**Wahidullah Haidari, 113523274, wahid@ou.edu**

**Assignment: Individual Project**

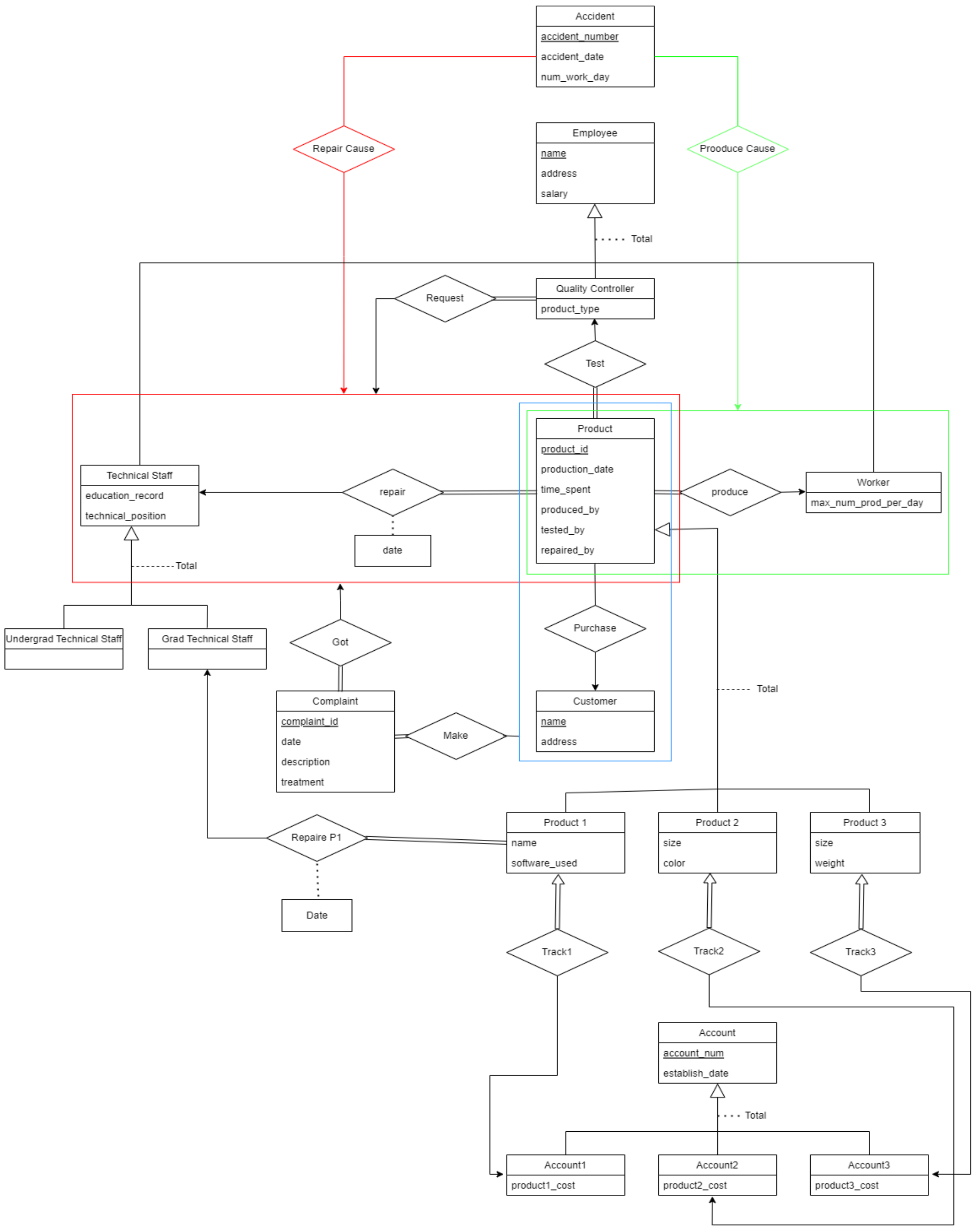
**Semester and Year: Fall 2022**

**Instructor: Dr. Le Gruenwald**

**SCORE:**

**Tasks Performed** **Page Number**

**Task 1: 1.1.** The ER Diagram



**1.2. Relational Database**

Worker (worker_name, addr , salary, max_num_prod_per_day)

QualityController (qual_cont_name, addr, salary, product_type)

TechnicalStaff (tech_staff_name, addr, salary, education_record, technical_position)

GradTechnicalStaff (tech_staff_name, addr, salary, education_record, technical_position)

UndergradTechnicalStaff (tech_staff_name, addr, salary, education_record, technical_position)

Account1(account1_num, establish_date, product1_cost)

Account2(account2_num, establish_date, product2_cost)

Account3(account3_num, establish_date, product3_cost)

Product (product_id, production_date, time_spent, produced_by, tested_by, repaired_by)

Product1 (product1_id, production_date, time_spent, produced_by, tested_by, repaired_by, name, software_used, account1_num)

Product2 (product2_id, production_date, time_spent, produced_by, tested_by, repaired_by, size, color, account2_num)

Product3 (product3_id, production_date, time_spent, produced_by, tested_by, repaired_by, size, weight, account3_num)

Produce (worker_name, product_id)

Test (qual_cont_name, product_id)

Repair (tech_staff_name, product_id, date)

RepairP1 (tech_staff_name, product_id, date)

Request (qual_cont_name, product_id, tech_staff_name)

Accident (accident_number, accident_date, num_work_day)

ProduceCause (accident_number, worker_name, product_id)

RepairCause (accident_number, tech_staff_name, product_id)

Customer (customer_name, addr)

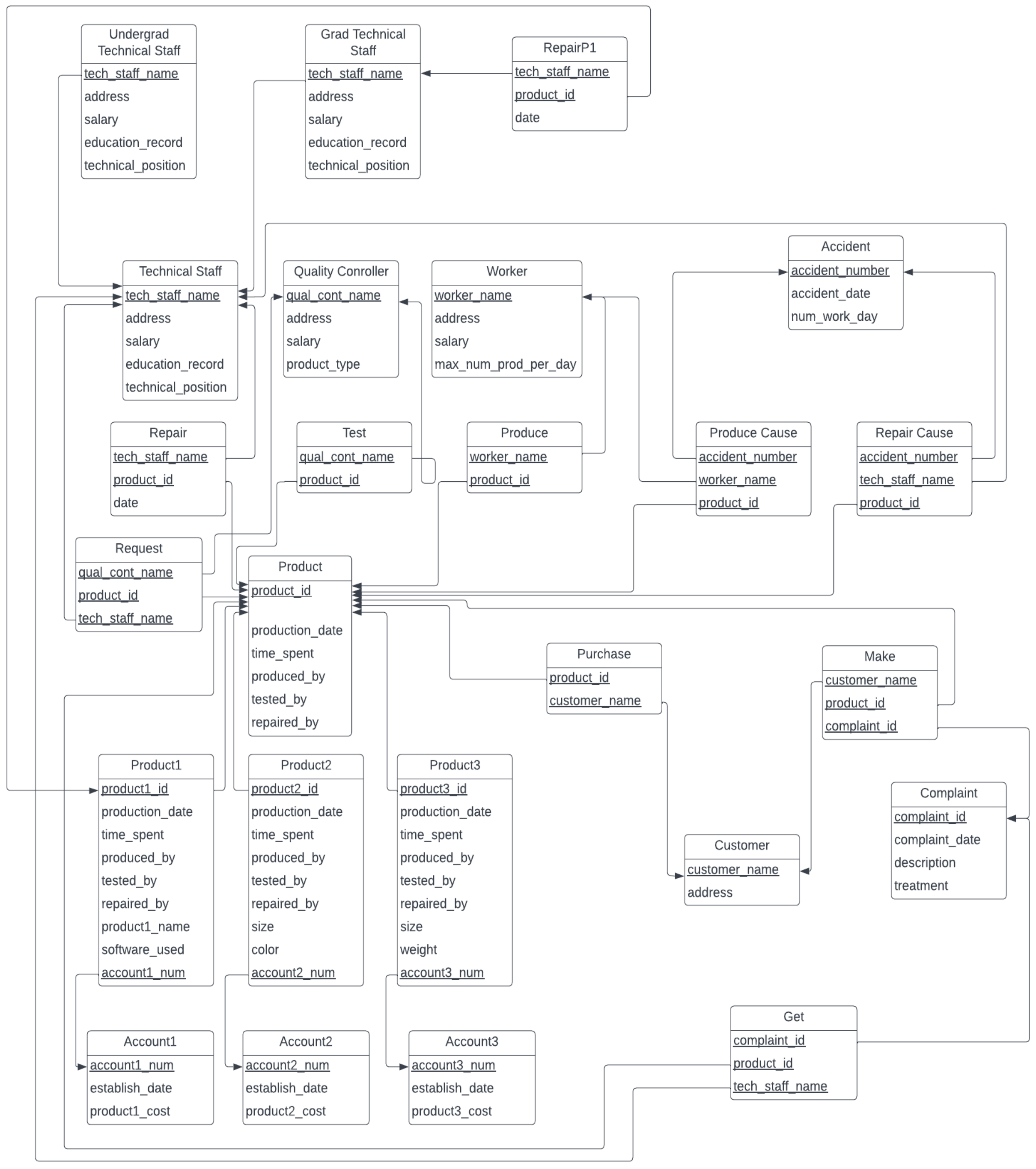
Purchase (product_id, customer_name)

Complaint (complaint_id, complaint_date, description, treatment)

Make (customer_name, product_id, complaint_id)

Got (complaint_id, product_id, tech_staff_name)

**Task 2:** Schema Diagram



**Task 3:**

**3.1.** Appropriate Storage Structure

| Table Name | Query# and Type | Search Key | Query Frequency | Selected Fie Organization | Justification |
|---|---|---|---|---|---|
| Worker | 1 (Insertion)<br>12 (range search) | | 2/month<br><br>1/month | Heap file | We insert more frequently than range search. |
| Quality Controller | 1 (Insertion)<br>12 (range search) | | 2/month<br><br>1/month | Heap file | We insert more frequently than range search |
| Technical Staff | 1 (Insertion)<br>12 (range search) | | 2/month<br>1/month | Heap file | We insert more frequently than range search |
| Grad Technical Staff | 1 (Insertion) | | 2/month | Heap file | We just insertion date |
| Undergrad Technical Staff | 1 (Insertion) | | 2/month | Heap file | We just insert date |
| Product | 2 (Insertion)<br>7 (random search)<br>8 (random search)<br>14 (random search) | worker_n ame | 400/day<br><br>100/day<br><br><br>2000/day<br><br>5/day | Hash File | The most frequent query is query 8, and it is a random search. Since the frequency is very high, we use Hash File because it is so fast. |
| Product1 | 2 (Insertion)<br>14 (random search) | | 400/day<br>5/day | Heap file | Insertion is done more frequently than random search. |
| Product2 | 2 (Insertion)<br>11 (random search)<br>14 (random search) | | 400/day<br>5/month<br><br>5/day | Heap file | Insertion is done more frequently than random search. |
| Product3 | 2 (Insertion)<br>10 (random search)<br>14 (random search) | | 400/day<br>40/day<br><br>5/day | Heap file | Insertion is done more frequently than random search. |
| Customer | 3 (Insertion)<br>11 (Random Search | | 50/day<br>5/Month | Heap file | Insertion is done more frequently than random search. |
| Account1 | 4 (Insertion)<br>14 (random search) | | 40/day<br>5/day | Heap file | Insertion is done more frequently than random search. |

| | | | | | |
|---|---|---|---|---|---|
| Account2 | 4 (Insertion)<br>14 (random search) | | 40/day<br>5/day | Heap file | Insertion is done more frequently than random search. |
| Account3 | 4 (Insertion)<br>10 (random search)<br>14 (random search) | | 40/day<br>40/day<br><br>5/day | Heap file | Since the frequency of insertion and search is the same, we use Heap file because it is easier to implement. |
| Complaint | 5 (Insertion) | | 30/day | Heap file | We only insert data. |
| Accident | 6 (Insertion)<br>13 (random search)<br>15 (range search) | accident _date | 1/week<br>1/month<br><br>1/day | Indexed Sequential File based on accident_date<br><br>Secondary index on accident_date | The most frequent query is deletion based on date. Since it is not very frequent, secondary index is fast enough. |
| Produce Cause | 6 (Insertion) | | 1/week | Heap file | We have only insertion. |
| Repair Cause | 6 (Insertion)<br><br>13 (random search) | | 1/week<br>1/month | Heap File | Insertion happens more frequently than random search. |
| Request | 10 (random search) | product_ id | 40/day | Indexed Sequential File based on search key product_id<br><br>Primary index on search key product_id | 40 queries per day is quite a lot, but still, it is not too much to use B-Tree. An Indexed Sequential File, and a Primary index is fast enough. |
| Purchase | 3 (Insertion)<br>11 (random search) | | 50/day<br>5/month | Heap File | Insertion is done more frequently than random search. |
| Make | 5 (Insertion)<br>9 (random search) | product_ id | 30/day<br>400/day | Sequential File based on primary key product_id<br><br>B-Tree based on the searcah key product_id | For query 9, we have to find the products that got complain. 400 queries per day is a lot, B-Tree index is a fast file organization to look for a particular product id. |
| Produce | 2 (Insertion)<br>8 (random search) | worker_n ame | 400/day<br>2000/day | Hash File with the hash value worker_name | 2000 query per day is a lot. The fastest way to bring all of the products that is made by one person is to |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | store them in hash files, each bucket for one Worker. |
| Test | 2 (Insertion)<br>9 (random search) | product_<br>id | 400/day<br>400/day | Heap File | Since the frequency of insertion and random search is equal. We will just have a heap file, because sorting takes time. |
| Repair | 2 (Insertion)<br>10 (random search) | | 400/day<br><br>40/day | Heap file | Insertion is done more frequently than random search. |
| RepairP1 | 2 (Insertion) | | 400/day | Heap File | We only have insertion. |
| Got | 13 (random search) | product_<br>id | 1/month | Sequential File based on primary key product_id<br><br>Secondary index based on the product_id | For Query 13, we should look for products that have the same id as the ones in ‘Repair Cause’ and in ‘Accident’. Since one product can have multiple complaints, we will use secondary index on product_id. |

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

Worker: Azure SQL automatically makes a clustered index based on the primary key. So when implantation it will be stored as a clustered index based on the search key *worker_name.*

Quality Controller: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key qual_cont_*name*.

Technical Staff: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key tech_staff_name.

Grad Technical Staff: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key tech_staff_name.

Undergrad Technical Staff: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key tech_staff_name.

Product: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key product_id.

Product1: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key product1_id.

Product2: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key product2_id.

Product3: Azure SQL automatically makes a clustered index based on the primary key. So, during implantation it will be stored as a clustered index based on the search key product3_id.

Customer: Azure SQL automatically makes a clustered index based on the primary key. So, when implantated it will be stored as a clustered index based on the search key customer_name.

Account1: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key account1_num.

Account2: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key account2_num.

Account3: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key account3_num.

Complaint: Azure SQL automatically makes a clustered index based on the primary key. So, when implantation it will be stored as a clustered index based on the search key complaint_id.

Accident: The secondary index in Azure SQL is called nonclustered index. So, we use nonclustered index based on the search key accident_date.

Request: Azure SQL automatically makes a clustered index based on all three primary keys. And since Azure cannot have two clustered indices, we will make a nonclustered index based on the search key product_id.

Purchase: Because Azure makes a clustered index anyways, we leave it as it is by default. A clustered index based on all three primary keys.

Make: Our student subscription for MS Azure does not cover making B-Tree index. Therefore, we will just use a nonclustered index based on the search key product_id. Clustered index will not work because Azure has already made a clustered index based on all three primary keys.

Produce: MS Azure does not have hash tables. Therefore, we use a nonclustered index based on the search key worker_name.

Test: Because Azure makes a clustered index anyways, we leave it as it is by default. A clustered index based on both primary keys.

Repair: Because Azure makes a clustered index anyways, we leave it as it is by default. A clustered index based on both primary keys.

RepairP1: Because Azure makes a clustered index anyways, we leave it as it is by default. A clustered index based on both primary keys.

Got: MS Azure automatically creates a clustered index based on the three primary keys in this table. Therefore, we will have to make a nonclustered index based on the search key product_id.

