

Enter assembly details :
```

```
1> select * from assemblies
2> go
id          customer_name                                                                                                    dateordered             assemblydetails
----------- ---------------------------------------------------------------------------------------------------------------- ----------------------- ----------------------------------------------------------------------------------------------------------------------------------------------------------------
          1                                                                                                                  2010-02-02 10:10:12.000 Assembly details
          2                                                                                                                  1776-07-04 16:04:04.000 jkfjdkjdffddfkdfkkjkfdkjdfjk
          5 Regular Joe                                                                                                      1900-01-01 00:00:00.000

(3 rows affected)
1>
```

6.4 screenshots showing the testing of query 4




1> select * from paint_processes
2> select * from fit_processes
3> select * from cut_processes
4> go
id department painttype paintingmethod
123 4 Black Black
(1 rows affected)
id department fittype
4 4 fit
9 9 the fit type
(2 rows affected)
id department cuttingtype machinetype
3342 1 Zig-Zaggy cut The slicey type
(1 rows affected)

6.5 screenshots showing the testing of query 5

```
5
Enter the account type
Must be one of Process, Assembly or Department: Process

Enter the accounts unique number:      1234

Enter the Id associated with the Process4

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

```
 Must be one of Process, Assembly or Department: Assembly

 Enter the accounts unique number:       444

 Enter the Id associated with the Assembly1

 WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
 (1) Enter a new customer
 (2) Enter a new department
 (3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
 (4) Enter a new process-id and its department together with its type and information relevant to the
 (5) Create a new account and associate it with the process, assembly, or department to which it is a
 (6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
 (7) At the completion of a job, enter the date it completed and the information relevant to the type
 (8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accou
 (9) Retrieve the cost incurred on an assembly-id
 (10) Retrieve the total labor time within a department for jobs completed in the department during a
 (11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced or
 (12) Retrieve the jobs (together with their type information and assembly-id) completed during a giv
 (13) Retrieve the customers (in name order) whose category is in a given range
 (14) Delete all cut-jobs whose job-no is in a given range
 (15) Change the color of a given paint job
 (16) Import: enter new customers from a data file until the file is empty
 (17) Export: Retrieve the customers (in name order) whose category is in a given range and output th
em ) Quit
 to a data file instead of screen
 (18) Quit
 5
 Enter the account type
 Must be one of Process, Assembly or Department: Department

 Enter the accounts unique number:       431

 Enter the Id associated with the Department1

 WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
 (1) Enter a new customer
 (2) Enter a new department
 (3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
 (4) Enter a new process-id and its department together with its type and information relevant to the
  ty At the completion of a job, enter the date it completed and the information relevant to the type
 (17) Export: Retrieve the customers (in name order) whose category is in a given range and output th
em to a data file instead of screen
 (18) Quit
 5
 Enter the account type
 Must be one of Process, Assembly or Department: Department

 Enter the accounts unique number:       431

 Enter the Id associated with the Department1

 WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
 (1) Enter a new customer
 (2) Enter a new department
 (3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
 (4) Enter a new process-id and its department together with its type and information relevant to the
  type
 (5) Create a new account and associate it with the process, assembly, or department to which it is a
pplicable
 (6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
 (7) At the completion of a job, enter the date it completed and the information relevant to the type
  of job
 (8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accou
nts by adding sup-cost to their current values of details
 (9) Retrieve the cost incurred on an assembly-id
 (10) Retrieve the total labor time within a department for jobs completed in the department during a
  given date
 (11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced or
der) and the department responsible for each process
 (12) Retrieve the jobs (together with their type information and assembly-id) completed during a giv
en 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
```

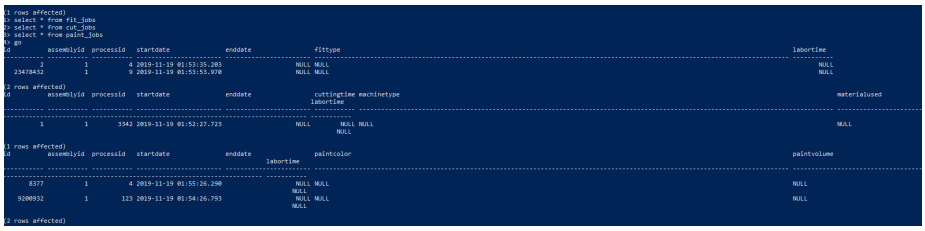
```
1> select * from Assembly_Accounts
2> go
id          dateestablished         assemblyid
----------- ----------------------- -----------
        444 2019-11-19 01:22:18.257           1

(1 rows affected)
1> select * from Department_accounts
2> go
id          dateestablished         departmentid
----------- ----------------------- ------------
        431 2019-11-19 01:22:47.290            1
       4444 2019-11-19 01:24:06.473            2

(2 rows affected)
1> select * from process_accounts
2> go
id          dateestablished         processid
----------- ----------------------- -----------
       1234 2019-11-19 01:20:39.610           4

(1 rows affected)
```

6.6 screenshots showing the testing of query 6

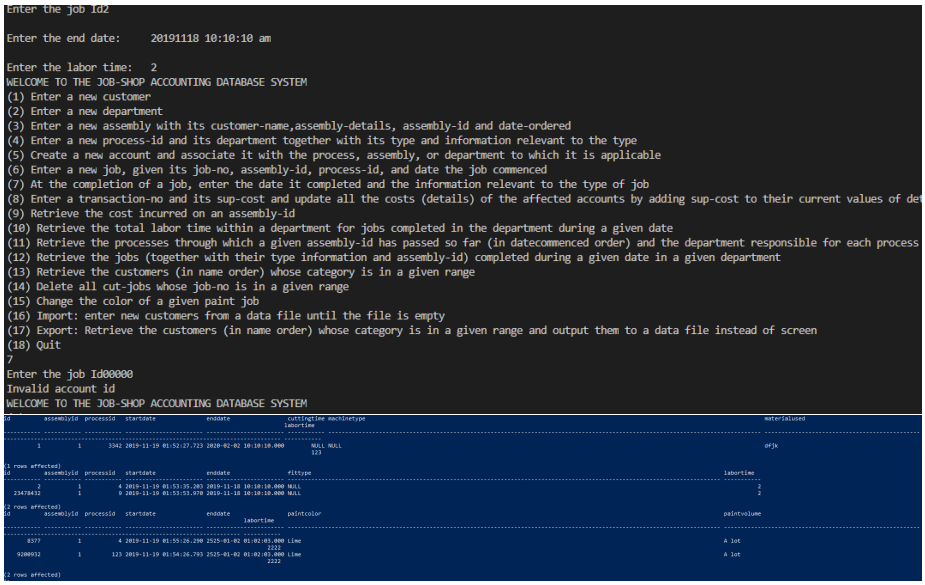
```
Enter the job type
(must be one of 'Cut', 'Fit' or 'Paint'):       Paint

Enter a unique job Id:  8377

Enter an assembly Id:   1

Enter a process Id:     4
```

```
Enter the job type
(must be one of 'Cut', 'Fit' or 'Paint'):       Cut

Enter a unique job Id:  1

Enter an assembly Id:   1

Enter a process Id:     3342

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
6

Enter the job type
(must be one of 'Cut', 'Fit' or 'Paint'):       Fit

Enter a unique job Id:  2

Enter an assembly Id:   1

Enter a process Id:     4
```