**Task 4.**

SQL statements and screenshots showing the creation of tables in Azure SQL Database

```sql
--Individual Project
-- While working on the database design, it's useful to start from scratch every time
-- Hence, we drop tables in reverse order they are created (so the foreign key constraints are not violated)

DROP TABLE IF EXISTS Got;
DROP TABLE IF EXISTS Make;
DROP TABLE IF EXISTS Complaint;
DROP TABLE IF EXISTS Purchase;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS RepairCause;
DROP TABLE IF EXISTS ProduceCause;
DROP TABLE IF EXISTS Accident;
DROP TABLE IF EXISTS RepairP1;
DROP TABLE IF EXISTS Product3;
DROP TABLE IF EXISTS Product2;
DROP TABLE IF EXISTS Product1;
DROP TABLE IF EXISTS Account3;
DROP TABLE IF EXISTS Account2;
DROP TABLE IF EXISTS Account1;
DROP TABLE IF EXISTS Request;
DROP TABLE IF EXISTS Repair;
DROP TABLE IF EXISTS Test;
DROP TABLE IF EXISTS Produce;
DROP TABLE IF EXISTS Product;
DROP TABLE IF EXISTS UndergradTechnicalStaff;
DROP TABLE IF EXISTS GradTechnicalStaff;
DROP TABLE IF EXISTS TechnicalStaff;
DROP TABLE IF EXISTS QualityCOntroller;
DROP TABLE IF EXISTS Worker;


-- Create tables

--Worker
CREATE TABLE Worker (
    worker_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100),
    salary REAL,
    max_num_prod_per_day INT
);


--Quality Controller
CREATE TABLE QualityController(
    qual_cont_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100),
    salary REAL,
    product_type VARCHAR(10)
    CONSTRAINT CHK_product_type CHECK (product_type IN ('type1', 'type2',
'type3'))
);

--Technical Staff
CREATE TABLE TechnicalStaff(
    tech_staff_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100),
    salary REAL,
    education_record VARCHAR(3),
    technical_position VARCHAR(30),
    CONSTRAINT CHK_education_record CHECK (education_record IN ('BS', 'MS',
'PhD'))

);

--Grad Technical Staff
CREATE TABLE GradTechnicalStaff(
    tech_staff_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100),
    salary REAL,
    education_record VARCHAR(3),
    technical_position VARCHAR(30),
    CONSTRAINT CHK_grad_education_record CHECK (education_record IN ('MS',
'PhD'))
);

--Undergrad Technical Staff
CREATE TABLE UndergradTechnicalStaff(
    tech_staff_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100),
    salary REAL,
    education_record VARCHAR(2),
    technical_position VARCHAR(30),
    CONSTRAINT CHK_undergrad_education_record CHECK (education_record IN
('BS'))
);

--Account1
CREATE TABLE Account1(
    account1_num INT PRIMARY KEY,
    establish_date DATE NOT NULL,
    product1_cost REAL
);

--Account2
```

```sql
CREATE TABLE Account2(
    account2_num INT PRIMARY KEY,
    establish_date DATE NOT NULL,
    product2_cost REAL
);

--Account3
CREATE TABLE Account3(
    account3_num INT PRIMARY KEY,
    establish_date DATE NOT NULL,
    product3_cost REAL
);

--Product
CREATE TABLE Product(
    product_id INT PRIMARY KEY,
    production_date DATE NOT NULL, -- YYYY-MM-DD
    time_spent REAL NOT NULL, --THis is in hours
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64)
);

--Product1
CREATE TABLE Product1(
    product1_id INT,
    production_date DATE NOT NULL,
    time_spent REAL NOT NULL, --THis is in hours
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64), --It is repaired only by a technical staff
    product1_name VARCHAR(64),
    software_used VARCHAR(64),
    account1_num INT,
    PRIMARY KEY (product1_id, account1_num),
    FOREIGN KEY (account1_num) REFERENCES Account1
);

--Product2
CREATE TABLE Product2(
    product2_id INT,
    production_date DATE NOT NULL,
    time_spent REAL NOT NULL, --THis is in hours
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64),
    size CHAR(1), -- S, M, L
    color VARCHAR(10),
    account2_num INT,
    PRIMARY KEY (product2_id, account2_num),
    FOREIGN KEY (account2_num) REFERENCES Account2,
    CONSTRAINT CHK_product2_size CHECK (size IN ('S', 'M', 'L'))
);

--Product3
CREATE TABLE Product3(
    product3_id INT,
    production_date DATE NOT NULL,
    time_spent REAL NOT NULL, --THis is in hours
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64),
    size CHAR(1), -- S, M, L
    weight REAL, -- In grams
    account3_num INT,
    PRIMARY KEY (product3_id, account3_num),
    FOREIGN KEY (account3_num) REFERENCES Account3,
    CONSTRAINT CHK_product3_size CHECK (size IN ('S', 'M', 'L'))
);

--Produce
CREATE TABLE Produce(
    worker_name VARCHAR(64),
    product_id INT,
    PRIMARY KEY (worker_name, product_id),
    FOREIGN KEY (worker_name) REFERENCES Worker,
    FOREIGN KEY (product_id) REFERENCES Product
);

CREATE NONCLUSTERED INDEX produce_nonclustered ON Produce (worker_name);

--Check
CREATE TABLE Test(
    qual_cont_name VARCHAR(64),
    product_id INT,
    PRIMARY KEY (qual_cont_name, product_id),
    FOREIGN KEY (qual_cont_name) REFERENCES QualityController,
    FOREIGN KEY (product_id) REFERENCES Product
);

--Repair
CREATE TABLE Repair(
    tech_staff_name VARCHAR(64),
    product_id INT,
    repair_date DATE,
    PRIMARY KEY (tech_staff_name, product_id),
    FOREIGN KEY (tech_staff_name) REFERENCES TechnicalStaff,
    FOREIGN KEY (product_id) REFERENCES Product,

);
```

```
--Repair for product1 and graduate technical staff
CREATE TABLE RepairP1(
    tech_staff_name VARCHAR(64),
    product1_id INT,
    account1_num INT,
    repair_date DATE,
    PRIMARY KEY (tech_staff_name, product1_id),
    FOREIGN KEY (tech_staff_name) REFERENCES GradTechnicalStaff,
    FOREIGN KEY (product1_id, account1_num) REFERENCES Product1
);

--Request
CREATE TABLE Request(
    qual_cont_name VARCHAR(64),
    product_id INT,
    tech_staff_name VARCHAR(64),
    PRIMARY KEY (qual_cont_name, product_id, tech_staff_name),
    FOREIGN KEY (qual_cont_name) REFERENCES QualityController,
    FOREIGN KEY (product_id) REFERENCES Product,
    FOREIGN KEY (tech_staff_name) REFERENCES TechnicalStaff
);

CREATE NONCLUSTERED INDEX request_nonclustered ON Request(product_id);

--Accident

CREATE TABLE Accident(
    accident_number INT PRIMARY KEY,
    accident_date DATE NOT NULL,
    num_work_day INT
);

CREATE NONCLUSTERED INDEX accident_clustered ON Accident (accident_date); --
DOUBLE CHECK


--Produce Cause
CREATE TABLE ProduceCause(
    accident_number INT,
    worker_name VARCHAR(64),
    product_id INT,
    PRIMARY KEY(accident_number,worker_name, product_id),
    FOREIGN KEY (accident_number) REFERENCES Accident,
    FOREIGN KEY (worker_name) REFERENCES Worker,
    FOREIGN KEY (product_id) REFERENCES Product,
);

--Repair Cause
CREATE TABLE RepairCause(
```

```sql
    accident_number INT,
    tech_staff_name VARCHAR(64),
    product_id INT,
    PRIMARY KEY(accident_number,tech_staff_name, product_id),
    FOREIGN KEY (accident_number) REFERENCES Accident,
    FOREIGN KEY (tech_staff_name) REFERENCES TechnicalStaff,
    FOREIGN KEY (product_id) REFERENCES Product,
);


--Customer
CREATE TABLE Customer(
    customer_name VARCHAR(64) PRIMARY KEY,
    addr VARCHAR(100)
);

--Purchase
CREATE TABLE Purchase(
    product_id INT,
    customer_name VARCHAR(64),
    PRIMARY KEY (product_id, customer_name),
    FOREIGN KEY (product_id) REFERENCES Product,
    FOREIGN KEY (customer_name) REFERENCES Customer,
);

--Complaint
CREATE TABLE Complaint(
    complaint_id INT PRIMARY KEY,
    complaint_date DATE,
    description VARCHAR(1000),
    treatment VARCHAR(1000),
);

--Make. Make is a table for storing data, when making complains. It shoud not
be mistaken by Produce.
CREATE TABLE Make(
    customer_name VARCHAR(64),
    product_id INT,
    complaint_id INT,
    PRIMARY KEY (customer_name, product_id, complaint_id),
    FOREIGN KEY (customer_name) REFERENCES Customer,
    FOREIGN KEY (product_id) REFERENCES Product,
    FOREIGN KEY (complaint_id) REFERENCES Complaint,
);

-- Nonclusted index because azure has already made a clustered index based
all three primary keys.
CREATE NONCLUSTERED INDEX make_nonclustered ON Make (product_id);

--Got
```

```sql
CREATE TABLE Got(
    complaint_id INT,
    product_id INT,
    tech_staff_name VARCHAR(64),
    PRIMARY KEY (complaint_id, product_id, tech_staff_name),
    FOREIGN KEY (complaint_id) REFERENCES Complaint,
    FOREIGN KEY (product_id) REFERENCES Product,
    FOREIGN KEY (tech_staff_name) REFERENCES TechnicalStaff

);

CREATE NONCLUSTERED INDEX got_nonclustered ON Got (product_id);
```

**Task 5**

***5.1. SQL statements for queries 1-15.***

```
-- Query 1 -----------------------------------------------------

INSERT INTO Worker
    (worker_name, addr, salary, max_num_prod_per_day)
VALUES
    ('Wahid', 'Norman', 3000, 4);
    --These values are just examples. They are not stored in the database

INSERT INTO QualityController
    (qual_cont_name, addr, salary, product_type)
VALUES
    ('Ghulam', 'Norman', 3000, 'type1');
    --These values are just examples. They are not stored in the database

INSERT INTO TechnicalStaff
    (tech_staff_name, addr, salary, education_record)
VALUES
    ('Shohruz', 'Norman', 300, 'BS');
    --These values are just examples. They are not stored in the database

INSERT INTO GradTechnicalStaff
    (tech_staff_name, addr, salary, education_record)
VALUES

    ('Shohruz', 'Norman', 3000, 'MS');
    --These values are just examples. They are not stored in the database

INSERT INTO UndergradTechnicalStaff
    (tech_staff_name, addr, salary, education_record)
VALUES
    ('Shohruz', 'Norman', 3000, 'BS');
    --These values are just examples. They are not stored in the database



--Query 2 -------------------------------------------------------------

INSERT INTO Product
    (product_id, production_date, time_spent, produced_by, tested_by,
repaired_by)
VALUES
    (1, '2018-02-22', 24, 'Wahid', 'Ghulam', 'Shohruz');
    --These values are just examples. They are not stored in the database

INSERT INTO Product1
```

```
    (product1_id, production_date, time_spent, produced_by, tested_by,
repaired_by, product1_name, software_used, account1_num)
VALUES
    (1, '2019-02-22', 23, 'Wahid', 'Ghulam', 'Shohruz', 'book', 'sofware1',
1);
   --These values are just examples. They are not stored in the database

INSERT INTO Product2
    (product2_id, production_date, time_spent, produced_by, tested_by,
repaired_by, size, color, account2_num)
VALUES
    (2, '2019-02-22', 23, 'Wahid', 'Ghulam', 'Shohruz', 'M', 'red', 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Product3
    (product3_id, production_date, time_spent, produced_by, tested_by,
repaired_by, size, weight, account3_num)
VALUES
    (1, '2019-02-22', 23, 'Wahid', 'Ghulam', 'Shohruz', 'm', 15, 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Produce
    (worker_name, product_id)
VALUES
    ('Wahid', 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Test
    (qual_cont_name, product_id)
VALUES
    ('Ghulam', 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Repair
    (tech_staff_name, product_id, repair_date)
VALUES
    ('Shohruz', 1, '2019-09-19');
    --These values are just examples. They are not stored in the database

INSERT INTO RepairP1
    (tech_staff_name, product1_id, account1_num)
VALUES
    ('Shohruz', 1, 1);
    --These values are just examples. They are not stored in the database


INSERT INTO Request
    (qual_cont_name, product_id, tech_staff_name)
VALUES
    ('Ghulam', 1, 'Shohruz');
```

```sql
--Query 3 ---------------------------------------------------------------

INSERT INTO Customer
    (customer_name, addr)
VALUES
    ('Jack', 'NYC');
    --These values are just examples. They are not stored in the database

INSERT INTO Purchase
    (product_id, customer_name)
VALUES
    (1, 'Jack');
    --These values are just examples. They are not stored in the database


--Query 4 ---------------------------------------------------------------

INSERT INTO Account1
    (account1_num, establish_date, product1_cost)
VALUES
    (1, '2022-01-01', 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Account2
    (account2_num, establish_date, product2_cost)
VALUES
    (2, '2022-01-01', 4);
   --These values are just examples. They are not stored in the database


INSERT INTO Account3
    (account3_num, establish_date, product3_cost)
VALUES
    (1, '2022-01-01', 1);
    --These values are just examples. They are not stored in the database


-- Query 5 -------------------------------------------------------------

INSERT INTO Complaint
    (complaint_id, complaint_date, description, treatment)
VALUES
    (1, '2010-10-10', 'book is old', 'refund');
    --These values are just examples. They are not stored in the database


INSERT INTO Make -- Used for when a customer makes complaint
```

```sql
    (customer_name, product_id, complaint_id)
VALUES
    ('Jack', 1, 1);
    --These values are just examples. They are not stored in the database

INSERT INTO Got
    (complaint_id, product_id, tech_staff_name)
VALUES
    (1, 1, 'Shohruz'),
    (2, 2, 'Shohruz');

-- Query 6 ----------------------------------------------------------

INSERT INTO Accident
    (accident_number, accident_date, num_work_day)
VALUES
    (1, '2010-09-20', 3);
    --These values are just examples. They are not stored in the database

INSERT INTO ProduceCause
    (accident_number, worker_name, product_id)
VALUES
    (1, 'Wahid', 1);
    --These values are just examples. They are not stored in the database

INSERT INTO RepairCause
    (accident_number, tech_staff_name, product_id)
VALUES
    (1, 'Shohruz', 1);
    --These values are just examples. They are not stored in the database


-- Query 7 -----------------------------------------------------------

SELECT production_date, time_spent
FROM Product
WHERE product_id = 1

-- Query 8 ----------------------------------------------------------

SELECT Product.*
FROM Produce JOIN Product ON Produce.product_id = Product.product_id
WHERE worker_name = 'Jack';


-- Query 9 ---------------------------------------------------------

SELECT COUNT (Make.product_id) as tot_product
FROM Make
JOIN Test ON Make.product_id = Test.product_id
```

```sql
WHERE qual_cont_name = 'Ali';


-- Query 10 ---------------------------------------------------------

SELECT SUM (product3_cost) AS total_product3_cost
FROM Product3
      JOIN Account3 ON  Product3.account3_num = Account3.account3_num
      JOIN Request ON Request.product_id = Product3.product3_id
      JOIN Repair ON Repair.product_id = Product3.product3_id
      WHERE Request.qual_cont_name = 'John';


-- Query 11 ---------------------------------------------------------

SELECT Customer.customer_name
FROM Customer, Product2, Purchase
WHERE Customer.customer_name = Purchase.customer_name
    AND product_id = Product2.product2_id
    AND Product2.color = 'red'
ORDER BY customer_name;

-- Query 12 --------------------------------------------------------

WITH Employee as (
SELECT salary, addr , worker_name  as employee_name
FROM Worker
UNION
SELECT salary, addr , qual_cont_name as employee_name
FROM QualityController
UNION
SELECT salary, addr, tech_staff_name as employee_name
FROM TechnicalStaff
)
select employee_name, addr as address, salary
FROM Employee Where salary > 100;

-- Query 13 ------------------------------------------------------

SELECT SUM (Accident.num_work_day) as total_number_of_workdays
FROM Accident
JOIN RepairCause ON RepairCause.accident_number = Accident.accident_number
JOIN Got ON Got.product_id = RepairCause.product_id;

-- Query 14 --------------------------------------------------------

--Since there are three different products, and three different accounts,
--we join product and product1, product and product2, product and product3,
--and union all of them.
WITH unionized as (
SELECT product_id, Product.production_date,  product1_cost as cost
FROM Product
    JOIN Product1 ON Product.product_id = Product1.product1_id
    JOIN Account1 ON Product1.account1_num = Account1.account1_num

UNION ALL

SELECT product_id, Product.production_date, product2_cost as cost
FROM Product
    JOIN Product2 ON Product.product_id = Product2.product2_id
    JOIN Account2 ON Product2.account2_num = Account2.account2_num

UNION ALL

SELECT product_id, Product.production_date, product3_cost as cost
FROM Product
    JOIN Product3 ON Product.product_id = Product3.product3_id
    JOIN Account3 ON Product3.account3_num = Account3.account3_num
)
select AVG(cost) FROM unionized
WHERE YEAR(production_date) = 2019;

-- Query 15 -------------------------------------------------------

-- Query 15 -------------------------------------------------------

-- Before deleting from accident, we should delete from ProduceCasue because
-- the data in ProduceCasue refer to Accident
DELETE ProduceCause FROM ProduceCause pc
     RIGHT JOIN Accident ac on ac.accident_number = pc.accident_number
WHERE accident_date >= '2010-01-01'
    AND accident_date <= '2015-01-01';
    --The values here are just examples. They are not run in the database.

-- Before deleting from accident, we should delete from RepairCasue because
-- the data in RepairCasue refer to Accident
DELETE RepairCause FROM RepairCause rc
     RIGHT JOIN Accident ac on ac.accident_number = rc.accident_number
WHERE accident_date >= '2010-01-01'
    AND accident_date <= '2015-01-01';
    --The values here are just examples. They are not run in the database.

DELETE FROM Accident
    WHERE accident_date >= '2010-01-01'
    AND accident_date <= '2015-01-01';
    --The values here are just examples. They are not run in the database.
```

***5.2. Java application program that uses JDBC***

Content of the file MyProducts.java

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.util.Arrays;

 public class MyProducts {

    // Database credentials
    final static String HOSTNAME = "haid0000-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "haid0000";
    final static String PASSWORD = "Changquan2023";


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

    // Query templates

    //Query 1, insert to Worker
    final static String QUERY1_WORKER = "INSERT INTO Worker " +
                  "VALUES (?, ?, ?, ?);";

  //Query 1, insert to QualityController
    final static String QUERY1_QUALITY_CONTROLLER = "INSERT INTO
QualityController " +
                  "VALUES (?, ?, ?, ?);";

  //Query 1, insert to TechnicalStaff
    final static String QUERY1_TECHNICAL_STAFF = "INSERT INTO TechnicalStaff
" +
                  "VALUES (?, ?, ?, ?, ?);";
  //Query 1, insert to Undergrad Technical Staff
    final static String QUERY1_UNDERGRAD_TECHNICAL_STAFF = "INSERT INTO 
UndergradTechnicalStaff " +
                  "VALUES (?, ?, ?, ?, ?);";

    //Query 1, insert to Grad Technical Staff
    final static String QUERY1_GRAD_TECHNICAL_STAFF = "INSERT INTO 
GradTechnicalStaff " +
                  "VALUES (?, ?, ?, ?, ?);";


    //Query 2, insert to Product
    final static String QUERY2_PRODUCT = "INSERT INTO Product " +
                  "VALUES (?, ?, ?, ?, ?, ?);";

    //Query 2, insert to Product type 1
    final static String QUERY2_PRODUCT1 = "INSERT INTO Product1 " +
                  "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";

    //Query 2, insert to Product type 2
   final static String QUERY2_PRODUCT2 = "INSERT INTO Product2 " +
                  "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";

   //Query 2, insert to Product type 3
   final static String QUERY2_PRODUCT3 = "INSERT INTO Product3 " +
                  "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";

   //Query 2, insert to Produce
   final static String QUERY2_PRODUCE = "INSERT INTO Produce " +
                  "VALUES (?, ?);";

   //Query 2, insert to Test
   final static String QUERY2_TEST = "INSERT INTO Test " +
                  "VALUES (?, ?);";
   //Query 2, insert to Repair
   final static String QUERY2_REPAIR = "INSERT INTO Repair " +
                  " (tech_staff_name, product_id) VALUES (?, ?);";

   //Query 2, insert to RepairP1. This table includes all the products of 
type 1,
   // and their associated graduate technical staff who repaired them.
   final static String QUERY2_REPAIR_P1 = "INSERT INTO RepairP1 " +
                  " (tech_staff_name, product1_id, account1_num) VALUES 
(?, ?, ?);";

   //Query 2, insert to Request. When a product is repaired because it
   //is requested by a quality controller.
   final static String QUERY2_REQUEST = "INSERT INTO Request " +
                  " (qual_cont_name, product_id, tech_staff_name) VALUES
(?, ?, ?);";

   //Query 3 insert to customer,
   final static String QUERY3_INSERT_CUSTOMER = "INSERT INTO Customer " +
                  " VALUES (?, ?);";

   //Query 3 insert into purchase,
   final static String QUERY3_INSERT_PURCHASE = "INSERT INTO Purchase " +
                  " VALUES (?, ?);";

   //Query 4, Insert into account1
   final static String QUERY4_ACCOUNT1 = "INSERT INTO Account1 " +
               "VALUES (?, ?, ?);";

   //Query 4, Insert into account2
   final static String QUERY4_ACCOUNT2 = "INSERT INTO Account2 " +
               "VALUES (?, ?, ?);";

   //Query 4, Insert into account3
   final static String QUERY4_ACCOUNT3 = "INSERT INTO Account3 " +
               "VALUES (?, ?, ?);";

   //Query 5, Insert into Complaint
   final static String QUERY5_INSERT_COMPLAINT= "INSERT INTO Complaint " +
               "VALUES (?, ?, ?, ?);";

   //Query 5, Insert into Make
   final static String QUERY5_INSERT_MAKE= "INSERT INTO Make " +
               "VALUES (?, ?, ?);";

   //Query 5, Insert into Got
   final static String QUERY5_INSERT_GOT = "INSERT INTO Got " +
               "VALUES (?, ?,? );";

   //Query 6, Insert into Accident
   final static String QUERY6_INSERT_ACCIDENT = "INSERT INTO Accident " +
               "VALUES (?, ?, ?);";

   //Query 6, Insert into ProduceCause
   final static String QUERY6_INSERT_PRODUCE_CAUSE = "INSERT INTO
ProduceCause " +
               "VALUES (?, ?, ?);";

   //Query 6, Insert into RepairCause
   final static String QUERY6_INSERT_REPAIR_CAUSE = "INSERT INTO RepairCause
" +
               "VALUES (?, ?, ?);";

   //Query 7
   final static String QUERY7 = "SELECT production_date , time_spent FROM 
Product " +
               "WHERE product_id = ?;";