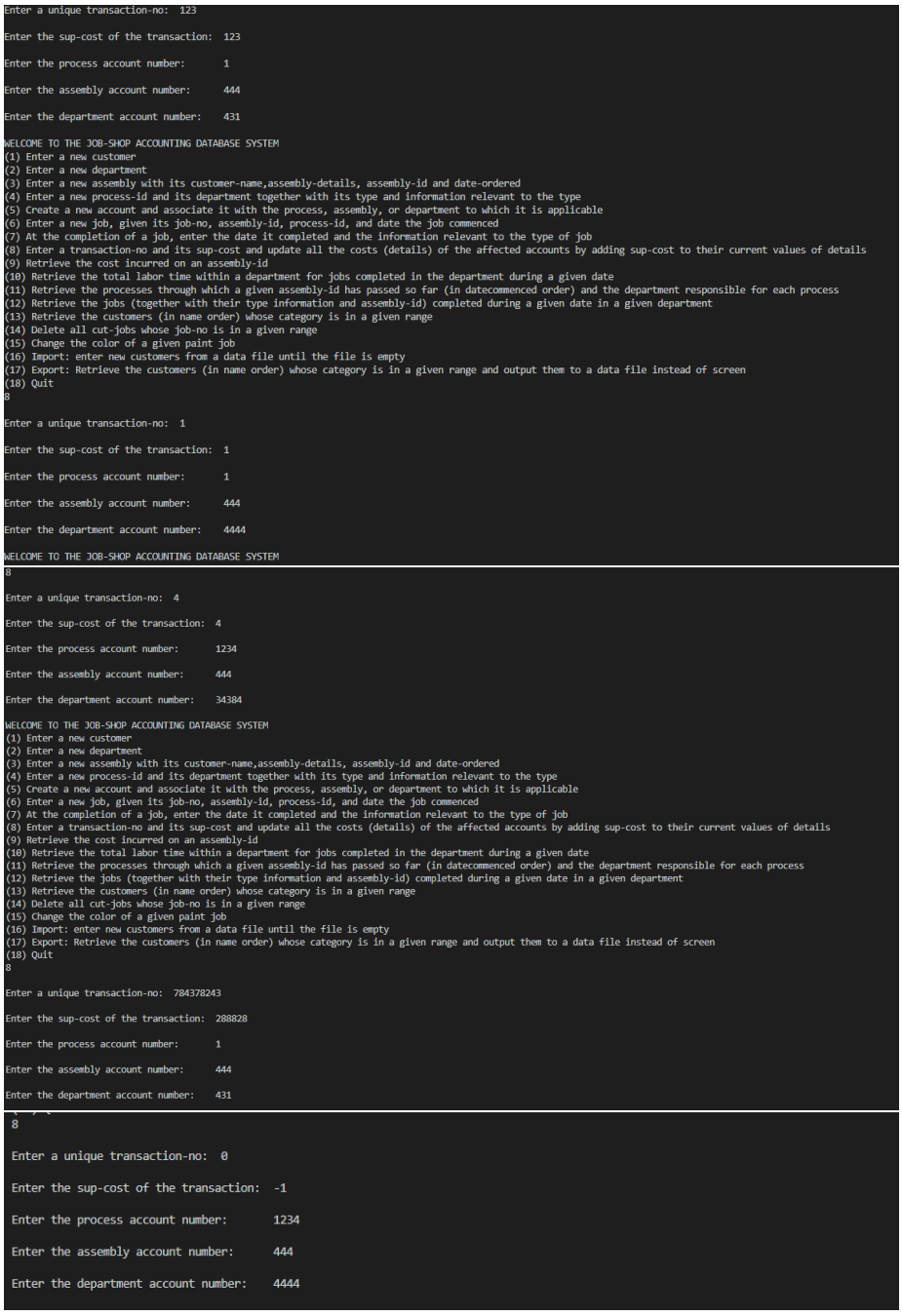
```
Enter the job type
(must be one of 'Cut', 'Fit' or 'Paint'):       Fit

Enter a unique job Id:  23478432

Enter an assembly Id:   1

Enter a process Id:     9

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
6

Enter the job type
(must be one of 'Cut', 'Fit' or 'Paint'):       Paint

Enter a unique job Id:  9200932

Enter an assembly Id:   1

Enter a process Id:     123
```



6.7 screenshots showing the testing of query 7


Enter the job id2
Enter the end date: 20191118 10:10:10 am
Enter the labor time: 2
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of det
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
7
Enter the job Id00000
Invalid account id
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1 rows effected)
(2 rows effected)
(2 rows effected)

6.8 screenshots showing the testing of query 8



```
(0 rows affected)
1> select * from transactions
2> go
id          supcost              assemblyaccountid processaccountid departmentaccountid
----------- -------------------- ----------------- ---------------- -------------------
          0              -1.0000               444             1234                4444
          1               1.0000               444                1                4444
          4               4.0000               444             1234               34384
        123             123.0000               444                1                 431
  784378243          288828.0000               444                1                 431

(5 rows affected)
```