   //Query 8
   final static String QUERY8 = "SELECT Product.* FROM Produce JOIN Product " 
+
               "ON Produce.product_id = Product.product_id " +
               "WHERE worker_name = ?;";

   //Query 9
   final static String QUERY9 = "SELECT COUNT (Make.product_id) FROM Make 
JOIN Test " +
               "ON Make.product_id = Test.product_id " +
               "WHERE qual_cont_name = ?;";

   //Query 10
   final static String QUERY10 = "SELECT SUM (product3_cost) AS 
total_product3_cost " +
               "FROM Product3 " +
               "JOIN Account3 ON  Product3.account3_num = 
Account3.account3_num " +
               "JOIN Request ON Request.product_id = Product3.product3_id " +
               "JOIN Repair ON Repair.product_id = Product3.product3_id " +
               "WHERE Request.qual_cont_name = ?;";

   //Query 11
   final static String QUERY11 = "SELECT Customer.customer_name " +
               "FROM Customer, Product2, Purchase " +
               "WHERE Customer.customer_name = Purchase.customer_name " +
               "AND product_id = Product2.product2_id " +
               "AND Product2.color = ? " +
               "ORDER BY customer_name;";

   //Query 12
   final static String QUERY12 = "WITH Employee as ( " +
               "SELECT salary,addr, worker_name as employee_name FROM Worker 
" +
               "UNION " +
               "SELECT salary, addr, qual_cont_name as employee_name FROM 
QualityController " +
               "UNION " +
               "SELECT salary, addr, tech_staff_name as employee_name " +
               "FROM TechnicalStaff ) " +
               "SELECT employee_name, addr AS address, salary " +
               "FROM Employee Where salary > ?;";

   //Query 13
   final static String QUERY13 = "SELECT SUM (Accident.num_work_day) " +
               "FROM Accident " +
               "JOIN RepairCause " +
               "ON RepairCause.accident_number = Accident.accident_number " +
               "JOIN Got ON Got.product_id = RepairCause.product_id;";

   //Query 14
   final static String QUERY14 = "WITH unionized as ( " +
               "SELECT product_id, Product.production_date,  product1_cost as
cost " +
               "FROM Product " +
               "JOIN Product1 ON Product.product_id = Product1.product1_id "
+
               "JOIN Account1 ON Product1.account1_num =
Account1.account1_num " +
               "UNION " +
               "SELECT product_id, Product.production_date, product2_cost as
cost " +
               "FROM Product " +
               "JOIN Product2 ON Product.product_id = Product2.product2_id "
+
               "JOIN Account2 ON Product2.account2_num =
Account2.account2_num " +
               "UNION " +
               "SELECT product_id, Product.production_date, product3_cost as
cost " +
               "FROM Product " +
               "JOIN Product3 ON Product.product_id = Product3.product3_id "
+
               "JOIN Account3 ON Product3.account3_num =
Account3.account3_num " +
               ") " +
               "SELECT AVG(cost) FROM unionized "+
               "WHERE YEAR(production_date) = ?";

   //Query 15

   final static String QUERY15_PRODUCE_CAUSE = "DELETE ProduceCause FROM
ProduceCause pc " +
               "RIGHT JOIN Accident ac on ac.accident_number =
pc.accident_number " +
               "WHERE accident_date >= ? AND accident_date <= ?";

   final static String QUERY15_REPAIR_CAUSE = "DELETE RepairCause FROM
RepairCause rc " +
               "RIGHT JOIN Accident ac on ac.accident_number =
rc.accident_number " +
               "WHERE accident_date >= ? AND accident_date <= ?";

   final static String QUERY15_ACCIDENT = "DELETE FROM Accident " +
               "WHERE accident_date >= ? AND accident_date <= ?";
```

```
    //Query 15. Print Accident after deletion
    final static String QUERY15_PRINT = "SELECT * FROM Accident;";

     // User input prompt//
     final static String PROMPT =
             "\nPlease select one of the options below: \n" +
             "1) Enter a new employee. \n" +
             "2) Enter a new product associated with the person who made the
product, \n" +
             "repaired the product if it is repaired, or checked the product
\n" +
             "3) Enter a customer associated with some products.\n" +
             "4) Create a new account associated with a product.\n" +
             "5) Enter a complaint associated with a customer and product.\n"
+
             "6) Enter an accident associated with an appropriate employee and
product.\n" +
             "7) Retrieve the date produced and time spent to produce a
particular product.\n" +
             "8) Retrieve all products made by a particular worker.\n" +
             "9) Retrieve the total number of errors a particular quality
controller made.\n" +
             "This is the total number of products certified by this
controller and got some complaints.\n" +
             "10) Retrieve the total costs  of the  products  in the product3
category which \n" +
             "were repaired at the request of a particular quality controller.
\n" +
             "11) Retrieve  all  customers  (in  name  order)  who  purchased
all  products  of  a  particular  color.\n" +
             "12) Retrieve all employees whose salary is above a particular
salary.\n" +
             "13) Retrieve the total number of workdays lost due to accidents
in \n" +
                         "repairing the products which got complaints.\n" +
             "14) Retrieve the average cost of all products made in a
particular year.\n" +
             "15) Delete all accidents whose dates are in some range.\n" +
             "16) Import\n" +
             "17) Export\n" +
             "18) Quit\n";

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

       //print the the welcome message
         System.out.println("WELCOME TO THE DATABASE SYSTEM OF MyProducts,
Inc.");
```

```
        final Scanner sc = new Scanner(System.in); // Scanner is used to
collect the user input
        String option = ""; // Initialize user option selection as nothing
        while (!option.equals("18")) { // As user for options until option 18
is selected
            System.out.println(PROMPT); // Print the available options
            option = sc.next(); // Read in the user option selection

            switch (option) { // Switch between different options
                case "1": // Insert a new employee
                    option1(sc);
                    break;
                case "2": // Insert a new product
                  option2(sc);
                  break;
                case "3":
                  option3(sc);
                  break;
                case "4":
                  option4(sc);
                  break;
                case "5":
                  option5(sc);
                  break;
                case "6":
                  option6(sc);
                  break;
                case "7":
                  option7(sc);
                  break;
                case "8":
                  option8(sc);
                  break;
                case "9":
                  option9(sc);
                  break;
                case "10":
                  option10(sc);
                  break;
                case "11":
                  option11(sc);
                  break;
                case "12":
                  option12(sc);
                  break;
                case "13":
                  option13();
                  break;
                case "14":
                  option14(sc);
                  break;
                case "15":
                  option15(sc);
                  break;
                case "16":  //Import
                  option16(sc);
                  break;
                case "17": //Export
                  option17(sc);
                  break;
                case "18": // Do nothing, the while loop will terminate upon
the next iteration
                    System.out.println("Exiting! Good-buy!");
                    break;
                default: // Unrecognized option, re-prompt the user for the
correct one
                    System.out.println(String.format(
                        "Unrecognized option: %s\n" +
                        "Please try again!",
                        option));
                    break;
            }
        }

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

    //Option1 ---------------------------------------------------------------
-----------------
    public static void option1(Scanner sc) throws SQLException {
      // Since we have three different tables for the 3 types of employees,
we ask the user to pick what kind
      //of employee they want to insert.
        System.out.println("Do you want enter a worker, a quality controller
or a technical staff?\n"
                  + "Enter w for worker.\n"
                  + "Enter q for quality controller.\n"
                  + "Enter t for technical staff.\n"
                  );
        char employeeOption = sc.next().charAt(0);
        switch (employeeOption) {
            case 'w': //For worker
                  insertWorker(sc);
            break;
            case 'q': //For quality controller
                  insertQualityController(sc);
            break;
            case 't': // For technical staff
                  insertTechnicalStaff(sc);
```

```
            break;
            default:
                  System.out.println("unrecognizable option");

        }
    }

    //If the user wants to insert a worker
    public static void insertWorker(Scanner sc) throws SQLException {
      sc.nextLine();// Read next line if to consume any any newline
characters if left from other scannnings.
      System.out.println("Enter the name of the Worker.\n");
            String workerName = sc.nextLine();

            System.out.println("Enter the address of the Worker in one
line.\n");
            String workerAddress = sc.nextLine();
            System.out.println("Enter the salary of the Worker in
dollars.\n");
            double workerSalary = sc.nextDouble();
            System.out.println("Enter the maximum number of product a worker
produces per day.\n");
            int maxNamProdPerDay = sc.nextInt();

             System.out.println("Connecting to the database...");
         // Get a database connection and prepare a query statement
         try (final Connection connection = DriverManager.getConnection(URL))
{
             try (
                 final PreparedStatement statement =
connection.prepareStatement(QUERY1_WORKER)) {
                 // Populate the query template with the data collected from
the user
                 statement.setString(1, workerName);
                 statement.setString(2, workerAddress);
                 statement.setDouble(3, workerSalary);
                 statement.setInt(4, maxNamProdPerDay);

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

    //When the user inserts a new quality controller
    public static void insertQualityController(Scanner sc) throws
SQLException {
      sc.nextLine();
      System.out.println("Enter the name of the quality controller.\n");
            String qualContName = sc.nextLine();
            System.out.println("Enter the address of the quality controller
in one line.\n");
            String qualContAddress = sc.nextLine();
            System.out.println("Enter the salary of the quality
controller.\n");
            double qualContSalary = sc.nextDouble();
            System.out.println("Enter the product type the quality controller
tests.\n" +
                        "enter type1 for type 1\n" +
                        "enter type1 for type 2\n" +
                        "enter type1 for type 3\n"
                        );
            String productType = sc.next();

             System.out.println("Connecting to the database...");
         // Get a database connection and prepare a query statement
         try (final Connection connection = DriverManager.getConnection(URL))
{
             try (final PreparedStatement statement =
connection.prepareStatement(QUERY1_QUALITY_CONTROLLER)) {
                 // Populate the query template with the data collected from
the user
                 statement.setString(1, qualContName);
                 statement.setString(2, qualContAddress);
                 statement.setDouble(3, qualContSalary);
                 statement.setString(4, productType);


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

    //When the user inserts a new technical staff
    public static void insertTechnicalStaff(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the name of the technical staff.\n");
            String techStaffName = sc.nextLine();
            System.out.println("Enter the address of the technical staff in
one line.\n");
            String techStaffAddress = sc.nextLine();
            System.out.println("Enter the salary of the technical staff.\n");
            double techStaffSalary = sc.nextDouble();
            sc.nextLine();
            System.out.println("Enter the education record for the technical 
staff.\n");
            String educationRecord = sc.nextLine();
            System.out.println("Enter the technical position of the technical 
staff.\n");
            String technicalPosition = sc.nextLine();

             System.out.println("Connecting to the database...");
         // Get a database connection and prepare a query statement
         try (final Connection connection = DriverManager.getConnection(URL)) 
{
             try (final PreparedStatement statement = 
connection.prepareStatement(QUERY1_TECHNICAL_STAFF)) {
                 // Populate the query template with the data collected from 
the user
                 statement.setString(1, techStaffName);
                 statement.setString(2, techStaffAddress);
                 statement.setDouble(3, techStaffSalary);
                 statement.setString(4, educationRecord);
                 statement.setString(5, technicalPosition);

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


                 //If the technical staff has a BS degree, add it also to the 
table designated for the undergraduate technical staff
                 if (educationRecord.equals("BS")) {

                    try (final PreparedStatement statementForUndergrad = 
connection.prepareStatement(QUERY1_UNDERGRAD_TECHNICAL_STAFF)) {
                             // Populate the query template with the data 
collected from the user
                              statementForUndergrad.setString(1, 
techStaffName);
                              statementForUndergrad.setString(2, 
techStaffAddress);
                              statementForUndergrad.setDouble(3, 
techStaffSalary);
                              statementForUndergrad.setString(4, 
educationRecord);
                              statementForUndergrad.setString(5, 
technicalPosition);

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


                  //If the technical staff is not a BS, they are either MS or
PhD. So in addition to the technical staff table
                  // Put them in their designated table, 'Grad Tehcnical
Staff' too.
                  else {

                    try (final PreparedStatement statementForGrad =
connection.prepareStatement(QUERY1_GRAD_TECHNICAL_STAFF)) {
                             // Populate the query template with the data
collected from the user
                              statementForGrad.setString(1, techStaffName);
                             statementForGrad.setString(2,  techStaffAddress);
                              statementForGrad.setDouble(3, techStaffSalary);
                              statementForGrad.setString(4, educationRecord);
                              statementForGrad.setString(5,
technicalPosition);

                             // Actually execute the populated query
                             final int rows_inserted_grad =
statementForGrad.executeUpdate();
                             System.out.println(String.format("Done. %d rows
inserted.", rows_inserted_grad ));
                    } //Close try CHANGE, add small case also
                  }
              }//End of Second try
          }
    }

    //Option 2 ---------------------------------------------------------------
---------------------------------

    //Since their are 3 types of products, ask the user about the type of
product they want to insert.
    public static void option2(Scanner sc) throws SQLException {
      System.out.println("Which type of product you want to enter?\n"
                  + "Enter 1 for type1\n"
                  + "Enter 2 for type2\n"
                  + "Enter 3 for type3\n"
                  );

      int productTypeOption = sc.nextInt();
      System.out.print("Enter the product id\n");
```

```
            int productID = sc.nextInt();
            System.out.print("Enter the production date in the format YYYY-
MM-DD\n");
            String productionDate = sc.next();
            System.out.println("Enter the number of hours spent on the
product.\n");
            double timeSpent = sc.nextDouble();

            System.out.println("Enter the name of the employee who produced
the product.\n");
            sc.nextLine();
            String producedBy = sc.nextLine();
            System.out.println("Enter the name of the employee who tested the
product.\n");
            String testedBy = sc.nextLine();
            System.out.println("Is this product repaired?\n"
                        + "Enter y for yes.\n"
                        + "Enter n for no.");
            String isRepaired = sc.nextLine();

            String repairedBy ="";
            if(isRepaired.equals("y")){
                  System.out.println("Enter the name of the employee who
repaired the product.\n");
                  repairedBy = sc.nextLine();
            }


            //Insert any product, in the Product table
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                  try (final PreparedStatement statement =
connection.prepareStatement(QUERY2_PRODUCT)) {
                // Populate the query template with the data collected from
the user
                statement.setInt(1, productID);
                statement.setString(2, productionDate);
                statement.setDouble(3, timeSpent);
                statement.setString(4, producedBy);
                statement.setString(5, testedBy);
                statement.setString(6, repairedBy);

                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
             }
        }

            //Now in addition to adding the product in the Product table,
            //add them to the tables designated for their types also.
      switch(productTypeOption) {
            case 1: //If the type of the product a user wants to enter is
type 1.
                 insertProductType1(sc, productID, productionDate, timeSpent,
producedBy, testedBy, repairedBy, isRepaired);
                 break;
            case 2: //If the type of the product a user wants to enter is
type 2.
                 insertProductType2(sc, productID, productionDate, timeSpent,
producedBy, testedBy, repairedBy);
                  break;
            case 3: //If the type of the product a user wants to enter is
type 3.
                 insertProductType3(sc, productID, productionDate, timeSpent,
producedBy, testedBy, repairedBy);
      }

        //When a new product is added, a new entry is added to Produce table
and this will check if
      //the employee who produced it exists. If they do not exist, the sql
will give error.
         try (final Connection connection = DriverManager.getConnection(URL))
{
                 try ( final PreparedStatement statement =
connection.prepareStatement(QUERY2_PRODUCE)) {
                         // Populate the query template with the data
collected from the user
                     statement.setString(1, producedBy);
                     statement.setInt(2, productID);

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

         //When a new product is added, a new entry is added to Test table
and this will check if
      //the employee who tested it exists. If they do not exist, the sql will
give error.
         try (final Connection connection = DriverManager.getConnection(URL))
{
                 try ( final PreparedStatement statement =
connection.prepareStatement(QUERY2_TEST)) {
                         // Populate the query template with the data
collected from the user
                     statement.setString(1, testedBy);
                     statement.setInt(2, productID);
                     System.out.println("Dispatching the query...");
                     // Actually execute the populated query
                     final int rows_inserted = statement.executeUpdate();
                     System.out.println(String.format("Done. %d rows
inserted.", rows_inserted));
                 }
         }

         //When a new product is added, a new entry is added to Repair
Relation and this will check if
      //the employee who repaired it exists. If they do not exist, the sql
will give error.

         if(isRepaired.equals("y")) {
             try (final Connection connection =
DriverManager.getConnection(URL)) {
                 try ( final PreparedStatement statement =
connection.prepareStatement(QUERY2_REPAIR)) {
                         // Populate the query template with the data
collected from the user
                     statement.setString(1, repairedBy);
                     statement.setInt(2, productID);

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

             try (final Connection connection =
DriverManager.getConnection(URL)) {
                 try ( final PreparedStatement statement =
connection.prepareStatement(QUERY2_REQUEST)) {
                         // Populate the query template with the data
collected from the user
                     statement.setString(1, testedBy);
                     statement.setInt(2, productID);
                     statement.setString(3, repairedBy);


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

```
    }

    //---------------------------------------------------------------------
------------------------------------
    //This method is called to inserts products of type1 into Product1 table.
    public static void insertProductType1(Scanner sc, int productID, String
productionDate,
            double timeSpent, String producedBy, String testedBy, String
repairedBy, String isRepaired) throws SQLException {
      System.out.println("Enter the name of the product.\n");
            String product1Name = sc.nextLine();
            System.out.println("Enter the sofware used for this product.\n");
            String software = sc.nextLine();
            System.out.print("Enter the product's account number\n");
            int account1Num = sc.nextInt();

             System.out.println("Connecting to the database...");
         // Get a database connection and prepare a query statement
         try (final Connection connection = DriverManager.getConnection(URL))
{
             try (
                 final PreparedStatement statement =
connection.prepareStatement(QUERY2_PRODUCT1)) {
                 // Populate the query template with the data collected from
the user
                 statement.setInt(1, productID);
                 statement.setString(2, productionDate);
                 statement.setDouble(3, timeSpent);
                 statement.setString(4, producedBy);
                 statement.setString(5, testedBy);
                 statement.setString(6, repairedBy);
                 statement.setString(7, product1Name);
                 statement.setString(8, software);
                 statement.setInt(9, account1Num);


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

         /*After we insert the product of type1, we also add entries to the
table ReparеP1 which
         is a relation between product type 1 and Grad Technical Staff. This
will check if the
```

```
         technical staff who repairs the product is a graduate technical
staff. If it is not the SQL
         will give an error message. */
         if(isRepaired.equals("y")) {

             try (final Connection connection =
DriverManager.getConnection(URL)) {
                 try ( final PreparedStatement statement =
connection.prepareStatement(QUERY2_REPAIR_P1)) {
                         // Populate the query template with the data
collected from the user
                      statement.setString(1, repairedBy);
                      statement.setInt(2, productID);
                      statement.setInt(3, account1Num);

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

    }

    //-----------------------------------------------------------------------
---------------------------------
  //This method is called to inserts products of type2 into Product1 table.
    public static void insertProductType2(Scanner sc, int productID, String
productionDate,
            double timeSpent, String producedBy, String testedBy, String
repairedBy) throws SQLException {
      System.out.println("Enter the size of the product. S, M, or L\n");
            String product2Size = sc.nextLine();
            System.out.println("Enter the color of the product.\n");
            String color = sc.nextLine();
            System.out.print("Enter the product's account number\n");
            int account2Num = sc.nextInt();

             System.out.println("Connecting to the database...");


         // Get a database connection and prepare a query statement


               try (final Connection connection =
DriverManager.getConnection(URL)) {
                   try (
                       final PreparedStatement statement =
connection.prepareStatement(QUERY2_PRODUCT2)) {
                       // Populate the query template with the data collected
from the user
                       statement.setInt(1, productID);
                       statement.setString(2, productionDate);
                       statement.setDouble(3, timeSpent);
                       statement.setString(4, producedBy);
                       statement.setString(5, testedBy);
                       statement.setString(6, repairedBy);
                       statement.setString(7, product2Size);
                       statement.setString(8, color);
                       statement.setInt(9, account2Num);


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

    }

    //-----------------------------------------------------------------------
--------------------------
  //This method is called to inserts products of type3 into Product1 table.
    public static void insertProductType3(Scanner sc, int productID, String
productionDate,
            double timeSpent, String producedBy, String testedBy, String
repairedBy) throws SQLException {
      System.out.println("Enter the size of the product. S, M, or L\n");
            String product3Size = sc.next();
           System.out.println("Enter the weight of the product in grmas.\n");
            double weight = sc.nextDouble();
            System.out.print("Enter the product's account number\n");
            int account3Num = sc.nextInt();

             System.out.println("Connecting to the database...");


         // Get a database connection and prepare a query statement
               try (final Connection connection =
DriverManager.getConnection(URL)) {
                   try (final PreparedStatement statement =
connection.prepareStatement(QUERY2_PRODUCT3)) {
                       // Populate the query template with the data collected
from the user
                       statement.setInt(1, productID);
```

```
                        statement.setString(2, productionDate);
                        statement.setDouble(3, timeSpent);
                        statement.setString(4, producedBy);
                        statement.setString(5, testedBy);
                        statement.setString(6, repairedBy);
                        statement.setString(7, product3Size);
                        statement.setDouble(8, weight);
                        statement.setInt(9, account3Num);

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

    }

  //Option 3 -----------------------------------------------------------------
----------------
    public static void option3(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the customer name\n");
      String customerName = sc.nextLine();
      System.out.println("Enter the address of the customer in one line.\n");
      String customerAddress = sc.nextLine();
      System.out.println("Enter the product id of that the customer
bought.\n");
      int customerBuyProductID= sc.nextInt();

       try (final Connection connection = DriverManager.getConnection(URL)) {
             try (
                 final PreparedStatement statement =
connection.prepareStatement(QUERY3_INSERT_CUSTOMER)) {
                 // Populate the query template with the data collected from
the user
                 statement.setString(1, customerName);
                 statement.setString(2, customerAddress);
                 System.out.println("Dispatching the query...");
                 // Actually execute the populated query
                 final int rows_inserted = statement.executeUpdate();
                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
             }
         }

       try (final Connection connection = DriverManager.getConnection(URL)) {
             try (
                 final PreparedStatement statement =
connection.prepareStatement(QUERY3_INSERT_PURCHASE)) {
                 // Populate the query template with the data collected from
the user
                 statement.setInt(1, customerBuyProductID);
                 statement.setString(2, customerName);
                 System.out.println("Dispatching the query...");
                 // Actually execute the populated query
                 final int rows_inserted = statement.executeUpdate();
                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
             }
         }
    }

    //Option 4 --------------------------------------------------------------
------------------

    //We have 3 kinds of accounts each for a product type.
    //Therefore, we ask the user the type of account they want to enter.
    public static void option4(Scanner sc) throws SQLException {
      System.out.println("What type of Account do you want to enter?\n"
                   + "enter 1, for type 1\n"
                   + "enter 2, for type 2\n"
                   + "enter 3, for type 3");
      int accountTypeOption = sc.nextInt();
      System.out.println("Enter the account number?\n");
      int accounttNum = sc.nextInt();
      System.out.println("Enter the establish date?\n");
      String establishDate = sc.next();
      System.out.println("Enter the associated product cost?\n");
      double productCost = sc.nextDouble();

            //If the user wants to enter an account for type 1 product.
            if (accountTypeOption == 1){
                  try (final Connection connection =
DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement =
connection.prepareStatement(QUERY4_ACCOUNT1)) {
                        // Populate the query template with the data
collected from the user
                        statement.setInt(1, accounttNum);
                        statement.setString(2, establishDate);
                        statement.setDouble(3, productCost);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows
inserted.", rows_inserted));
```

```
                    }
                }
            }

            //If the user wants to enter an account for type 2 product.
            if (accountTypeOption == 2){
                  try (final Connection connection =
DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement =
connection.prepareStatement(QUERY4_ACCOUNT2)) {
                        // Populate the query template with the data
collected from the user
                        statement.setInt(1, accounttNum);
                        statement.setString(2, establishDate);
                        statement.setDouble(3, productCost);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows
inserted.", rows_inserted));
                    }
                }
            }

            //If the user wants to enter an account for type 2 product.
            if (accountTypeOption == 3){
                  try (final Connection connection =
DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement =
connection.prepareStatement(QUERY4_ACCOUNT3)) {
                        // Populate the query template with the data
collected from the user
                        statement.setInt(1, accounttNum);
                        statement.setString(2, establishDate);
                        statement.setDouble(3, productCost);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows
inserted.", rows_inserted));
                    }
                }
            }
        }

        //Option 5 --------------------------------------------------------------
-----------------------------
        public static void option5(Scanner sc) throws SQLException {
```

```
      sc.nextLine();
      System.out.println("Enter the complaint id.\n");
      int complaintID = sc.nextInt();
      System.out.println("Enter the complaint date.\n");
      String complaintDate = sc.next();
     System.out.println("Enter the description of the issue in one line.\n");
      sc.nextLine();
      String description = sc.nextLine();
      System.out.println("What is the treatment?\n");
      String treatment = sc.nextLine();
      System.out.println("Enter the customer's name.\n");
      String complainCustomerName = sc.nextLine();
      System.out.println("Enter the product ID.\n");
      int complainProductID = sc.nextInt();
      sc.nextLine();
      System.out.println("Enter the name of the technical staff who repaired
the product.\n");
      String complainTechStaffName = sc.nextLine();


      //Insert data into the Complaint Table
      try (final Connection connection = DriverManager.getConnection(URL)) {
            try ( final PreparedStatement statement =
connection.prepareStatement(QUERY5_INSERT_COMPLAINT)) {
                        // Populate the query template with the data collected
from the user
                  statement.setInt(1, complaintID);
                  statement.setString(2, complaintDate);
                  statement.setString(3, description);
                  statement.setString(4, treatment);

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

      //Insert data into the Make table
      try (final Connection connection = DriverManager.getConnection(URL)) {
            try ( final PreparedStatement statement =
connection.prepareStatement(QUERY5_INSERT_MAKE)) {
                        // Populate the query template with the data collected
from the user
                  statement.setString(1, complainCustomerName);
                  statement.setInt(2, complainProductID);
                  statement.setInt(3, complaintID);

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

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

      //Insert data into the table Got. This table is used when a product
      //is repaired by a technical staff because it got a complaint.
      try (final Connection connection = DriverManager.getConnection(URL)) {
            try ( final PreparedStatement statement =
connection.prepareStatement(QUERY5_INSERT_GOT)) {
                    // Populate the query template with the data collected
from the user
                  statement.setInt(1, complaintID);
                  statement.setInt(2, complainProductID);
                statement.setString(3, complainTechStaffName);



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

    //Option 6-----------------------------------------------------------
------------------------
    public static void option6(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the accident number");
      int accidentNum = sc.nextInt();
      System.out.println("Enter the accident date");
      String accidentDate = sc.next();
      System.out.println("Enter the number of work days lost due to the
accident.\n");
      int numDaysLost = sc.nextInt();
      System.out.println("Enter the product id that had the accident.\n");
      int accidentProductID = sc.nextInt();

      //There are two causes for the accident. We ask the user to enter the
cause.
      System.out.println("Is this accident caused during production or during
repair?\n" +

                  "Enter p for production\n" +
                  "Enter r for repair");
      char accidentCauseOption = sc.next().charAt(0);

      //Regardless of the cuase of the accident, we enter entry to the
accident table.
      try (final Connection connection = DriverManager.getConnection(URL)) {
            try ( final PreparedStatement statement =
connection.prepareStatement(QUERY6_INSERT_ACCIDENT)) {
                  // Populate the query template with the data collected from
the user
                  statement.setInt(1, accidentNum);
                  statement.setString(2, accidentDate);
                  statement.setInt(3, numDaysLost);


                  // Actually execute the populated query
                  final int rows_inserted = statement.executeUpdate();

            }
      }


      switch(accidentCauseOption) {
            case 'p': //If the accident is caused during the production
                  sc.nextLine();
                  System.out.println("Enter the name of the worker.");
                  String accidentWorkerName = sc.nextLine();

                  try (final Connection connection =
DriverManager.getConnection(URL)) {
                      try ( final PreparedStatement statement =
connection.prepareStatement(QUERY6_INSERT_PRODUCE_CAUSE)) {
                        // Populate the query template with the data
collected from the user
                        statement.setInt(1, accidentNum);
                        statement.setString(2, accidentWorkerName);
                        statement.setInt(3, accidentProductID);

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

            case 'r': //If the accident is caused during the repair.
                  sc.nextLine();
            System.out.println("Enter the name of the technical staff.");
            String accidentTechStaffName = sc.nextLine();
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                try ( final PreparedStatement statement =
connection.prepareStatement(QUERY6_INSERT_REPAIR_CAUSE)) {
                    // Populate the query template with the data collected
from the user
                    statement.setInt(1, accidentNum);
                    statement.setString(2, accidentTechStaffName);
                    statement.setInt(3, accidentProductID);

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

      }
    }

    //Option 7 --------------------------------------------------------------
---------------------------------------
    public static void option7(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the product id");

      int q7ProductID = sc.nextInt();
      System.out.println("Connecting to the database...");
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY7)){
                  statement.setInt(1, q7ProductID);
                  System.out.println("Here is the result:");
                  System.out.println("Production Date | Time Spent ");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s | %s ",
                        resultSet.getString(1),
                        resultSet.getString(2)));
                }
            }
      }

    }
    //Option 8 ---------------------------------------------------------------
----------------------------------------
    public static void option8(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the worker's name.");

      String q8workerName = sc.nextLine();
      System.out.println("Connecting to the database...");
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY8)){
                  statement.setString(1, q8workerName);
                  System.out.println("Here is the result:");
                  System.out.println("Product ID | Production Date | Time
Spent | Produced By |" +
                              " Tested By | Repaired By");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s | %s | %s | %s | %s
| %s ",
                              resultSet.getString(1),
                            resultSet.getString(2),
                            resultSet.getString(3),
                            resultSet.getString(4),
                            resultSet.getString(5),
                            resultSet.getString(6)));
                }
            }
      }
    }

    //Option 9 ---------------------------------------------------------------
------------------
    public static void option9(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the quality controller's name who made the
mistake.");

      String q9QualContName = sc.nextLine();
      System.out.println("Connecting to the database...");
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY9)){
                  statement.setString(1, q9QualContName);
                  System.out.println("Here is the result:");
                  System.out.println("Total Number of Products:");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s ",

                              resultSet.getString(1)));
                }
            }
      }
    }

    //Option 10 -------------------------------------------------------------
-------------------
    public static void option10(Scanner sc) throws SQLException {
      sc.nextLine();
      System.out.println("Enter the quality controller's name who requested
the repair.");

      String q10QualContName = sc.nextLine();
      System.out.println("Connecting to the database...");
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY10)){
                  statement.setString(1, q10QualContName);
                  System.out.println("Here is the result:");
                  System.out.println("Total Cost:");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s ",

                              resultSet.getString(1)));
                }
            }
      }
    }

    //Option 11 -------------------------------------------------------------
------------------
    public static void option11(Scanner sc) throws SQLException {
      System.out.println("Enter the color.");
      String q11Color = sc.next();
      System.out.println("Connecting to the database...");

      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY11)){
                  statement.setString(1, q11Color);
                  System.out.println("Here is the result:");
                  System.out.println("Customer Names:");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s ",

                              resultSet.getString(1)));
                }
            }
      }
    }

    //Option 12 -------------------------------------------------------------
-------------------
    public static void option12(Scanner sc) throws SQLException {
      System.out.println("Enter the salary.");
      double q12Salary = sc.nextDouble();
      System.out.println("Connecting to the database...");

      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY12)){
                  statement.setDouble(1, q12Salary);

                  System.out.println("Here is the result:");
                  System.out.println("Name | Address | Salary ");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s | %s | %s ",

                              resultSet.getString(1),
                              resultSet.getString(2),
                              resultSet.getString(3)));
                }
            }
      }
    }

    //Option 13 -------------------------------------------------------------
-------------------
    public static void option13() throws SQLException {
      System.out.println("Connecting to the database...");
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (
                  final Statement statement = connection.createStatement();
                 final ResultSet resultSet = statement.executeQuery(QUERY13))
{

                  System.out.println("Here is the result:");
                        System.out.print("Total number of workdays lost: ");
                        // Unpack the tuples returned by the database and
print them out to the user
                    while (resultSet.next()) {
                        System.out.println(String.format("%s ",

                                    resultSet.getString(1)));
                    }
                  }
      }
    }

    //Option 14 -------------------------------------------------------------
-------------------
    public static void option14(Scanner sc) throws SQLException {
      System.out.println("Enter the production year.");
      double productionYear14 = sc.nextDouble();
      System.out.println("Connecting to the database...");

      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY14)){
                  statement.setDouble(1, productionYear14);

                  System.out.println("Here is the result:");
                  System.out.println("The average cost:");
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                    System.out.println(String.format("%s ",

                              resultSet.getString(1)));
                }
            }
      }
    }

    //Option 15 -------------------------------------------------------------
-------------------
```

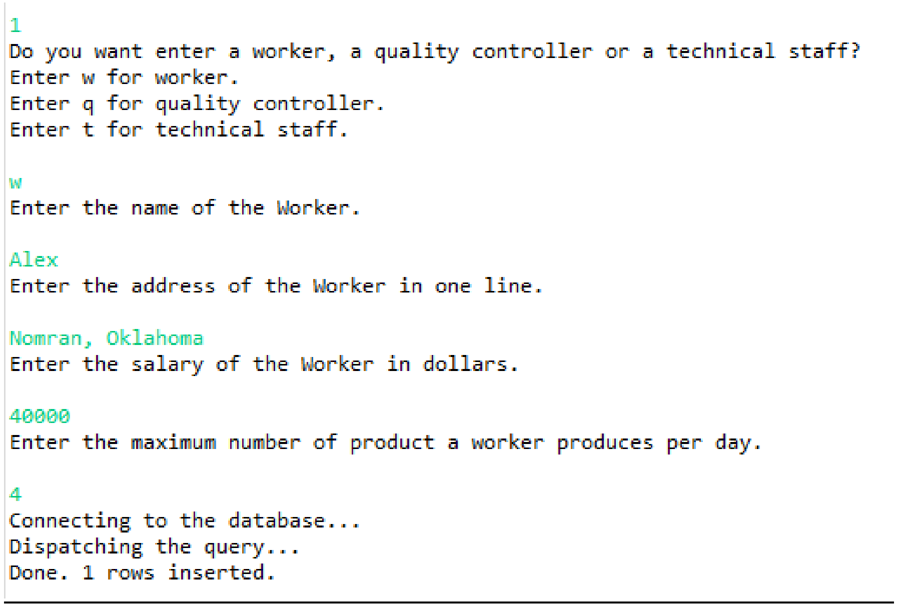
```
    public static void option15(Scanner sc) throws SQLException {

      //Ask the user for the starting year.
      System.out.println("Enter the earliest year in the range.");
      String startYear = sc.next();
      //Ask the user for the ending year.
      System.out.println("Enter the last year in the range.");
      String endYear = sc.next();

      System.out.println("Connecting to the database...");

      /*Before deleting from accident, we should delete from ProduceCasue
because
         the data in ProduceCasue refer to Accident*/
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY15_PRODUCE_CAUSE)){
                  statement.setString(1, startYear);
                  statement.setString(2, endYear);
                  final int rows_deleted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows deleted.",
rows_deleted));

            }
      }

      /*Before deleting from accident, we should delete from RepairCasue
because
        the data in RepairCasue refer to Accident*/
      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY15_REPAIR_CAUSE)){
                  statement.setString(1, startYear);
                  statement.setString(2, endYear);
                final int rows_deleted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows deleted.",
rows_deleted));
            }
      }

      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY15_ACCIDENT)){
                  statement.setString(1, startYear);
                  statement.setString(2, endYear);
                  System.out.println("DELETING");
                  final int rows_deleted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows deleted.", 
rows_deleted));
            }
      }


      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (
                  final Statement statement = connection.createStatement();
                  final ResultSet resultSet = 
statement.executeQuery(QUERY15_PRINT)) {

                  System.out.println("Here is the result after delete:");
                        System.out.println("Accident number | accident date | 
number of work day lost");
                        // Unpack the tuples returned by the database and 
print them out to the user
                    while (resultSet.next()) {
                        System.out.println(String.format("%s | %s | %s",

                                    resultSet.getString(1),
                                    resultSet.getString(2),
                                    resultSet.getString(3)));

                    }
                  }
      }

    }

    //Option 16 ----------------------------------------------------------------
-------------------
    public static void option16(Scanner sc) throws SQLException {
      System.out.println("Enter the name of the file including the file type. 
ex: .csv");
      String fileName = sc.next();

      //specify the location of the file.
      File file = new File("src/" + fileName);

      //Make a scanner and read the values separated by commas.
      try {
            Scanner myReader = new Scanner(file);
            myReader.useDelimiter(",");

            //This array includes all the attributes for a worker
            String[] workerAttr = new String[4];
           //This array includes all the attributes for a quality controller.
            String[] qualityControllerAttr = new String[4];
            //This array includes all the attributes for a technical staff.
            String[] techStaffAttr= new String[5];
            //a variable that will hold the values of the attributes.
            String data =""; //Initialize it to empty string.

            //Keep scanning the file until there is no more lines.
            while (myReader.hasNextLine()) {
              data = myReader.next();
              //If the row from the list is a worker, then take all the
              //attributes and put then in its array
              if(data.contains("worker")) {
                    for(int i=0 ; i<4 ;i++) {
                              workerAttr[i] = myReader.next();
                    }

                    try (final Connection connection =
DriverManager.getConnection(URL)) {
                          try (
                              final PreparedStatement statement =
connection.prepareStatement(QUERY1_WORKER)) {
                              // Populate the query template with the data
collected from the user
                              statement.setString(1, workerAttr[0]);
                              statement.setString(2, workerAttr[1]);
                              statement.setDouble(3,
Double.valueOf(workerAttr[2]));
                              statement.setInt(4,
Integer.valueOf(workerAttr[3]));

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

             //If the row from the list is a quality controller, then take
all the
              //attributes and put then in its array
              else if(data.contains("quality controller")) {
                    qualityControllerAttr[0] = myReader.next(); //read the
name
                    qualityControllerAttr[1] = myReader.next(); //read the
address
                    qualityControllerAttr[2] = myReader.next(); //read the
salary
```

```
                  //since this column includes the maximum number of 
production per year read it,
                    //but don't store it because it does not apply to quality 
controller
                    myReader.next();
                    qualityControllerAttr[3] = myReader.next();


                    try (final Connection connection = 
DriverManager.getConnection(URL)) {
                          try (
                              final PreparedStatement statement = 
connection.prepareStatement(QUERY1_QUALITY_CONTROLLER)) {
                              // Populate the query template with the data 
collected from the user
                             statement.setString(1,qualityControllerAttr[0]);
                              statement.setString(2, 
qualityControllerAttr[1]);
                              statement.setDouble(3, 
Double.valueOf(qualityControllerAttr[2]));
                              statement.setString(4, 
qualityControllerAttr[3]);

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

            //If the row from the list is a technical staff, then take all 
the
              //attributes and put then in its array
              else if(data.contains("technical staff")) {
                    techStaffAttr[0] = myReader.next();
                    techStaffAttr[1] = myReader.next();
                    techStaffAttr[2] = myReader.next();
                    myReader.next(); //this column does not apply to 
technical staff. Read but don't store.
                    myReader.next(); //this column does not apply to 
technical staff. Read but don't store.
                    techStaffAttr[3] = myReader.next();
                    techStaffAttr[4] = myReader.next();

                    try (final Connection connection = 
DriverManager.getConnection(URL)) {
                          try (
                              final PreparedStatement statement =
connection.prepareStatement(QUERY1_TECHNICAL_STAFF)) {
                              // Populate the query template with the data
collected from the user
                              statement.setString(1, techStaffAttr[0]);
                              statement.setString(2, techStaffAttr[1]);
                              statement.setDouble(3,
Double.valueOf(techStaffAttr[2]));
                              statement.setString(4, techStaffAttr[3]);
                              statement.setString(5, techStaffAttr[4]);

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

          }

          myReader.close();
          } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
          }
    }

    //Option 17 -------------------------------------------------------------
---------------------------------
  //Exports result of query 11 to a file
    public static void option17(Scanner sc) throws SQLException,
FileNotFoundException{
      System.out.println("Enter the name of the file including the format.");
      String fileName = sc.next();
      System.out.println("Enter the color.");
      String q11Color = sc.next();
      System.out.println("Connecting to the database...");

      try (final Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Dispatching the query...");
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY11)){
                  statement.setString(1, q11Color);

                PrintStream output = new PrintStream(fileName);
                output.printf("Here is the result:\n");
                output.printf("Customer Names:\n");
```