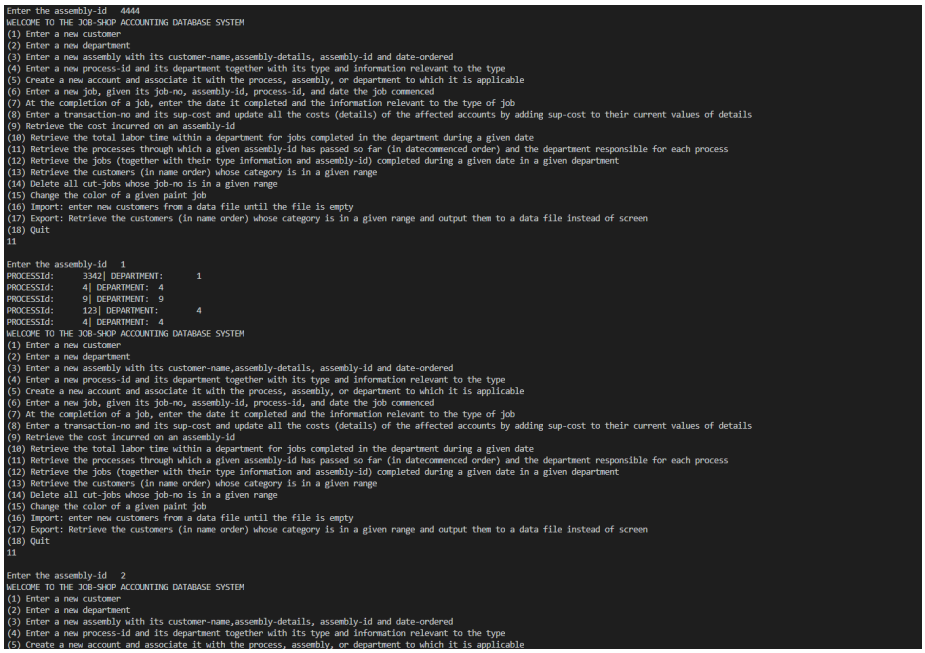
6.9 screenshots showing the testing of query 9

```
Enter an assembly number:      1

Total cost:      288955.0000
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
9

Enter an assembly number:      2

Total cost:      null
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible for each process
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
9

Enter an assembly number:      5
```

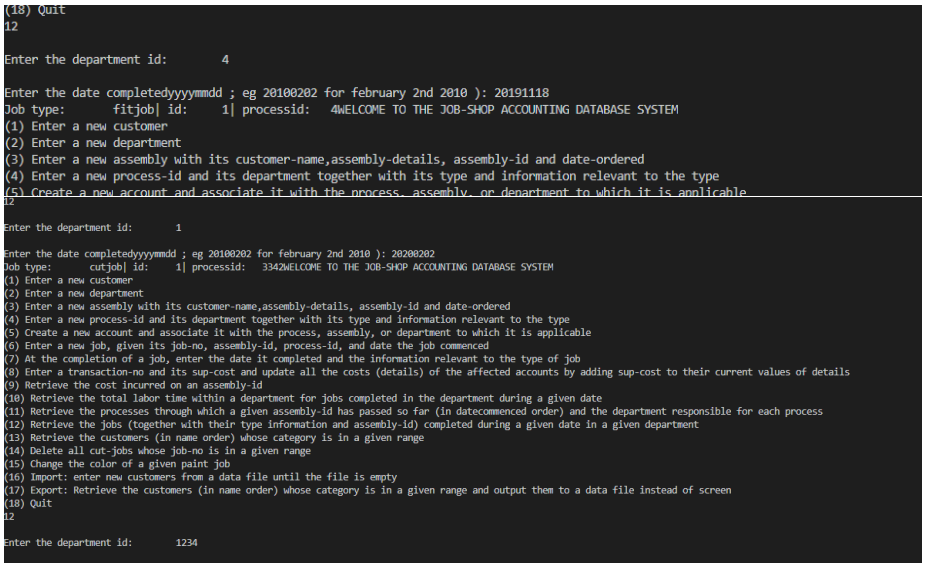
6.10 screenshots showing the testing of query 10

```
10

Enter the department id:        4

Enter the date completedyyyymmdd ; eg 20100202 for february 2nd 2010 ): 20191119
Total labor time :null
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their curr
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
10

Enter the department id:        4

Enter the date completedyyyymmdd ; eg 20100202 for february 2nd 2010 ): 20191118
Total labor time :2
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Enter a new customer
(2) Enter a new department
(3) Enter a new assembly with its customer-name,assembly-details, assembly-id and date-ordered
(4) Enter a new process-id and its department together with its type and information relevant to the type
(5) Create a new account and associate it with the process, assembly, or department to which it is applicable
(6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced
(7) At the completion of a job, enter the date it completed and the information relevant to the type of job
(8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their curr
(9) Retrieve the cost incurred on an assembly-id
(10) Retrieve the total labor time within a department for jobs completed in the department during a given date
(11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order) and the department responsible
(12) Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department
(13) Retrieve the customers (in name order) whose category is in a given range
(14) Delete all cut-jobs whose job-no is in a given range
(15) Change the color of a given paint job
(16) Import: enter new customers from a data file until the file is empty
(17) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen
(18) Quit
10

Enter the department id:        9

Enter the date completedyyyymmdd ; eg 20100202 for february 2nd 2010 ): 20191118
Total labor time :2
```