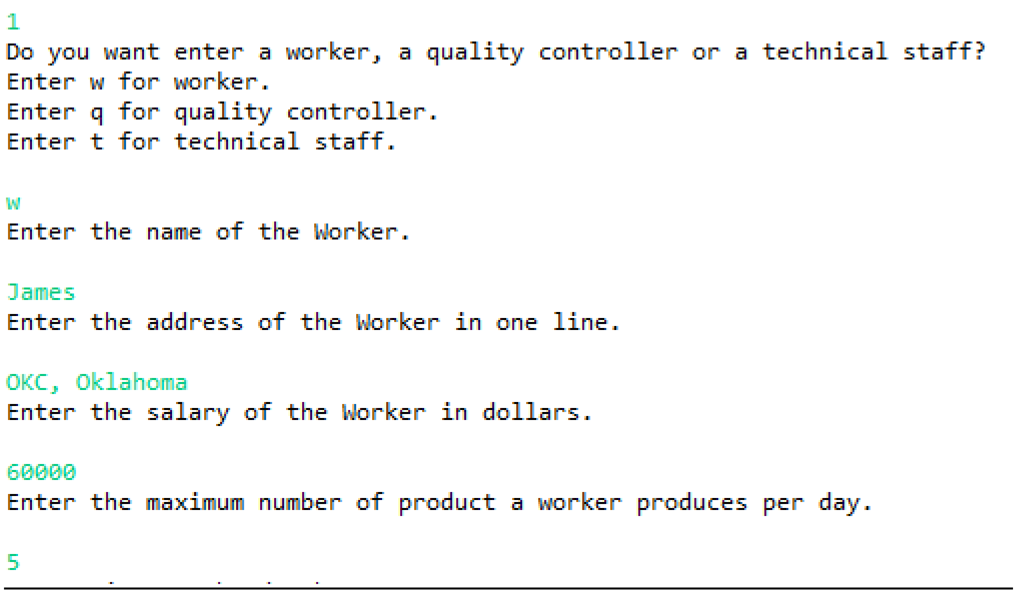
```
                  // Unpack the tuples returned by the database and print
them out to the user
                  ResultSet resultSet = statement.executeQuery();
                while (resultSet.next()) {
                  output.printf(String.format("%s \n",
resultSet.getString(1)));
                }
                output.close();
            }
      }

    }


}
```

**Task 6.**

## 6.1. Screenshots showing the testing of query 1



```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

w
Enter the name of the Worker.

Alex
Enter the address of the Worker in one line.

Nomran, Oklahoma
Enter the salary of the Worker in dollars.

40000
Enter the maximum number of product a worker produces per day.

4
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```



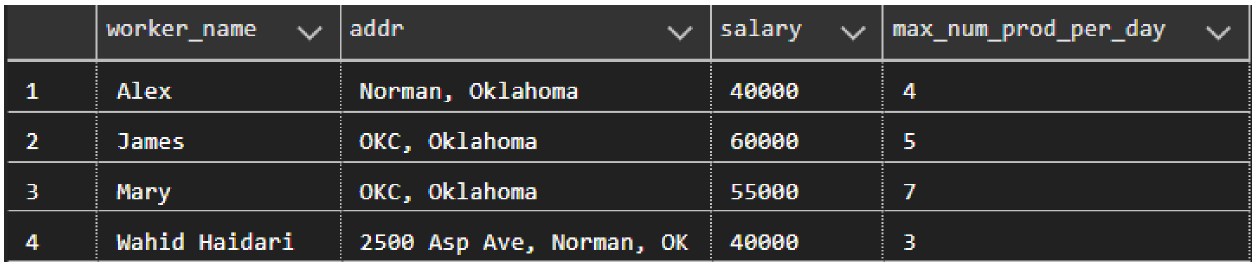
```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

w
Enter the name of the Worker.

James
Enter the address of the Worker in one line.

OKC, Oklahoma
Enter the salary of the Worker in dollars.

60000
Enter the maximum number of product a worker produces per day.

5
```

```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

w
Enter the name of the Worker.
Mary
Enter the address of the Worker in one line.
OKC, Oklahoma
Enter the salary of the Worker in dollars.
55000
Enter the maximum number of product a worker produces per day.
7
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

w
Enter the name of the Worker.
Wahid Haidari
Enter the address of the Worker in one line.
2500 Asp Ave, Norman, OK
Enter the salary of the Worker in dollars.
40000
Enter the maximum number of product a worker produces per day.
3
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Worker after insertions

SELECT * FROM Worker;



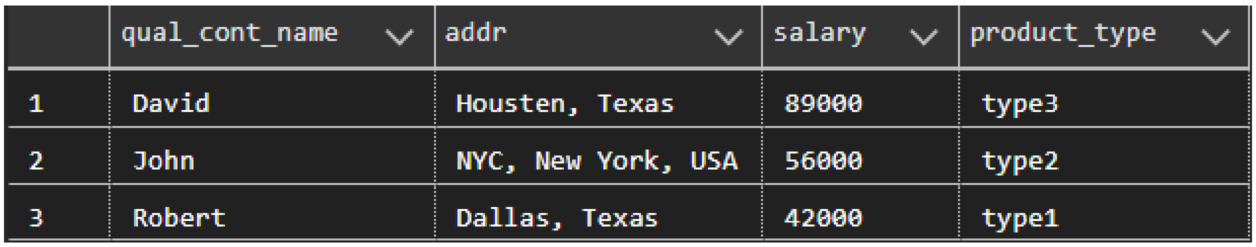
| | worker_name | addr | salary | max_num_prod_per_day |
|---|---|---|---|---|
| 1 | Alex | Norman, Oklahoma | 40000 | 4 |
| 2 | James | OKC, Oklahoma | 60000 | 5 |
| 3 | Mary | OKC, Oklahoma | 55000 | 7 |
| 4 | Wahid Haidari | 2500 Asp Ave, Norman, OK | 40000 | 3 |

```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

q
Enter the name of the quality controller.
David
Enter the address of the quality controller in one line.
Housten, Texas
Enter the salary of the quality controller.
89000
Enter the product type the quality controller tests.
enter type1 for type 1
enter type1 for type 2
enter type1 for type 3.
type3
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.


1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

q
Enter the name of the quality controller.
John
Enter the address of the quality controller in one line.
NYC, New York, USA
Enter the salary of the quality controller.
56000
Enter the product type the quality controller tests.
enter type1 for type 1
enter type1 for type 2
enter type1 for type 3.
type2
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

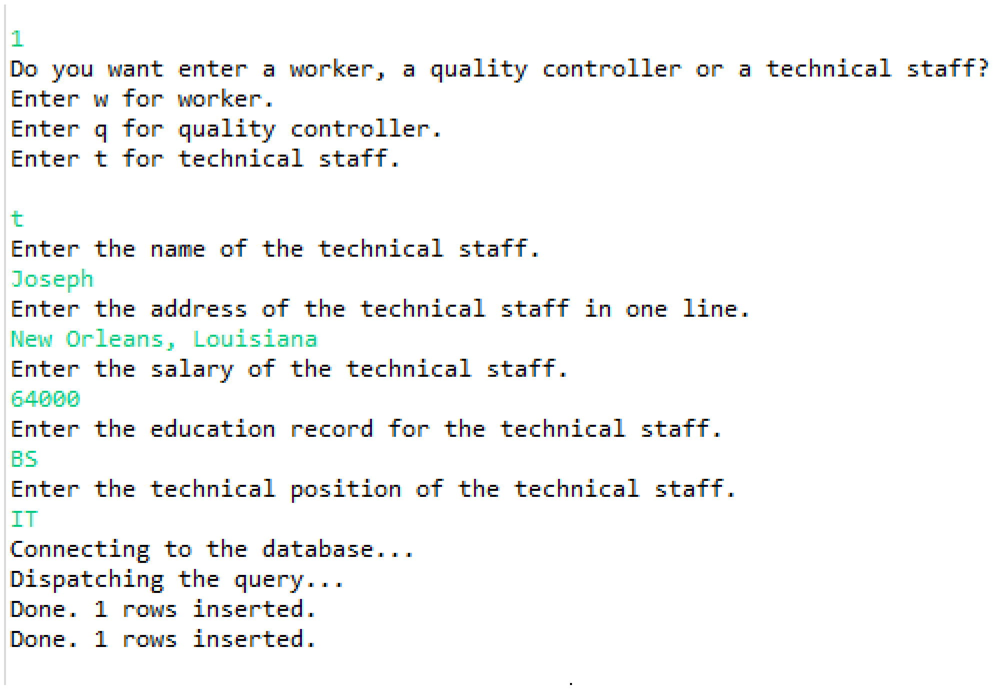
```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

q
Enter the name of the quality controller.
Robert
Enter the address of the quality controller in one line.
Dallas, Texas
Enter the salary of the quality controller.
42000
Enter the product type the quality controller tests.
enter type1 for type 1
enter type1 for type 2
enter type1 for type 3.
type1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table QualityController after insertions

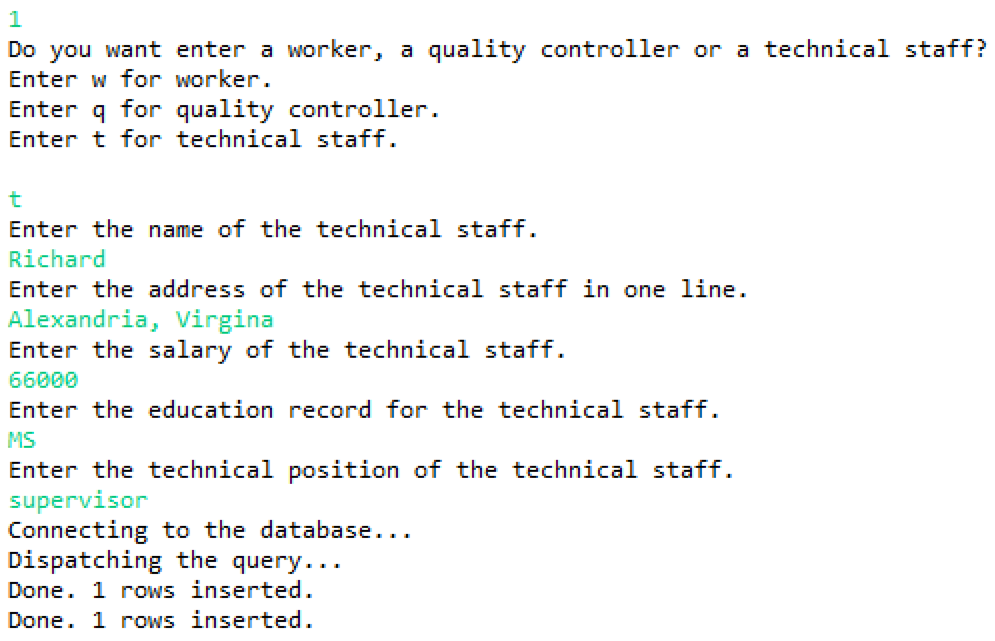
SELECT * FROM QualityController;



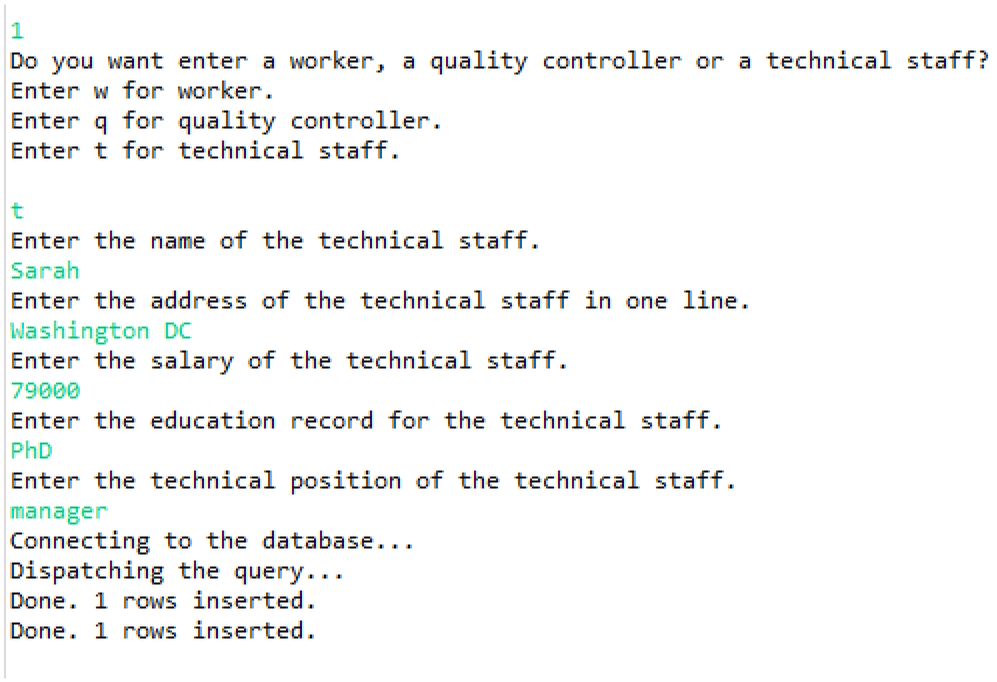
| | qual_cont_name | addr | salary | product_type |
|---|---|---|---|---|
| 1 | David | Housten, Texas | 89000 | type3 |
| 2 | John | NYC, New York, USA | 56000 | type2 |
| 3 | Robert | Dallas, Texas | 42000 | type1 |



```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

t
Enter the name of the technical staff.
Joseph
Enter the address of the technical staff in one line.
New Orleans, Louisiana
Enter the salary of the technical staff.
64000
Enter the education record for the technical staff.
BS
Enter the technical position of the technical staff.
IT
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Done. 1 rows inserted.
```



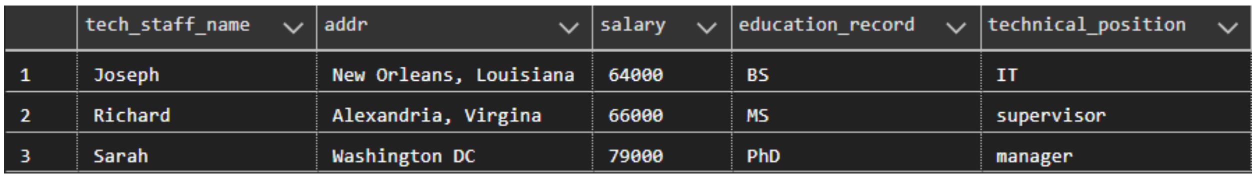
```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

t
Enter the name of the technical staff.
Richard
Enter the address of the technical staff in one line.
Alexandria, Virgina
Enter the salary of the technical staff.
66000
Enter the education record for the technical staff.
MS
Enter the technical position of the technical staff.
supervisor
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Done. 1 rows inserted.
```



```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

t
Enter the name of the technical staff.
Sarah
Enter the address of the technical staff in one line.
Washington DC
Enter the salary of the technical staff.
79000
Enter the education record for the technical staff.
PhD
Enter the technical position of the technical staff.
manager
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Done. 1 rows inserted.
```

Content of the Table TechnicalStaff after insertions:

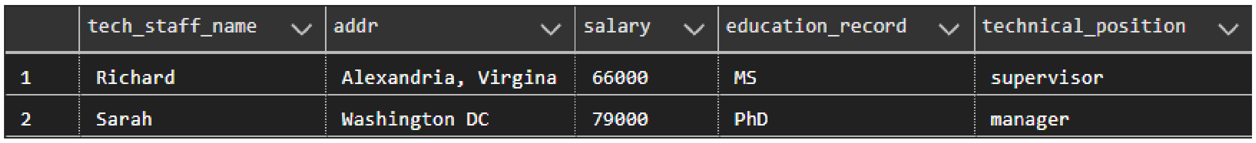
SELECT * FROM TechnicalStaff;



| | tech_staff_name | addr | salary | education_record | technical_position |
|---|---|---|---|---|---|
| 1 | Joseph | New Orleans, Louisiana | 64000 | BS | IT |
| 2 | Richard | Alexandria, Virgina | 66000 | MS | supervisor |
| 3 | Sarah | Washington DC | 79000 | PhD | manager |

Content of the Table GradTechnicalStaff after insertions:

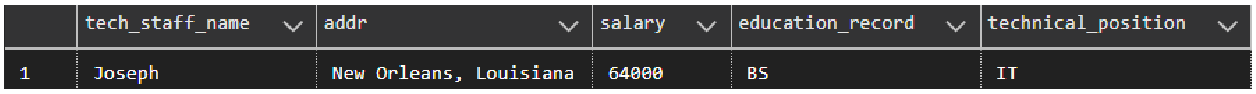
SELECT * FROM GradTechnicalStaff;



| | tech_staff_name | addr | salary | education_record | technical_position |
|---|---|---|---|---|---|
| 1 | Richard | Alexandria, Virgina | 66000 | MS | supervisor |
| 2 | Sarah | Washington DC | 79000 | PhD | manager |