6.11 screenshots showing the testing of query 11

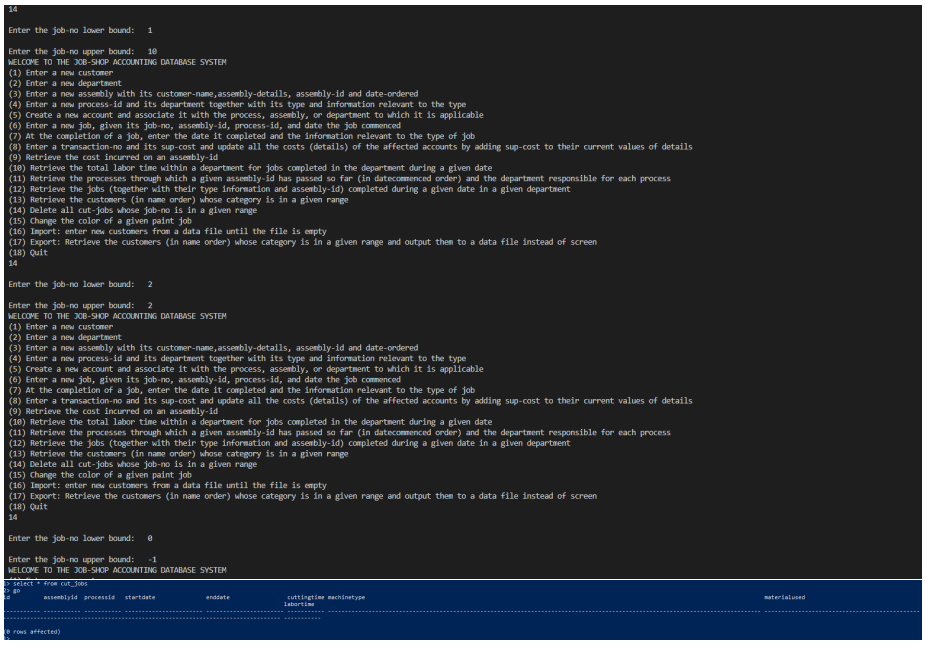


6.12 screenshots showing the testing of query 12



6.13 screenshots showing the testing of query 13

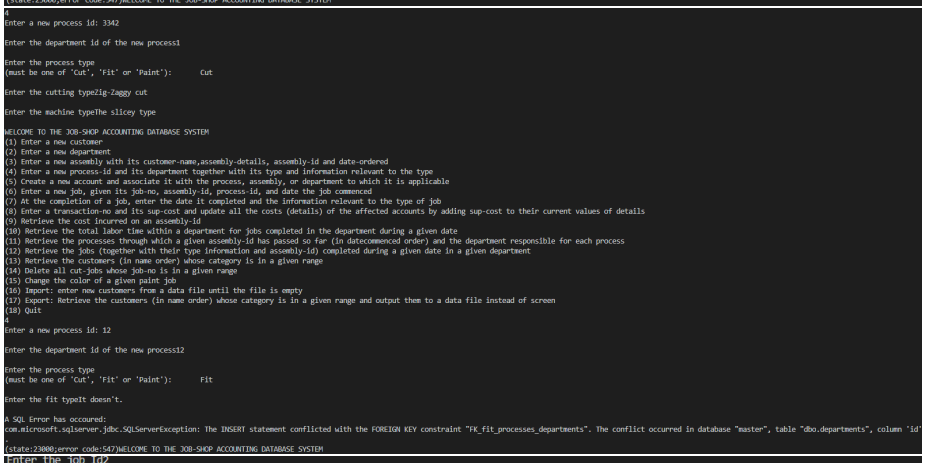
6.14 screenshots showing the testing of query 14

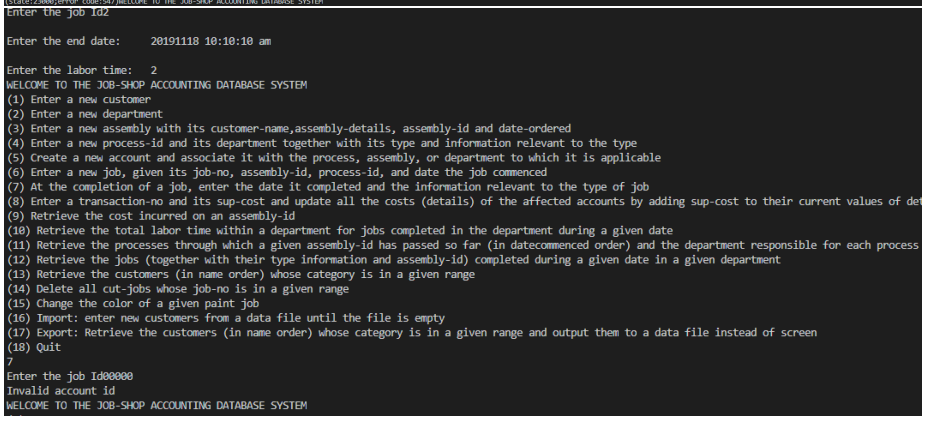


6.15 screenshots showing the testing of query 15

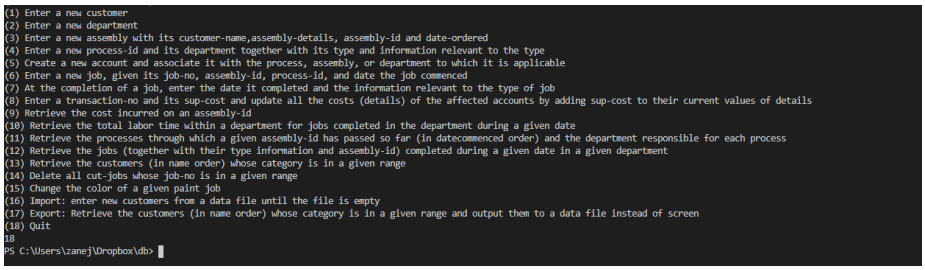
6.16 screenshot showing the testing of the import and export options

6.17 screenshots showing the testing of three types of errors





6.18 Screenshot showing the testing of the quit option



7 Task 7. Web database application and its execution

7.1 Web database application source program and screenshots showing its successful compilation

```
package jsp_azure_test;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.*;
import java.text.*;
import java.lang.String.*;
import java.util.*;
import microsoft.sql.*;
import java.sql.*;

public class DataHandler {
  private Connection conn;
  // Azure SQL connection credentials
  // Resulting connection string
  final private String url = "jdbc:sqlserver://localhost:1433;
      ↪ user=sa;password=Jbermine41611";

  // Initialize and save the database connection
  private void getDBConnection() throws SQLException,
      ↪ ClassNotFoundException{
    if (conn != null) {
      return;
    }
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    this.conn = DriverManager.getConnection(url);
  }

  // Return the result of selecting everything from the customers
      ↪  table
  public ResultSet getAllMovies(int lower, int upper) throws
      ↪ ClassNotFoundException, SQLException {
    getDBConnection();
    final String sqlQuery = "SELECT␣*␣FROM␣customers␣where␣
        ↪ category␣between␣?␣and␣?;";
```

```
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery)
        ↪ ;
    stmt.setInt(1,lower);
    stmt.setInt(2,upper);
    return stmt.executeQuery();
  }

  // Inserts a record into the customers table with the given
      ↪ attribute values
  public boolean addMovie(String name, String movieName, int
      ↪ category) throws SQLException, ClassNotFoundException{
    getDBConnection(); // Prepare the database connection
    // Prepare the SQL statement
    final String sqlQuery = "INSERT␣INTO␣customers␣" + "(name,␣
        ↪ address␣,␣category)␣" + "VALUES␣" + "(?,␣?,␣?)";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery)
        ↪ ;
    // Replace the '?' in the above statement with the given
        ↪ attribute values
    stmt.setString(1, name);
    stmt.setString(2, movieName);
    stmt.setInt(3, category);
    // Execute the query, if only one record is updated, then we
        ↪ indicate success by
    // returning true
    return stmt.executeUpdate() == 1;
  }
}

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
    ↪ http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF
      ↪ -8">
  <title>Query Result</title>
</head>

<body>
  <%@page import="jsp_azure_test.DataHandler"%>
  <%@page import="java.sql.ResultSet"%>
  <%@page import="java.sql.Array"%>
  <%
```

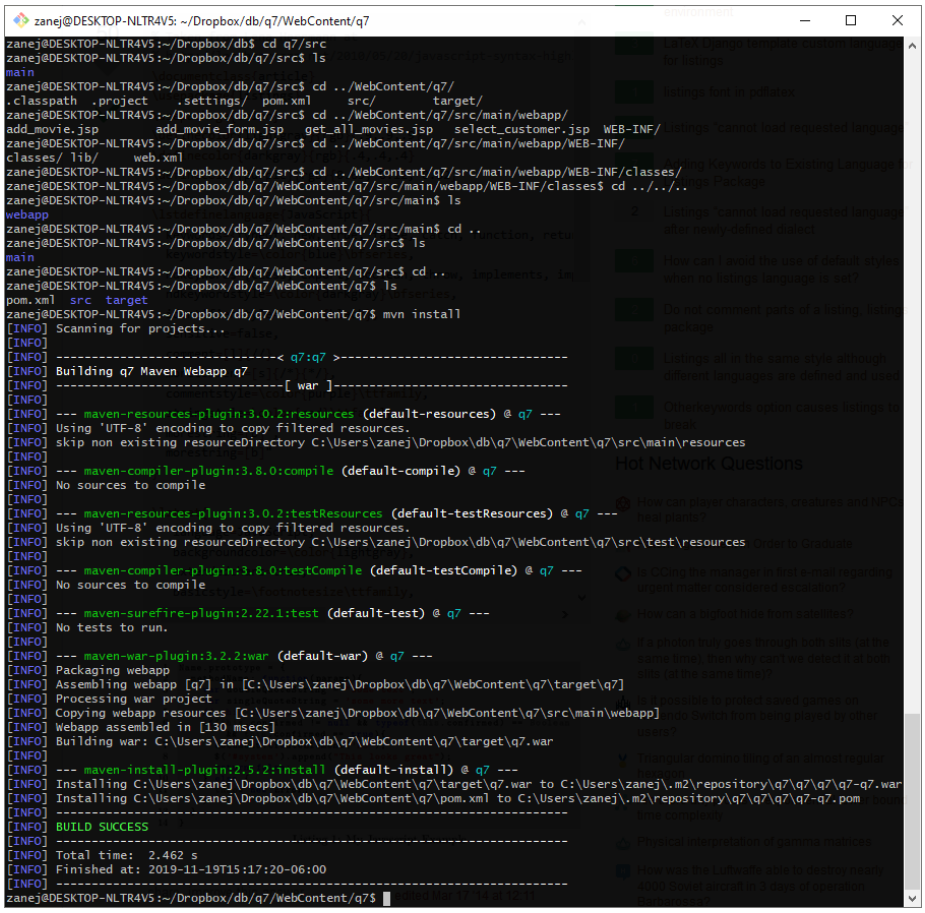
```
                        // The handler is the one in charge of
                            ↪ establishing the connection.
                        DataHandler handler = new DataHandler();
                        // Get the attribute values passed from
                            ↪ the input form.
                        String name = request.getParameter("name
                            ↪ ");
                        String address = request.getParameter("
                            ↪ address");
                        String durationString = request.
                            ↪ getParameter("category");
                        /*
                        * If the user hasn't filled out all the
                            ↪ time, movie name and duration.
                            ↪ This is very simple checking.
                        */
                        if (name==null||address==null||
                            ↪ durationString==null||name.equals
                            ↪ ("") || address.equals("") ||
                            ↪ durationString.equals("")) {
                        response.sendRedirect("add_movie_form.jsp
                            ↪ ");
                        } else {
                        int category = Integer.parseInt(
                            ↪ durationString);
                        // Now perform the query with the data
                            ↪ from the form.
                        boolean success = handler.addMovie(name,
                            ↪ address, category);
                        try{if (!success) { // Something went
                            ↪ wrong
                        %>
<h2>There was a problem inserting the customer</h2>
<%
                        } else { // Confirm success to the user
                        %>
<h2>The customer:</h2>
<ul>
  <li>Name: <%=name%></li>
  <li>Address: <%=address%></li>
  <li>category: <%=durationString%></li>

</ul>
<h2>Was successfully inserted.</h2>
<a href="get_all_movies.jsp">See all customers.</a>
<%
```

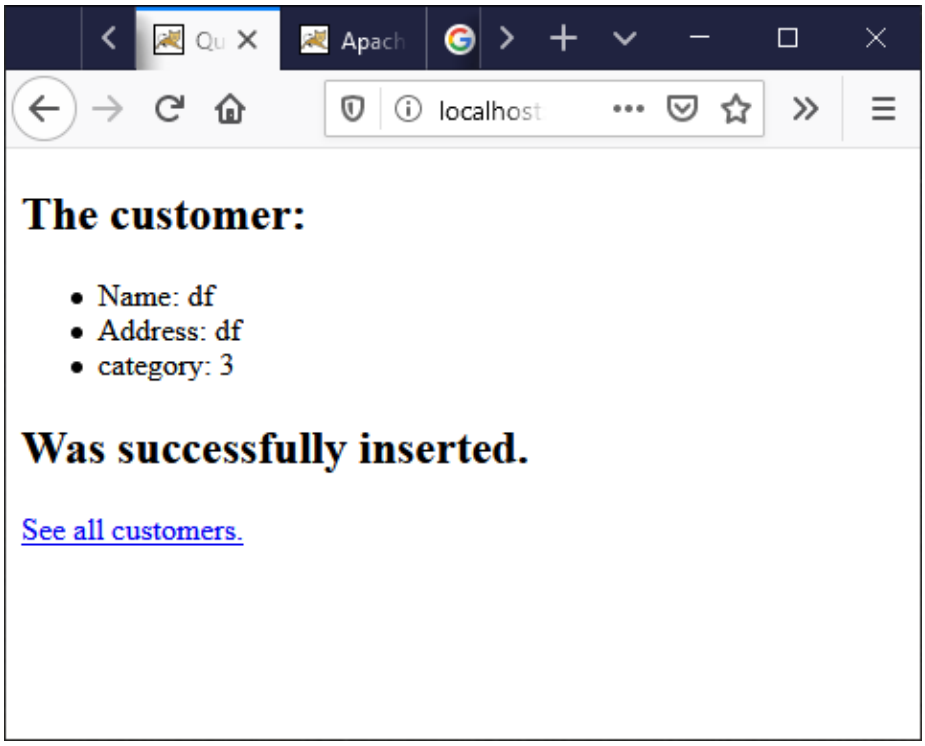
```
                        }
                        }catch(Exception e){// Something went
                            ↪ wrong
                        %>
  <h2>There was a problem inserting the customer</h2>
  <% } } %>
</body>

</html>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Add customer</title>
  </head>
  <body>
    <h2>Add customer </h2>
    <!--
          Form for collecting user input for the new movie_night
              ↪ record.
          Upon form submission, add_movie.jsp file will be invoked
              ↪ .
          -->
          <form action="add_movie.jsp">
            <!-- The form organized in an HTML table for better
                ↪ clarity. -->
            <table border=1>
              <tr>
                <th colspan="2">Enter the customer Data:</th>
              </tr>
              <tr>
                <td>Name:</td>
                <td><div style="text-align: center;">
                    <input type=text name=name>
                  </div></td>
              </tr>
              <tr>
                <td>address:</td>
                <td><div style="text-align: center;">
                    <input type=text name=address>
                  </div></td>
              </tr>
              <tr>
                <td>category:</td>
                <td><div style="text-align: center;">
```