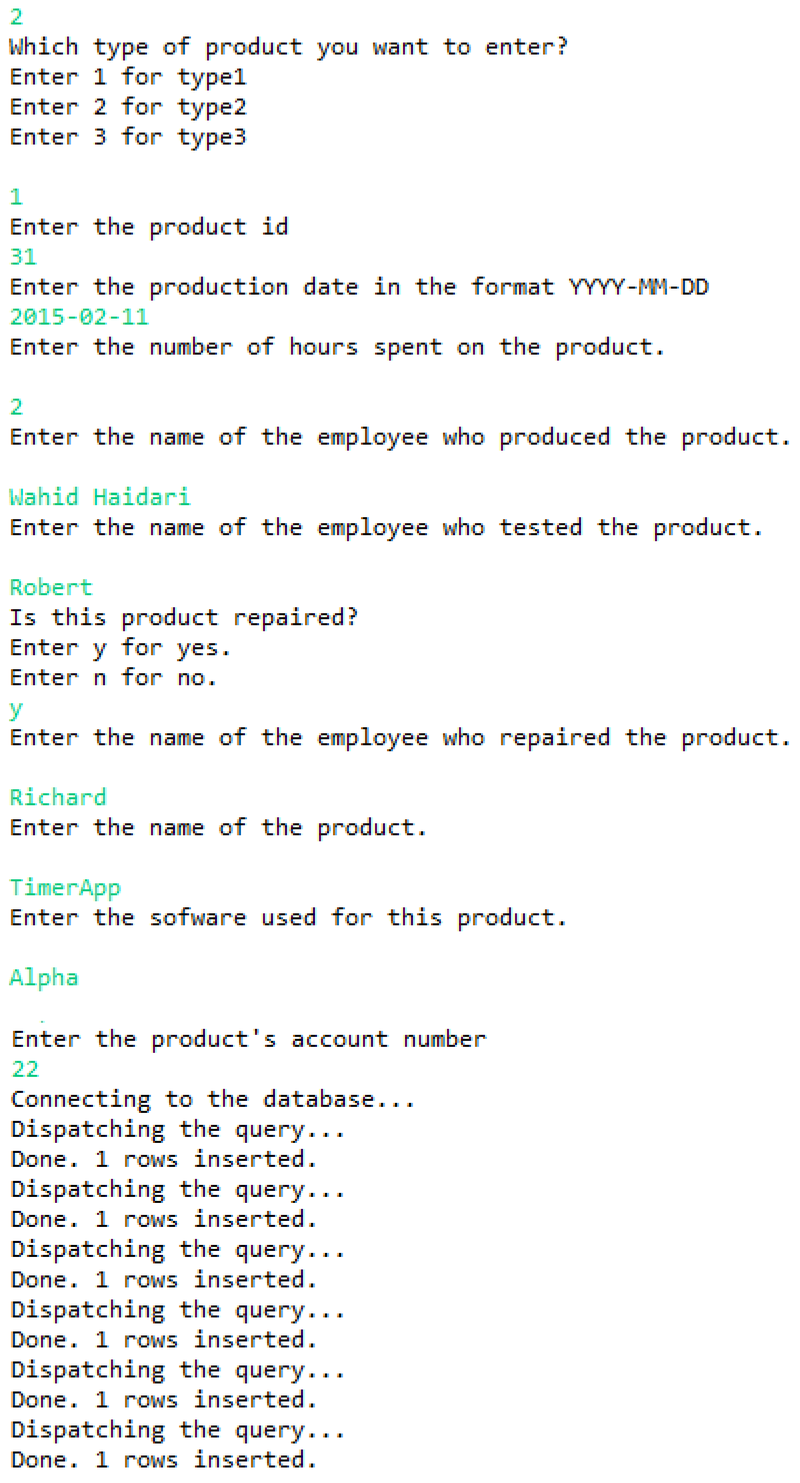
Content of the Table UndergradTechnicalStaff after insertions:

SELECT * FROM UndergradTechnicalStaff;

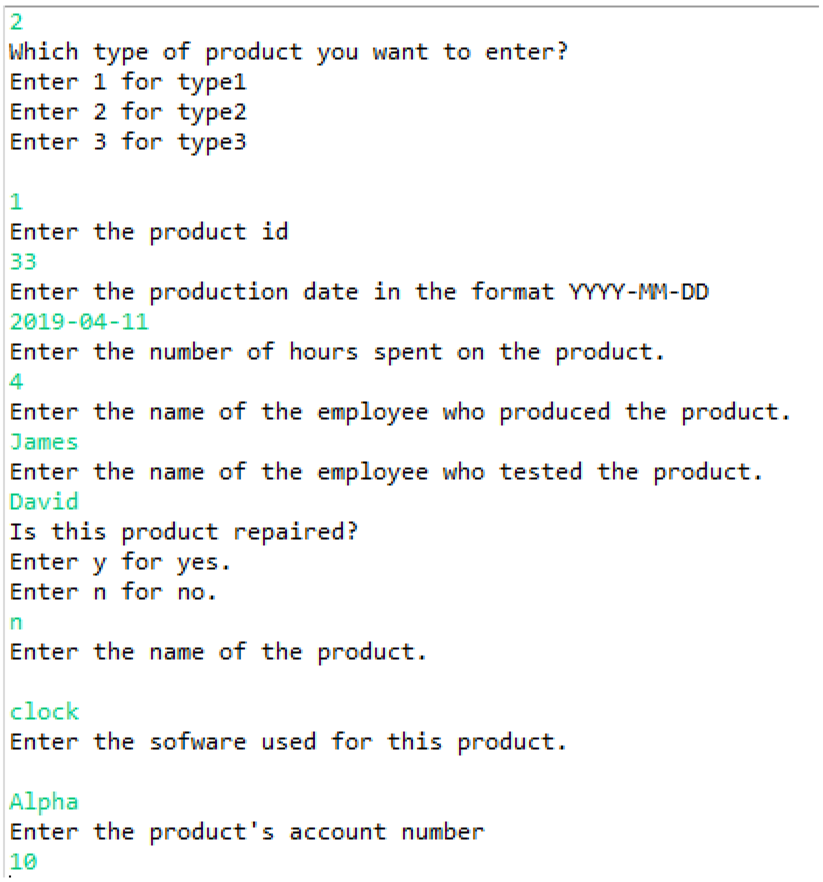


| | tech_staff_name | addr | salary | education_record | technical_position |
|---|---|---|---|---|---|
| 1 | Joseph | New Orleans, Louisiana | 64000 | BS | IT |

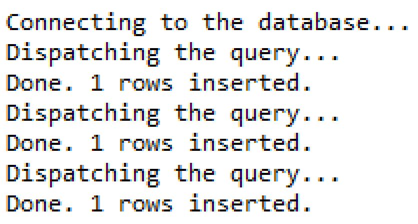
### 6.2. Screenshots showing the testing of query 2



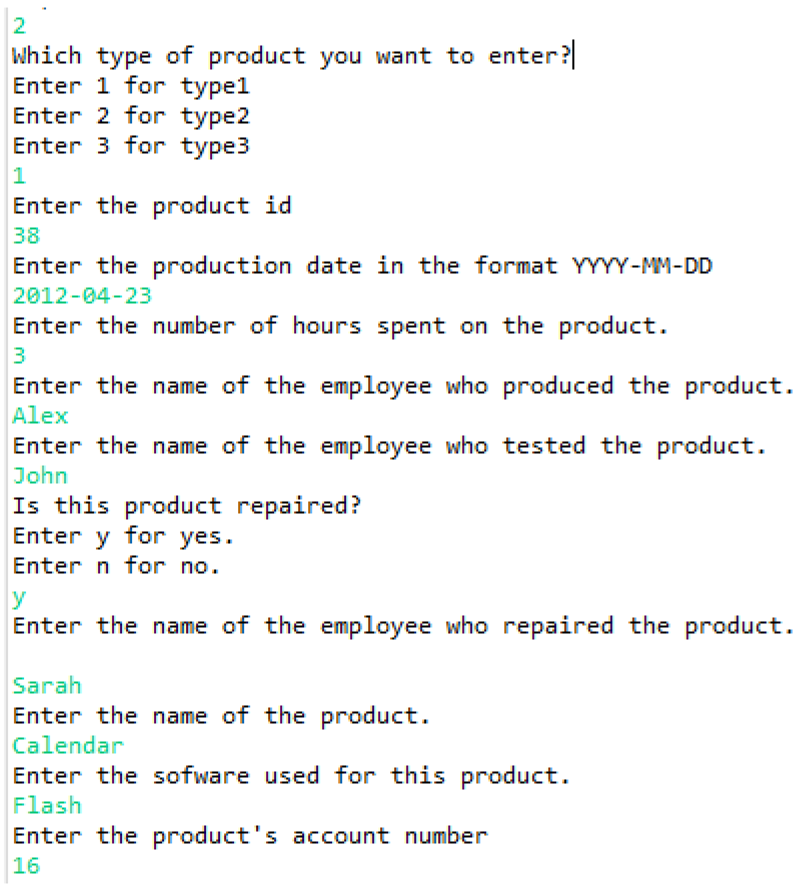
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3

1
Enter the product id
31
Enter the production date in the format YYYY-MM-DD
2015-02-11
Enter the number of hours spent on the product.

2
Enter the name of the employee who produced the product.

Wahid Haidari
Enter the name of the employee who tested the product.

Robert
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Richard
Enter the name of the product.

TimerApp
Enter the sofware used for this product.

Alpha

Enter the product's account number
22
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



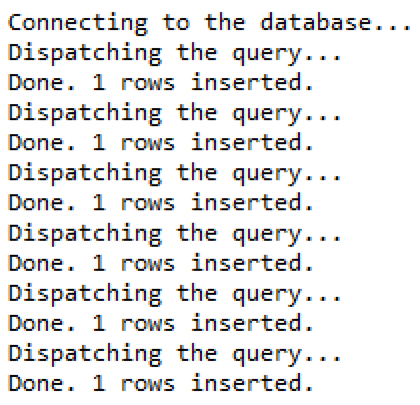
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3

1
Enter the product id
33
Enter the production date in the format YYYY-MM-DD
2019-04-11
Enter the number of hours spent on the product.
4
Enter the name of the employee who produced the product.
James
Enter the name of the employee who tested the product.
David
Is this product repaired?
Enter y for yes.
Enter n for no.
n
Enter the name of the product.

clock
Enter the sofware used for this product.

Alpha
Enter the product's account number
10
```



```
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



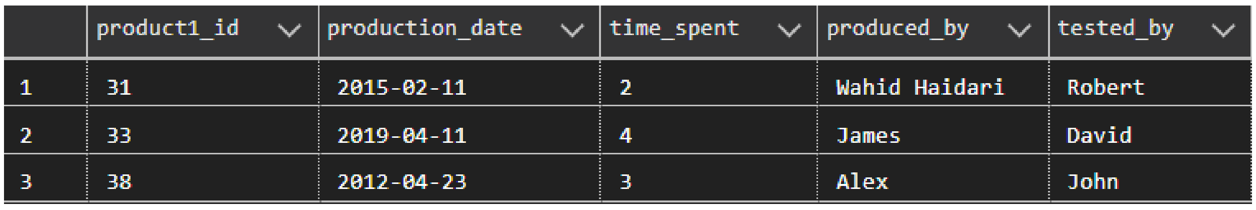
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
1
Enter the product id
38
Enter the production date in the format YYYY-MM-DD
2012-04-23
Enter the number of hours spent on the product.
3
Enter the name of the employee who produced the product.
Alex
Enter the name of the employee who tested the product.
John
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Sarah
Enter the name of the product.
Calendar
Enter the sofware used for this product.
Flash
Enter the product's account number
16
```



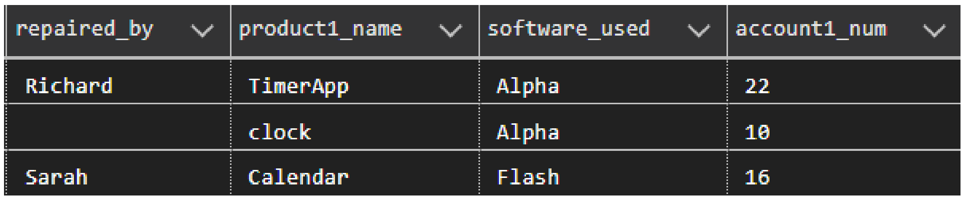
```
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Product1 after insertions:

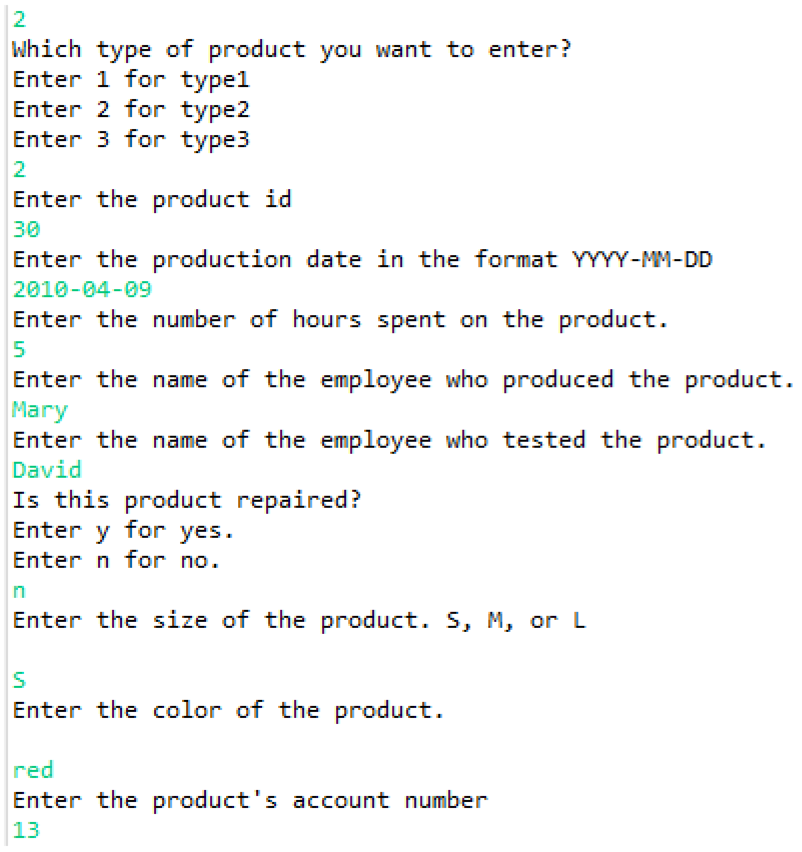
SELECT * FROM Product1;



| | product1_id | production_date | time_spent | produced_by | tested_by |
|---|---|---|---|---|---|
| 1 | 31 | 2015-02-11 | 2 | Wahid Haidari | Robert |
| 2 | 33 | 2019-04-11 | 4 | James | David |
| 3 | 38 | 2012-04-23 | 3 | Alex | John |

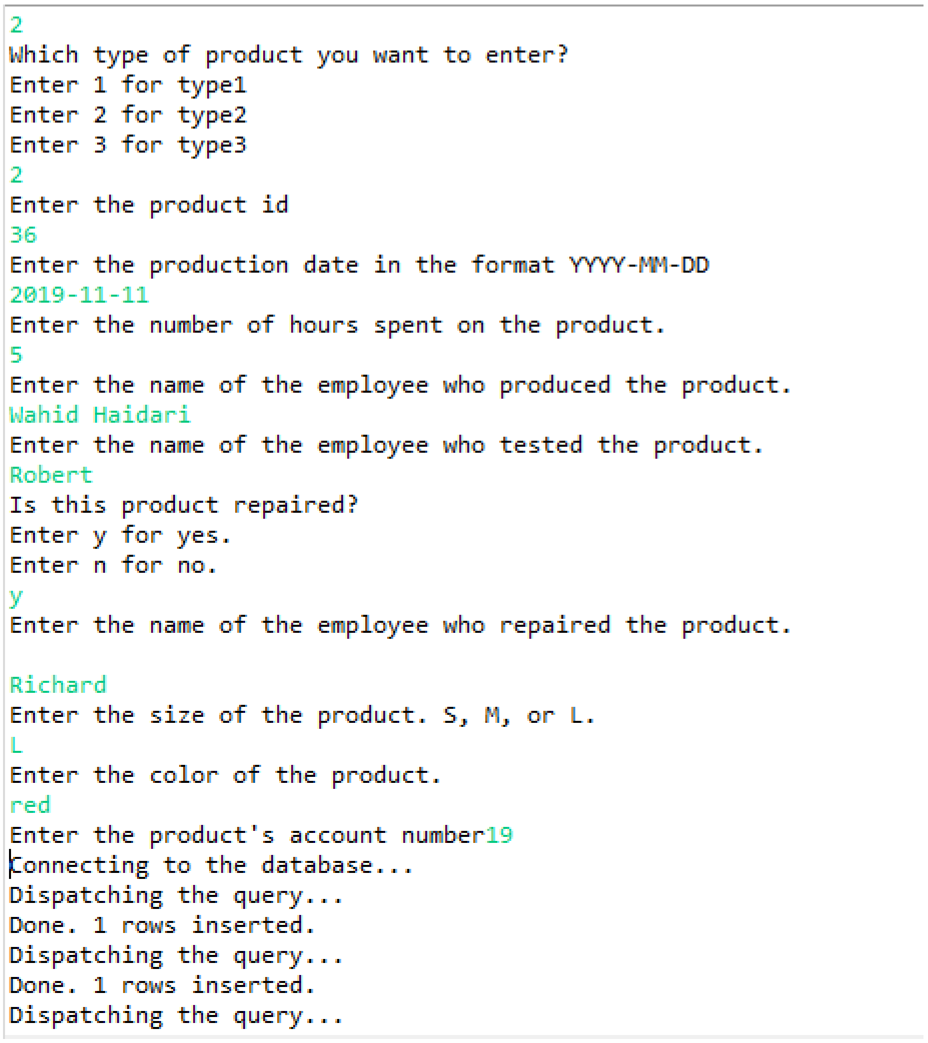


| repaired_by | product1_name | software_used | account1_num |
|---|---|---|---|
| Richard | TimerApp | Alpha | 22 |
| | clock | Alpha | 10 |
| Sarah | Calendar | Flash | 16 |

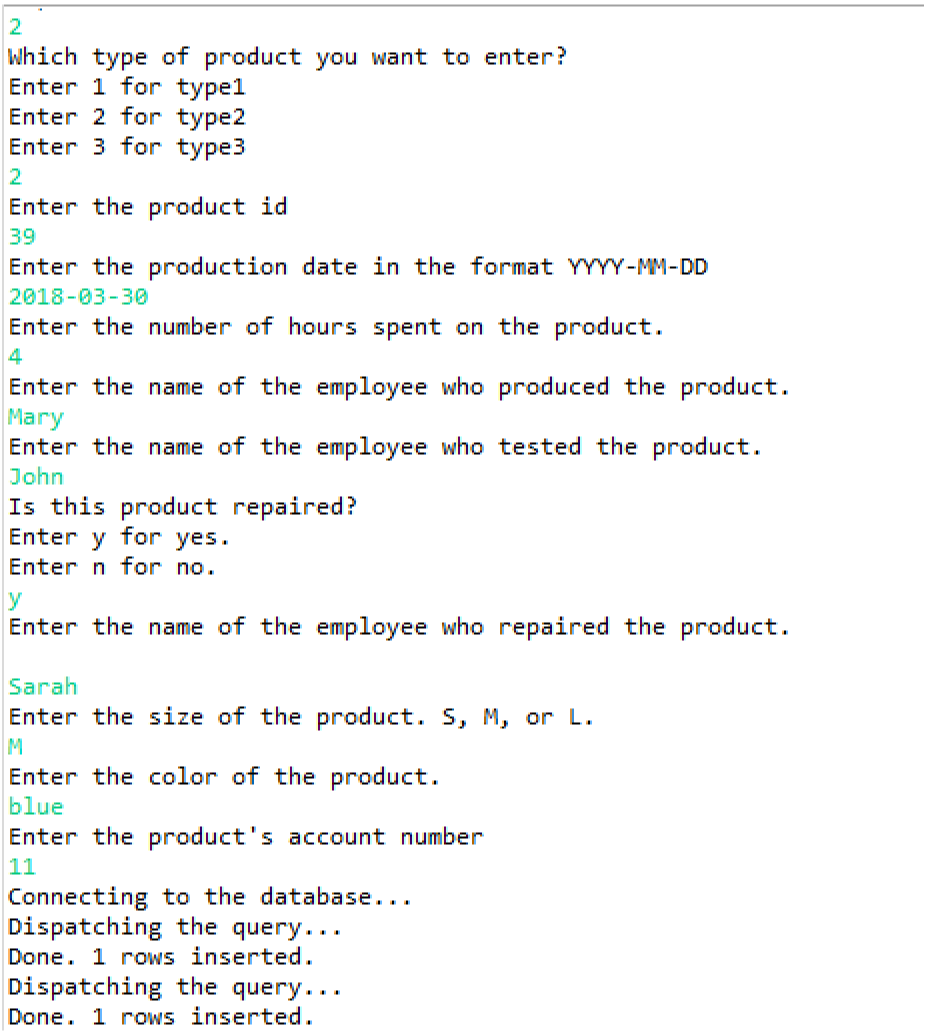


```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
2
Enter the product id
30
Enter the production date in the format YYYY-MM-DD
2010-04-09
Enter the number of hours spent on the product.
5
Enter the name of the employee who produced the product.
Mary
Enter the name of the employee who tested the product.
David
Is this product repaired?
Enter y for yes.
Enter n for no.
n
Enter the size of the product. S, M, or L

S
Enter the color of the product.

red
Enter the product's account number
13
```

```
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



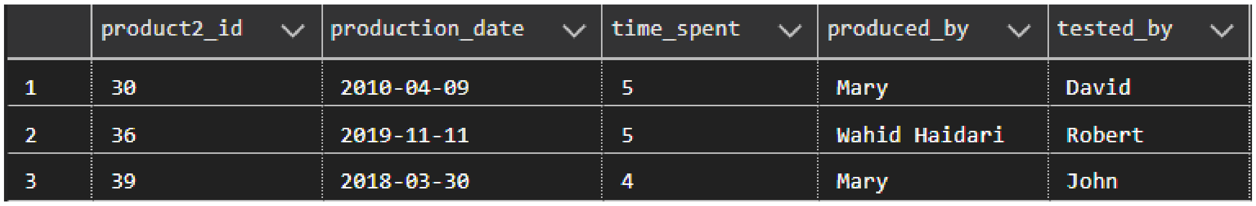
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
2
Enter the product id
36
Enter the production date in the format YYYY-MM-DD
2019-11-11
Enter the number of hours spent on the product.
5
Enter the name of the employee who produced the product.
Wahid Haidari
Enter the name of the employee who tested the product.
Robert
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Richard
Enter the size of the product. S, M, or L.
L
Enter the color of the product.
red
Enter the product's account number19
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
```

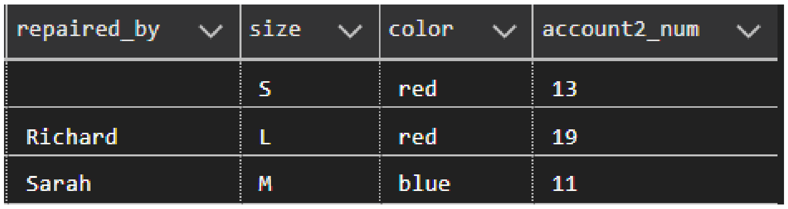


```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
2
Enter the product id
39
Enter the production date in the format YYYY-MM-DD
2018-03-30
Enter the number of hours spent on the product.
4
Enter the name of the employee who produced the product.
Mary
Enter the name of the employee who tested the product.
John
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Sarah
Enter the size of the product. S, M, or L.
M
Enter the color of the product.
blue
Enter the product's account number
11
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

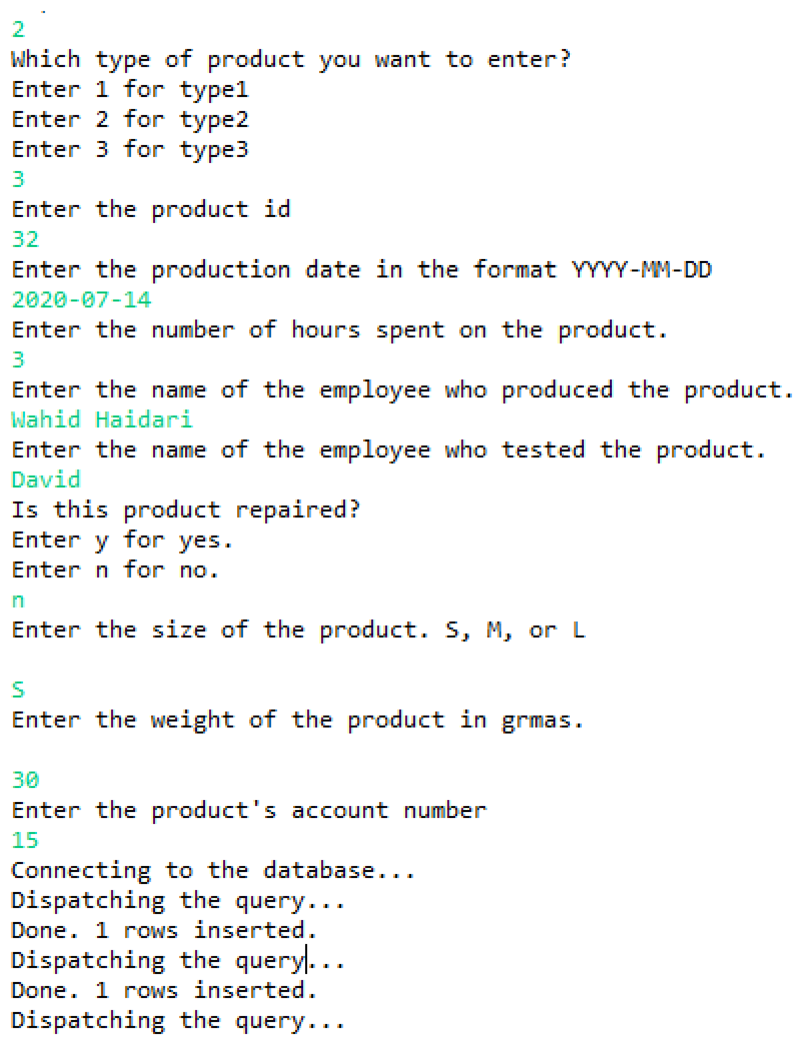
Content of the Table Product2 after insertions:

SELECT * FROM Product2;

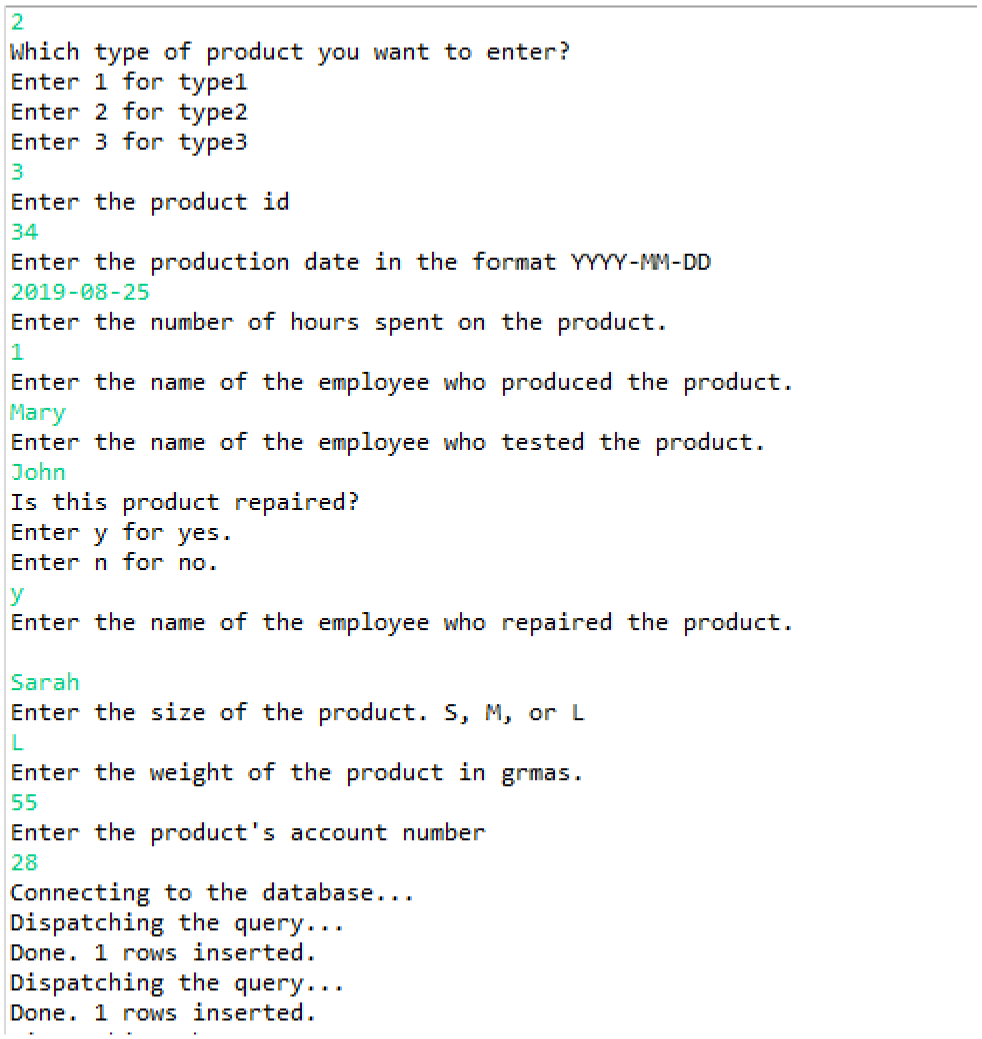




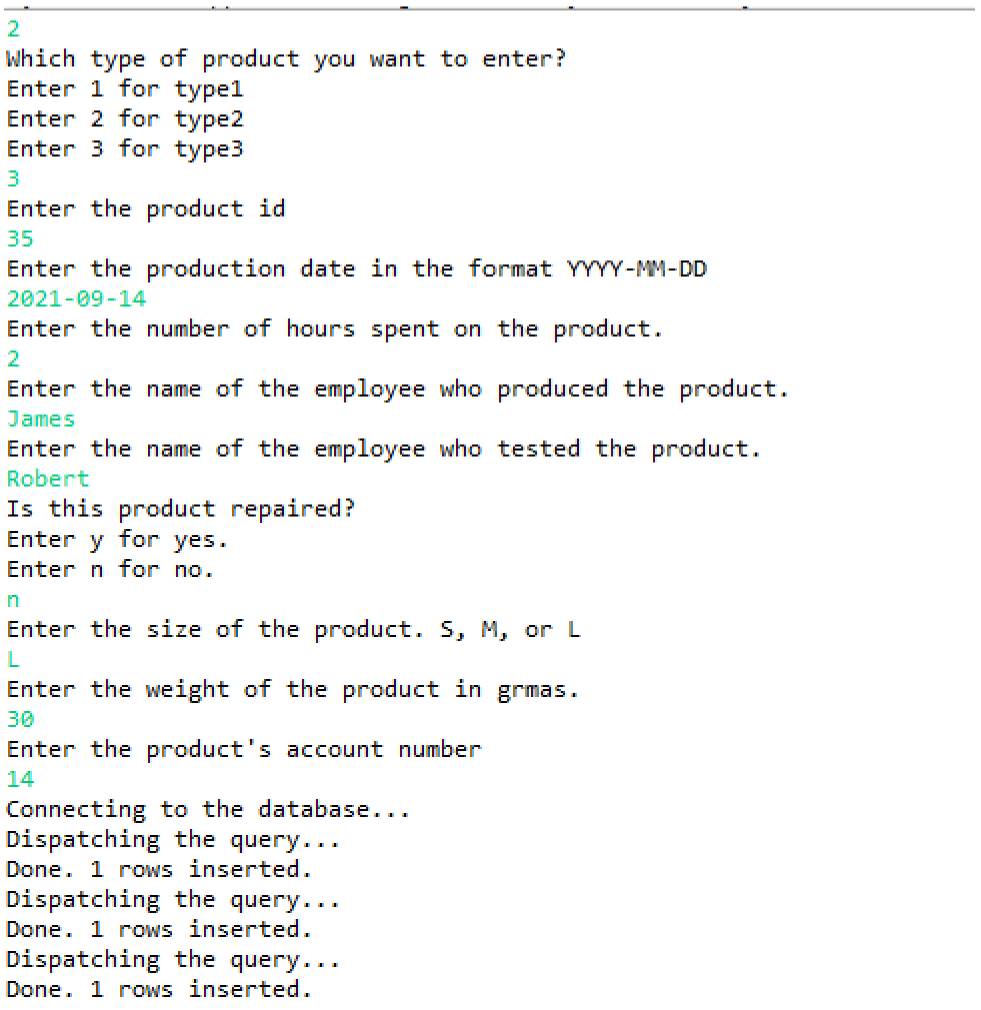
| | product2_id | production_date | time_spent | produced_by | tested_by | repaired_by | size | color | account2_num |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 30 | 2010-04-09 | 5 | Mary | David | | S | red | 13 |
| 2 | 36 | 2019-11-11 | 5 | Wahid Haidari | Robert | Richard | L | red | 19 |
| 3 | 39 | 2018-03-30 | 4 | Mary | John | Sarah | M | blue | 11 |



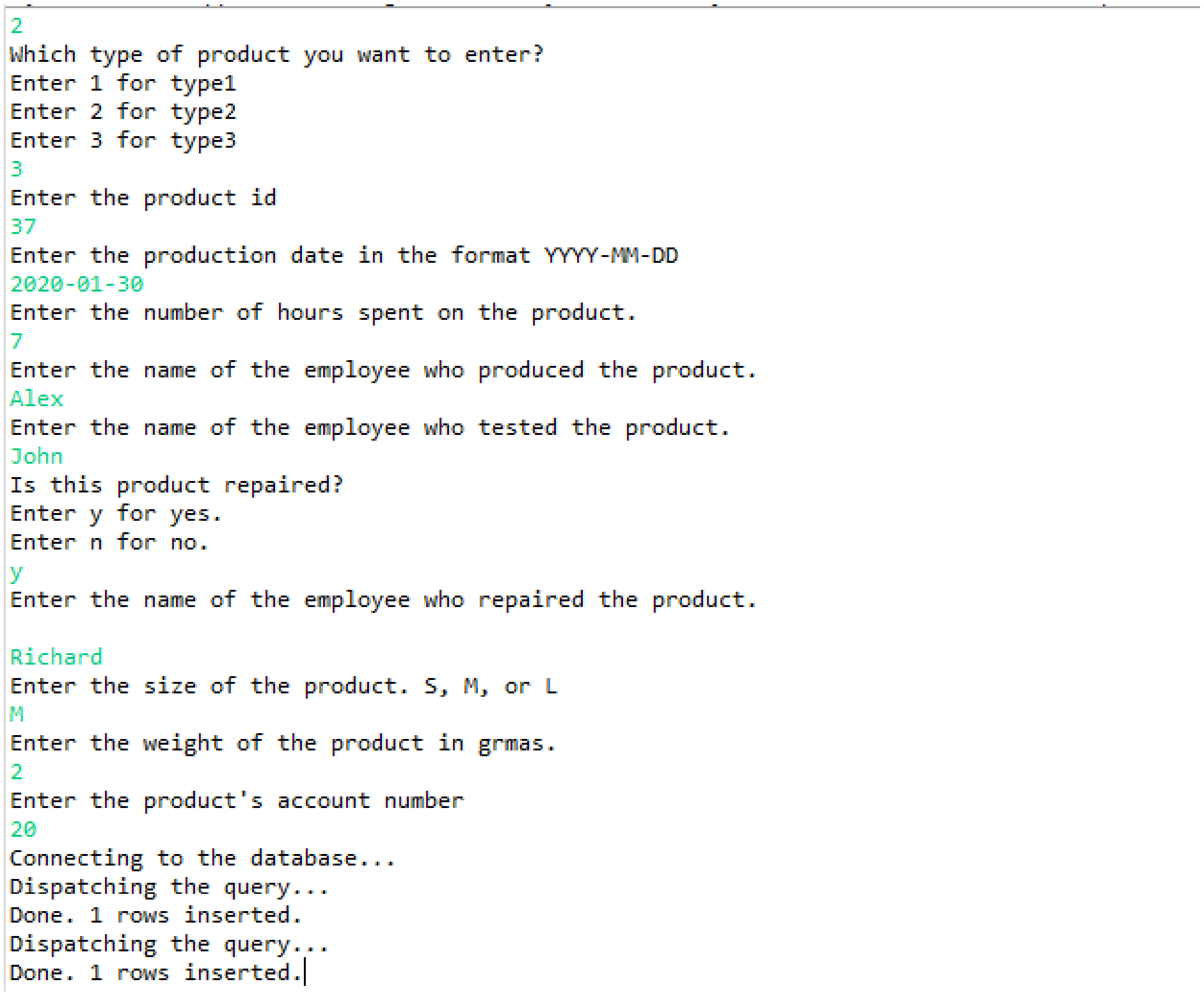
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
3
Enter the product id
32
Enter the production date in the format YYYY-MM-DD
2020-07-14
Enter the number of hours spent on the product.
3
Enter the name of the employee who produced the product.
Wahid Haidari
Enter the name of the employee who tested the product.
David
Is this product repaired?
Enter y for yes.
Enter n for no.
n
Enter the size of the product. S, M, or L

S
Enter the weight of the product in grmas.

30
Enter the product's account number
15
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
```



```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
3
Enter the product id
34
Enter the production date in the format YYYY-MM-DD
2019-08-25
Enter the number of hours spent on the product.
1
Enter the name of the employee who produced the product.
Mary
Enter the name of the employee who tested the product.
John
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Sarah
Enter the size of the product. S, M, or L
L
Enter the weight of the product in grmas.
55
Enter the product's account number
28
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



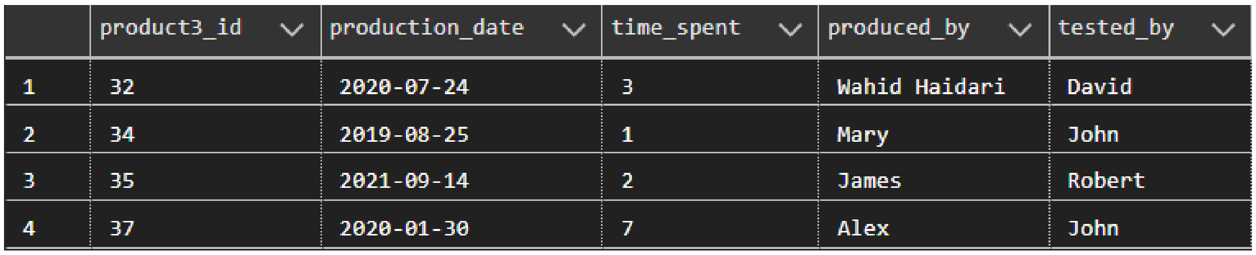
```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
3
Enter the product id
35
Enter the production date in the format YYYY-MM-DD
2021-09-14
Enter the number of hours spent on the product.
2
Enter the name of the employee who produced the product.
James
Enter the name of the employee who tested the product.
Robert
Is this product repaired?
Enter y for yes.
Enter n for no.
n
Enter the size of the product. S, M, or L
L
Enter the weight of the product in grmas.
30
Enter the product's account number
14
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