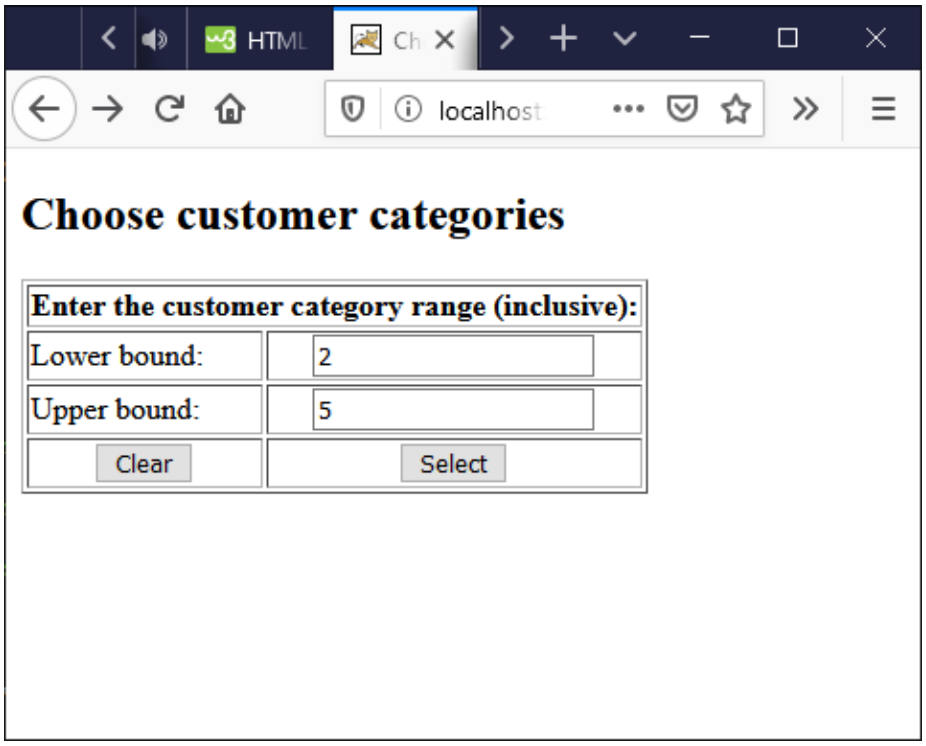
```
                <input type=text name=category>
              </div></td>
            </tr>
                        <tr>
              <td><div style="text-align: center;">
                <input type=reset value=Clear>
              </div></td>
              <td><div style="text-align: center;">
                  <input type=submit value=Insert>
              </div></td>
            </tr>
          </table>
        </form>
  </body>
</html>

<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Choose customer categories</title>
</head>

<body>
  <h2>Choose customer categories</h2>
  <!--
        Form for collecting user input for the new movie_night
            ↪ record.
        Upon form submission, add_movie.jsp file will be invoked
            ↪ .
        -->
  <form action="select_customer.jsp">
    <!-- The form organized in an HTML table for better clarity.
        ↪ -->
    <table border=1>
      <tr>
        <th colspan="2">Enter the customer category range (
            ↪ inclusive):</th>
      </tr>
      <tr>
        <td>Lower bound:</td>
        <td>
          <div style="text-align: center;">
            <input type=text name=lower>
          </div>
```

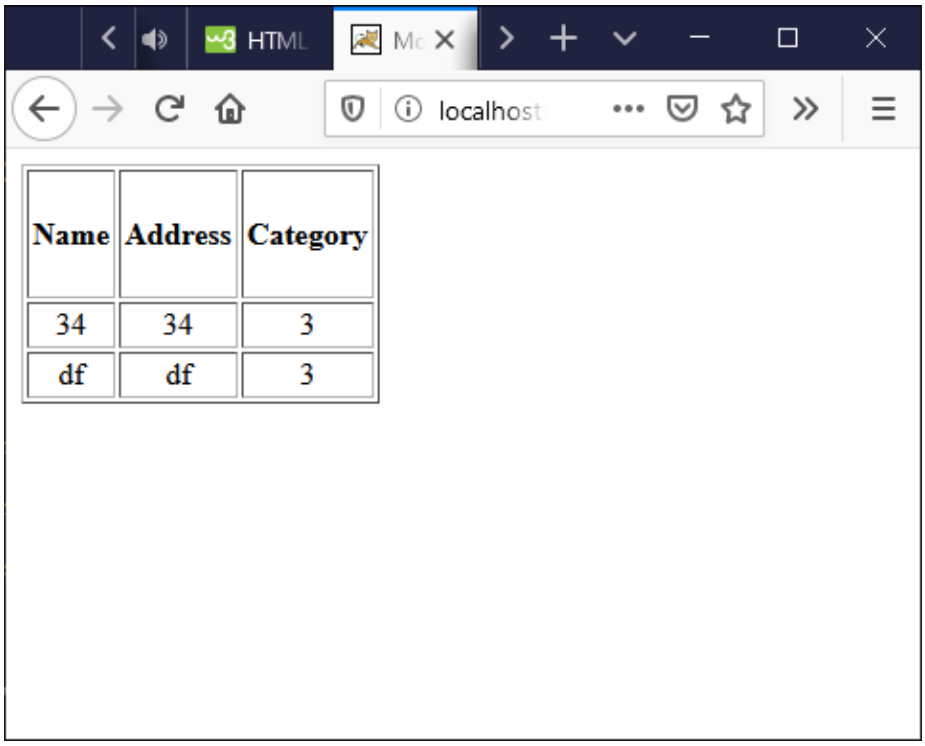
```
        </td>
      </tr>
      <tr>
        <td>Upper bound:</td>
        <td>
          <div style="text-align: center;">
            <input type=text name=upper>
          </div>
        </td>
      </tr>
      <tr>
        <td>
          <div style="text-align: center;">
            <input type=reset value=Clear>
          </div>
        </td>
        <td>
          <div style="text-align: center;">
            <input type=submit value=Select>
          </div>
        </td>
      </tr>
    </table>
  </form>
</body>

</html>

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

<head>
  <meta charset="UTF-8">
  <title>Movie Nights</title>
</head>

<body>
  <%@page import="jsp_azure_test.DataHandler"%>
  <%@page import="java.sql.ResultSet"%>
  <%
    // We instantiate the data handler here, and get all the
        ↪ movies from the database
    final DataHandler handler = new DataHandler();
    String stLower = request.getParameter("lower");
```

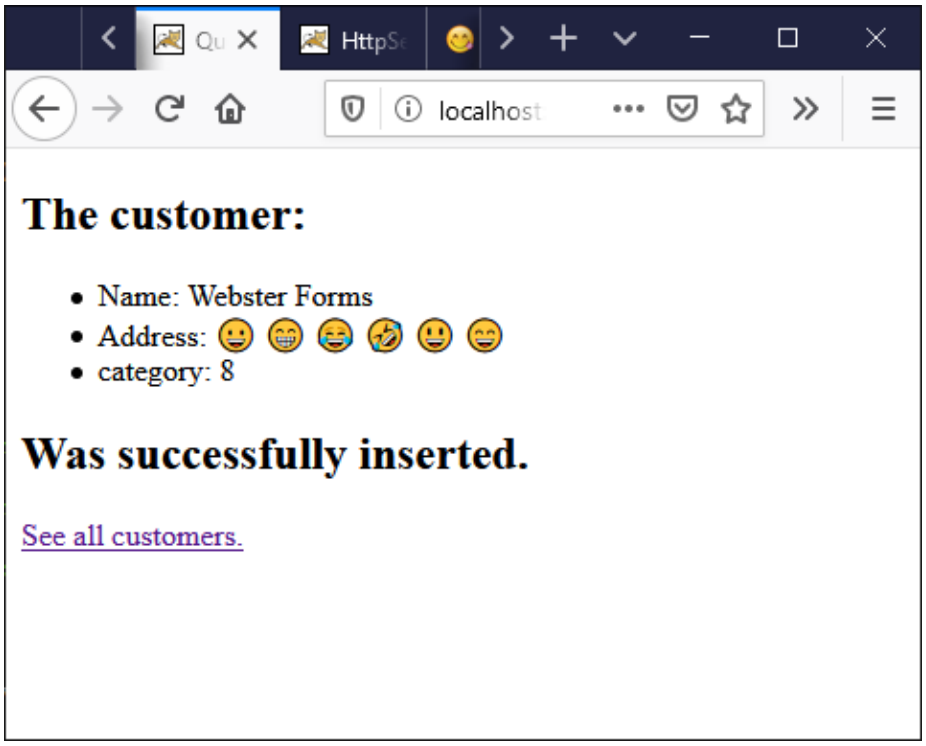
```
    String stUpper = request.getParameter("upper");
    if(stLower==null||stUpper==null||stLower==""||stUpper==""){
      response.sendRedirect("select_customer.jsp");
    }else{
      int lower = Integer.parseInt(stLower);
      int upper = Integer.parseInt(stUpper);
      final ResultSet movies = handler.getAllMovies(lower, upper);
    %>
  <!-- The table for displaying all the movie records -->
  <table cellspacing="2" cellpadding="2" border="1">
    <tr>
      <!-- The table headers row -->
      <td align="center">
        <h4>Name</h4>
      </td>
      <td align="center">
        <h4>Address</h4>
      </td>
      <td align="center">
        <h4>Category</h4>
      </td>
    </tr>
    <%
      while(movies.next()) { // For each movie_night record
          ↪ returned...
      // Extract the attribute values for every row returned
      final String name = movies.getString("name");
      final String address = movies.getString("address");
      final String category = movies.getString("category");
       out.println("<tr>"); // Start printing out the new table
           ↪ row
        out.println( // Print each attribute value
        "<td align=\"center\">" + name +
          "</td><td align=\"center\"> " + address +
          "</td><td align=\"center\"> " + category +
          "</td>");
        out.println("</tr>");
      }
    }
  %>
  </table>
</body>

</html>
```

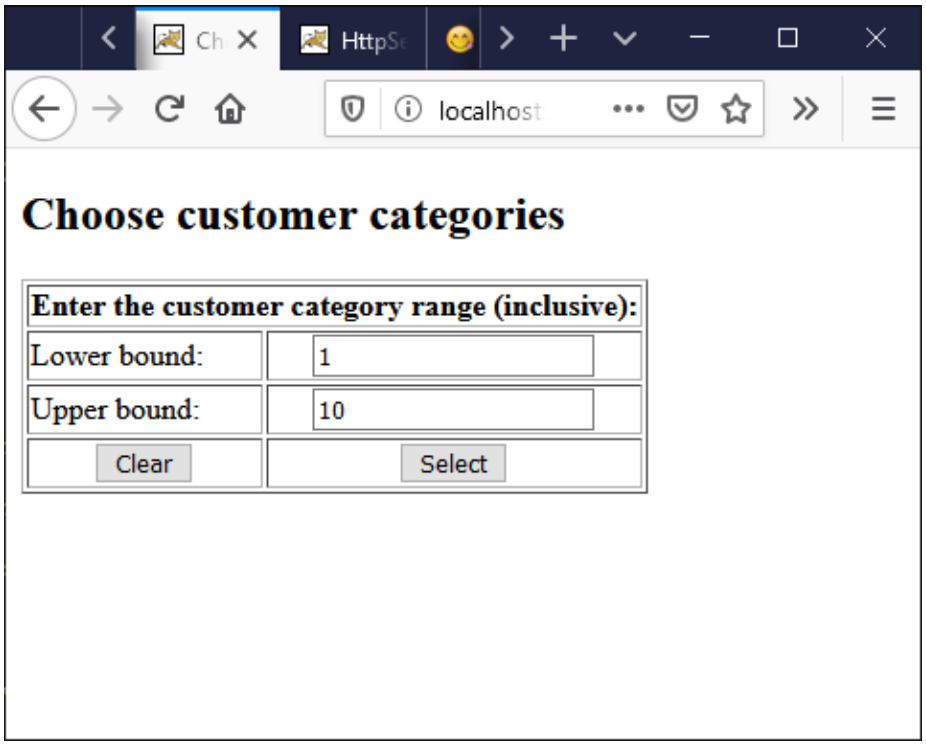

zanej@DESKTOP-NLTR4V5: ~/Dropbox/db/q7/WebContent/q7
zanej@DESKTOP-NLTR4V5:~/Dropbox/db$ cd q7/src
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/src$ ls
main
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/src$ cd ../WebContent/q7/
.classpath .project .settings/ pom.xml src/ target/
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/src$ cd ../WebContent/q7/src/main/webapp/
add_movie.jsp add_movie_form.jsp get_all_movies.jsp select_customer.jsp WEB-INF/
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/src$ cd ../WebContent/q7/src/main/webapp/WEB-INF/
classes/ lib/ web.xml
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/src$ cd ../WebContent/q7/src/main/webapp/WEB-INF/classes/
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7/src/main/webapp/WEB-INF/classes$ cd ../../..
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7/src/main$ ls
webapp
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7/src/main$ cd ..
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7/src$ ls
main
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7/src$ cd ..
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7$ ls
pom.xml src target
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------------< q7:q7 >----------------------------
[INFO] Building q7 Maven Webapp q7
[INFO] --------------------------------[war]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ q7 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\zanej\Dropbox\db\q7\WebContent\q7\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ q7 ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ q7 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\zanej\Dropbox\db\q7\WebContent\q7\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ q7 ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ q7 ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ q7 ---
[INFO] Packaging webapp
[INFO] Assembling webapp [q7] in [C:\Users\zanej\Dropbox\db\q7\WebContent\q7\target\q7]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Users\zanej\Dropbox\db\q7\WebContent\q7\src\main\webapp]
[INFO] Webapp assembled in [130 msecs]
[INFO] Building war: C:\Users\zanej\Dropbox\db\q7\WebContent\q7\target\q7.war
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ q7 ---
[INFO] Installing C:\Users\zanej\Dropbox\db\q7\WebContent\q7\target\q7.war to C:\Users\zanej\.m2\repository\q7\q7\q7\q7-q7.war
[INFO] Installing C:\Users\zanej\Dropbox\db\q7\WebContent\q7\pom.xml to C:\Users\zanej\.m2\repository\q7\q7\q7\q7-q7.pom
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 2.462 s
[INFO] Finished at: 2019-11-19T15:17:20-06:00
[INFO] --
zanej@DESKTOP-NLTR4V5:~/Dropbox/db/q7/WebContent/q7$

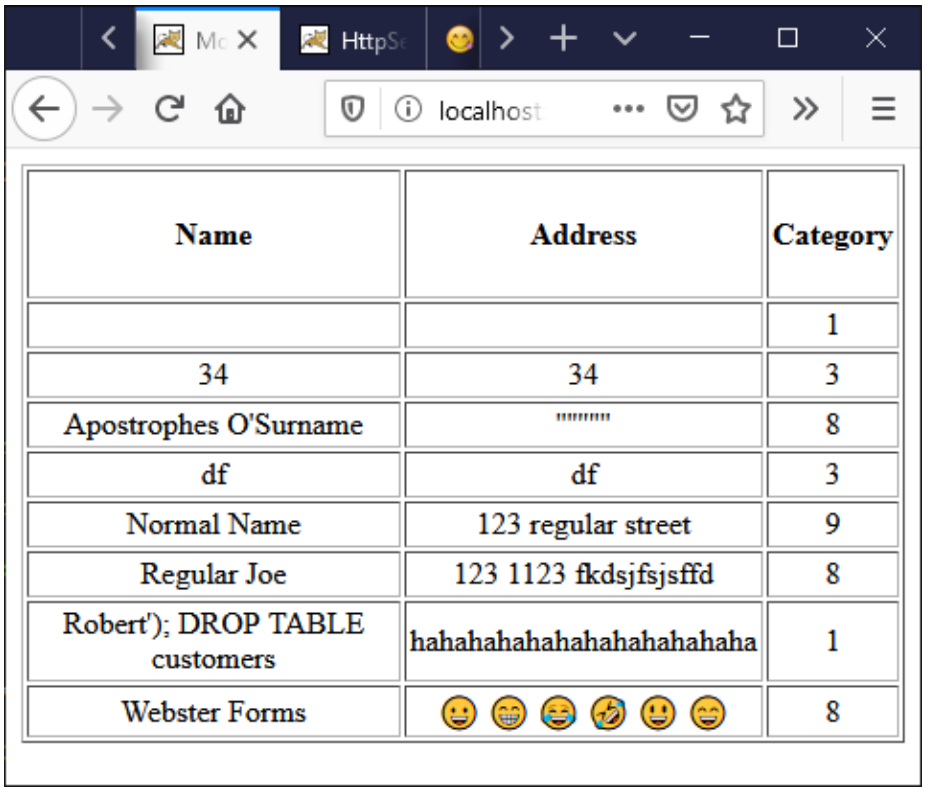

localhost
The customer:
Name: df
Address: df
category: 3
Was successfully inserted.
See all customers.


HTML
Ch
localhost
Choose customer categories
Enter the customer category range (inclusive):
Lower bound:
2
Upper bound:
5
Clear
Select


HTML
localhost
Name
Address
Category
34
34
3
df
df
3


localhost
The customer:
Name: Webster Forms
Address: 😀 😁 😂 🤣 😃 😄
category: 8
Was successfully inserted.
See all customers.


localhost
Choose customer categories
Enter the customer category range (inclusive):
Lower bound:
1
Upper bound:
10
Clear
Select



Name	Address	Category
		1
34	34	3
Apostrophes O'Surname	''''''''	8
df	df	3
Normal Name	123 regular street	9
Regular Joe	123 1123 fkdsjfsjsffd	8
Robert'); DROP TABLE customers	hahahahahahahahahahaha	1
Webster Forms	😀 😁 😂 🤣 😃 😄	8