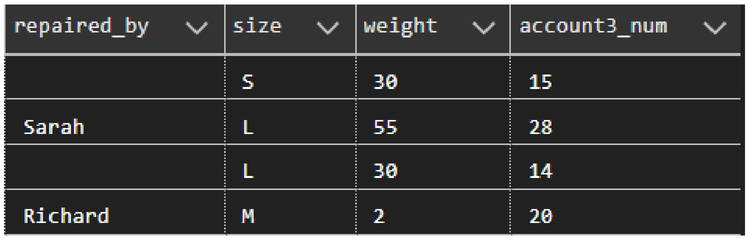


```
2
Which type of product you want to enter?
Enter 1 for type1
Enter 2 for type2
Enter 3 for type3
3
Enter the product id
37
Enter the production date in the format YYYY-MM-DD
2020-01-30
Enter the number of hours spent on the product.
7
Enter the name of the employee who produced the product.
Alex
Enter the name of the employee who tested the product.
John
Is this product repaired?
Enter y for yes.
Enter n for no.
y
Enter the name of the employee who repaired the product.

Richard
Enter the size of the product. S, M, or L
M
Enter the weight of the product in grmas.
2
Enter the product's account number
20
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Product3 after insertions:

SELECT * FROM Product3;

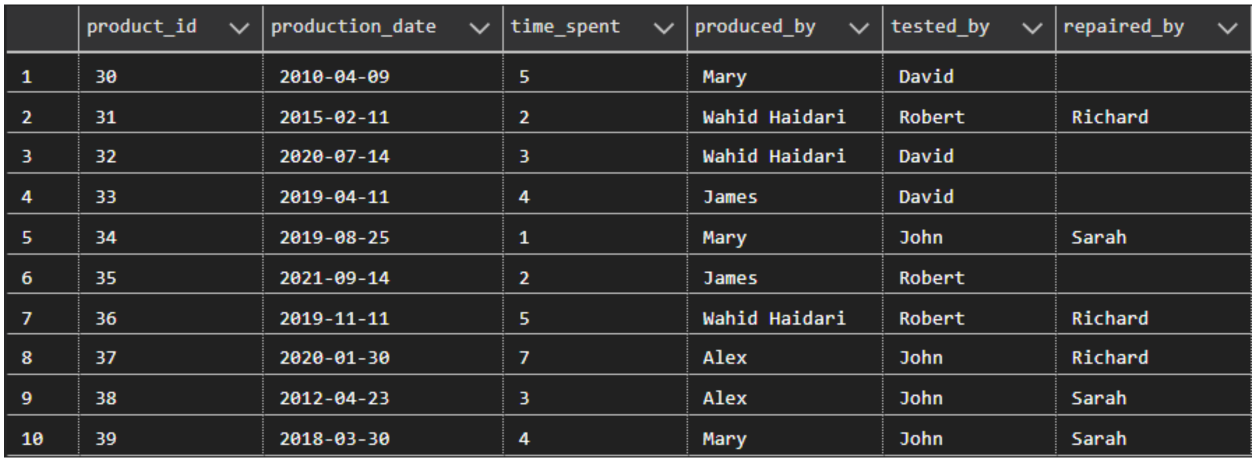




| | product3_id | production_date | time_spent | produced_by | tested_by | repaired_by | size | weight | account3_num |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 32 | 2020-07-24 | 3 | Wahid Haidari | David | | S | 30 | 15 |
| 2 | 34 | 2019-08-25 | 1 | Mary | John | Sarah | L | 55 | 28 |
| 3 | 35 | 2021-09-14 | 2 | James | Robert | | L | 30 | 14 |
| 4 | 37 | 2020-01-30 | 7 | Alex | John | Richard | M | 2 | 20 |

Content of the Table Product after insertions:

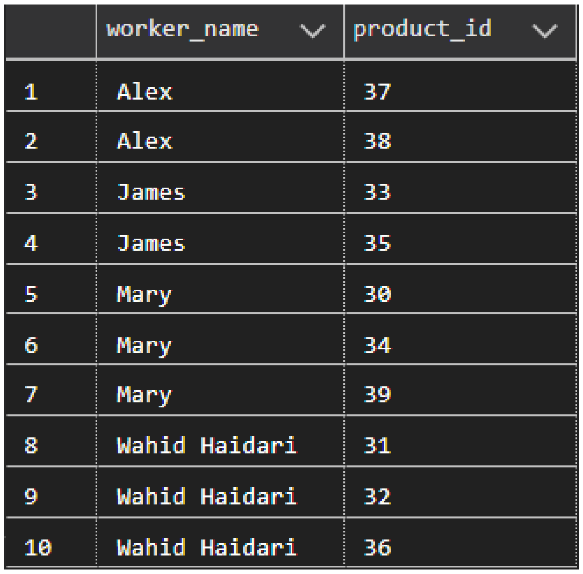
SELECT * FROM Product;



| | product_id | production_date | time_spent | produced_by | tested_by | repaired_by |
|---|---|---|---|---|---|---|
| 1 | 30 | 2010-04-09 | 5 | Mary | David | |
| 2 | 31 | 2015-02-11 | 2 | Wahid Haidari | Robert | Richard |
| 3 | 32 | 2020-07-14 | 3 | Wahid Haidari | David | |
| 4 | 33 | 2019-04-11 | 4 | James | David | |
| 5 | 34 | 2019-08-25 | 1 | Mary | John | Sarah |
| 6 | 35 | 2021-09-14 | 2 | James | Robert | |
| 7 | 36 | 2019-11-11 | 5 | Wahid Haidari | Robert | Richard |
| 8 | 37 | 2020-01-30 | 7 | Alex | John | Richard |
| 9 | 38 | 2012-04-23 | 3 | Alex | John | Sarah |
| 10 | 39 | 2018-03-30 | 4 | Mary | John | Sarah |

Content of the Table Produce after insertions:

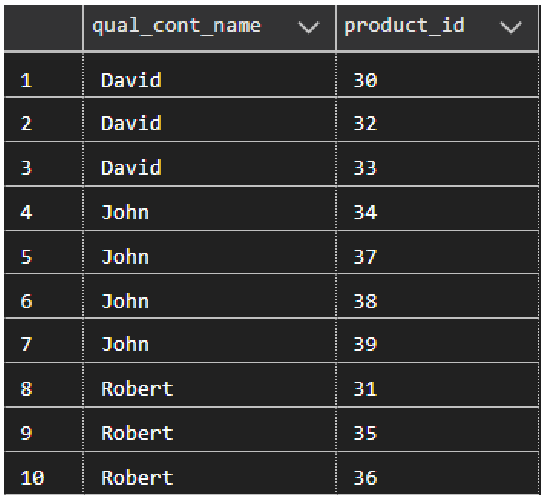
SELECT * FROM Produce;



| | worker_name | product_id |
|---|---|---|
| 1 | Alex | 37 |
| 2 | Alex | 38 |
| 3 | James | 33 |
| 4 | James | 35 |
| 5 | Mary | 30 |
| 6 | Mary | 34 |
| 7 | Mary | 39 |
| 8 | Wahid Haidari | 31 |
| 9 | Wahid Haidari | 32 |
| 10 | Wahid Haidari | 36 |

Content of the Table Test after insertions:

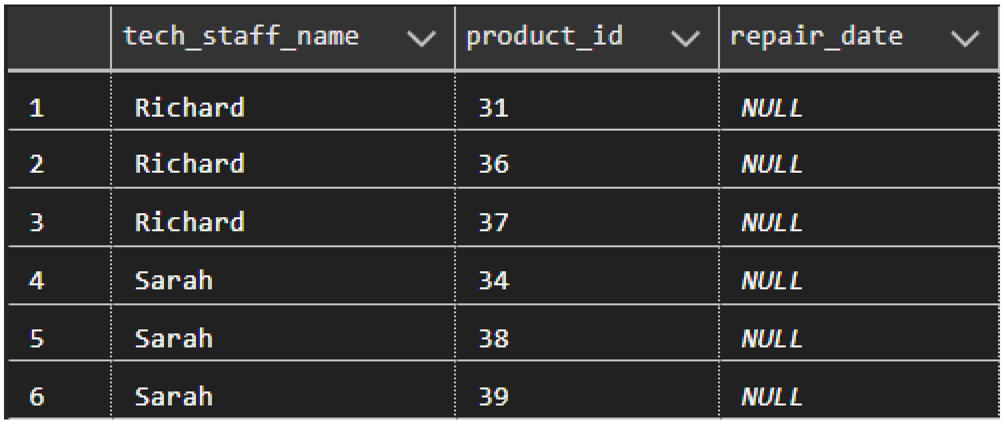
SELECT * FROM Test;



|  | qual_cont_name | product_id |
|---|---|---|
| 1 | David | 30 |
| 2 | David | 32 |
| 3 | David | 33 |
| 4 | John | 34 |
| 5 | John | 37 |
| 6 | John | 38 |
| 7 | John | 39 |
| 8 | Robert | 31 |
| 9 | Robert | 35 |
| 10 | Robert | 36 |

Content of the Table Repair after insertions:

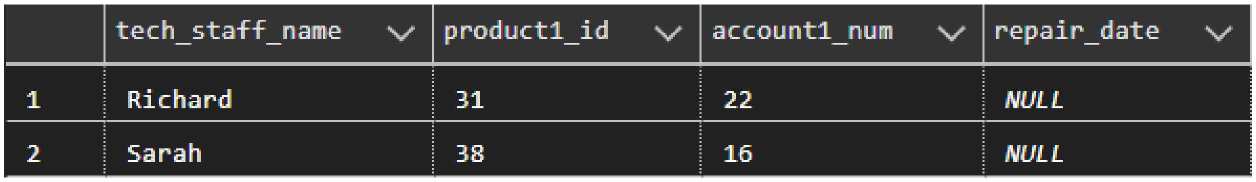
SELECT * FROM Repair;



|  | tech_staff_name | product_id | repair_date |
|---|---|---|---|
| 1 | Richard | 31 | *NULL* |
| 2 | Richard | 36 | *NULL* |
| 3 | Richard | 37 | *NULL* |
| 4 | Sarah | 34 | *NULL* |
| 5 | Sarah | 38 | *NULL* |
| 6 | Sarah | 39 | *NULL* |

Content of the Table RepairP1 after insertions:

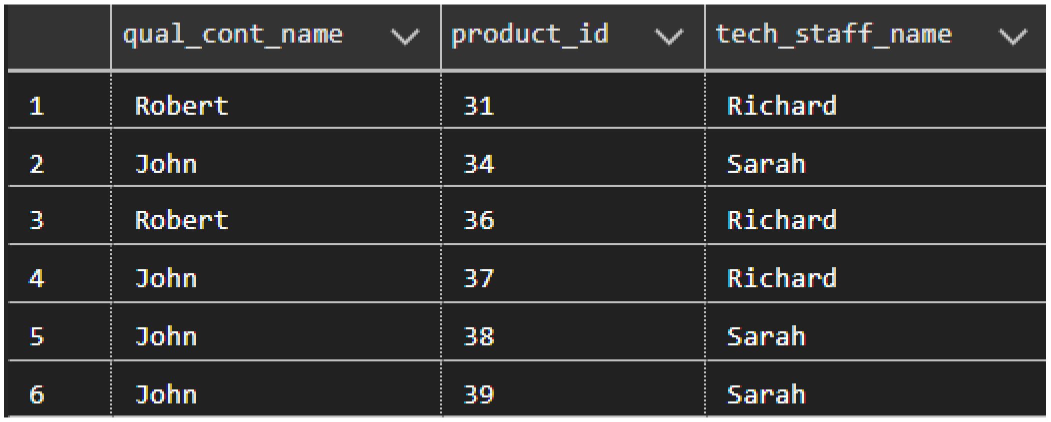
SELECT * FROM RepairP1;



| | tech_staff_name | product1_id | account1_num | repair_date |
|---|---|---|---|---|
| 1 | Richard | 31 | 22 | NULL |
| 2 | Sarah | 38 | 16 | NULL |

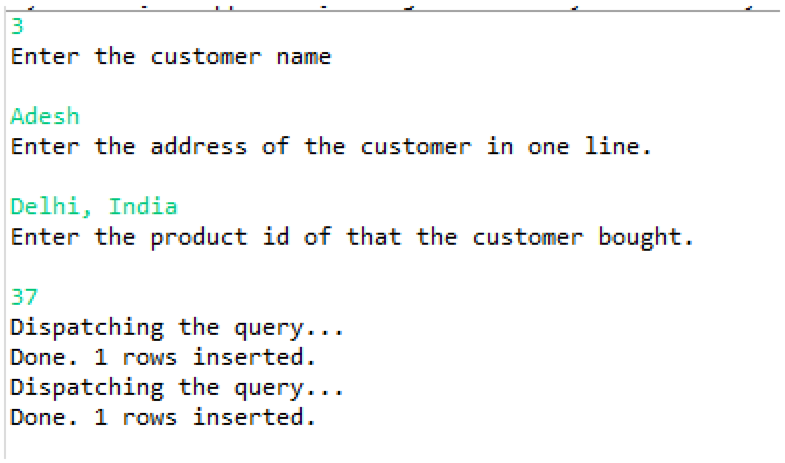
Content of the Table Request after insertions:

SELECT * FROM Request;

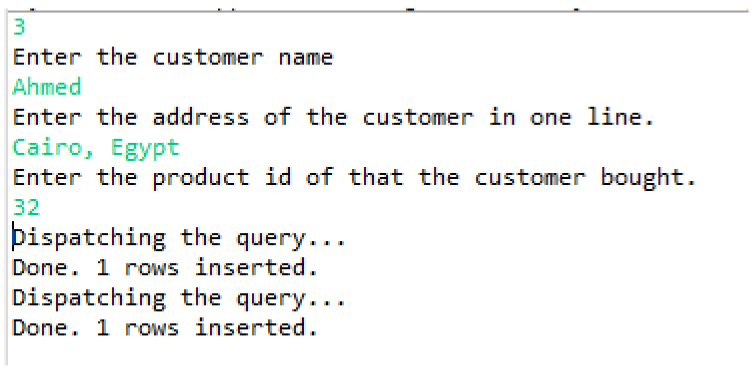


| | qual_cont_name | product_id | tech_staff_name |
|---|---|---|---|
| 1 | Robert | 31 | Richard |
| 2 | John | 34 | Sarah |
| 3 | Robert | 36 | Richard |
| 4 | John | 37 | Richard |
| 5 | John | 38 | Sarah |
| 6 | John | 39 | Sarah |

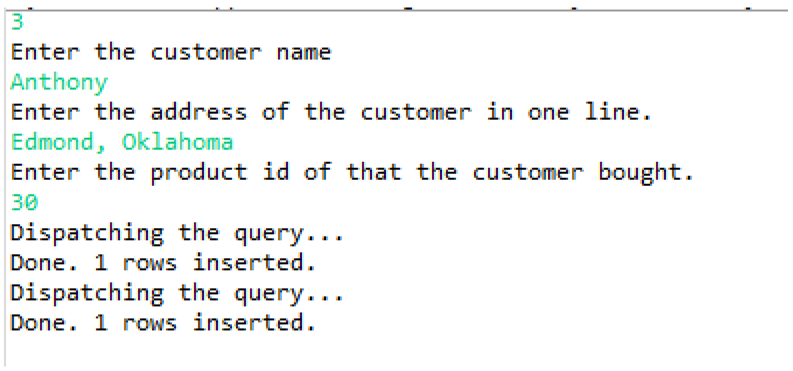
**6.3. Screenshots showing the testing of query 3**



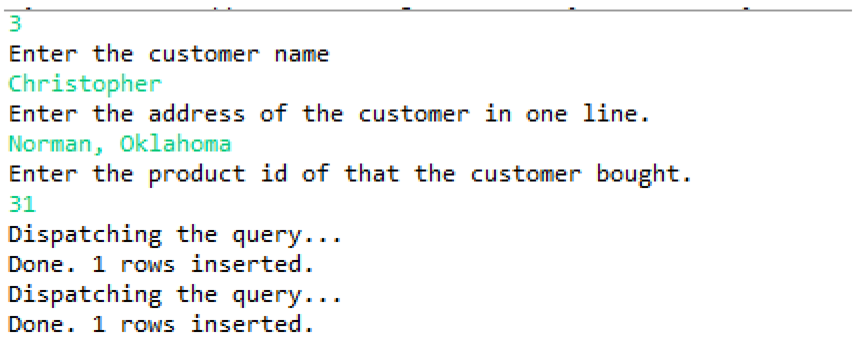
```
3
Enter the customer name

Adesh
Enter the address of the customer in one line.

Delhi, India
Enter the product id of that the customer bought.

37
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



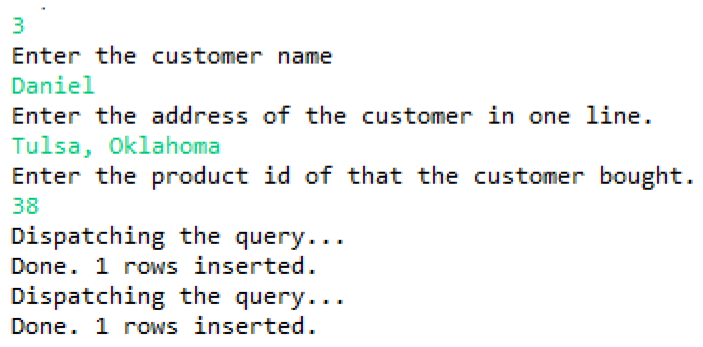
```
3
Enter the customer name
Ahmed
Enter the address of the customer in one line.
Cairo, Egypt
Enter the product id of that the customer bought.
32
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



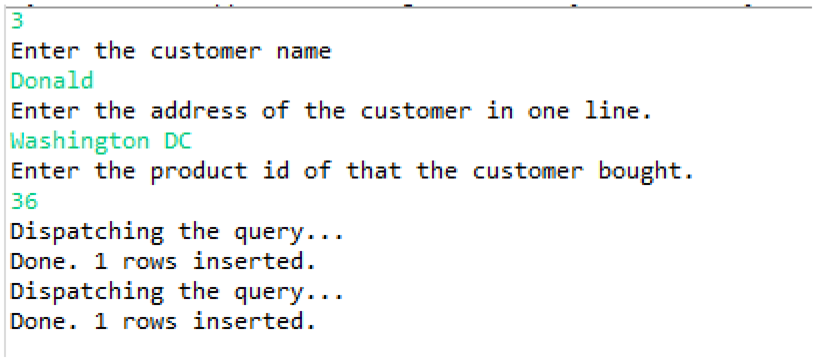
```
3
Enter the customer name
Anthony
Enter the address of the customer in one line.
Edmond, Oklahoma
Enter the product id of that the customer bought.
30
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



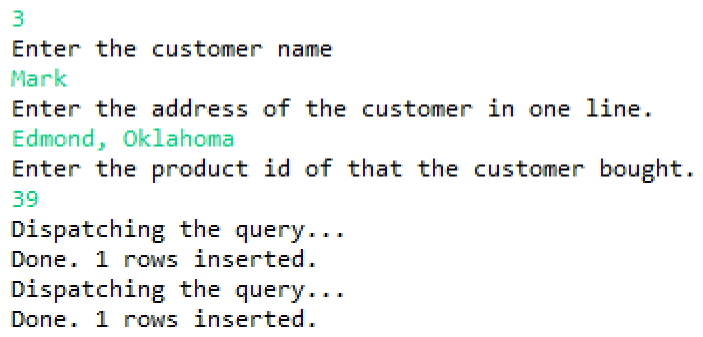
```
3
Enter the customer name
Christopher
Enter the address of the customer in one line.
Norman, Oklahoma
Enter the product id of that the customer bought.
31
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



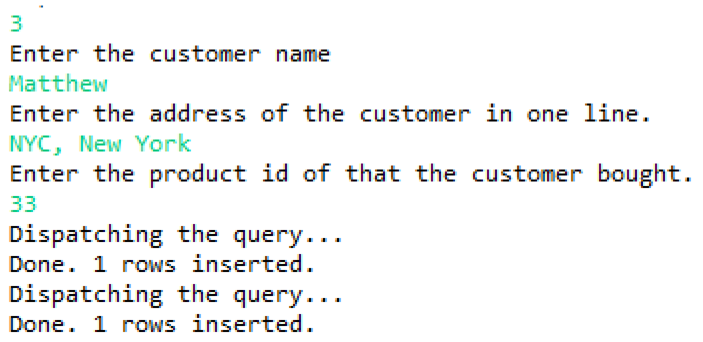
```
3
Enter the customer name
Daniel
Enter the address of the customer in one line.
Tulsa, Oklahoma
Enter the product id of that the customer bought.
38
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



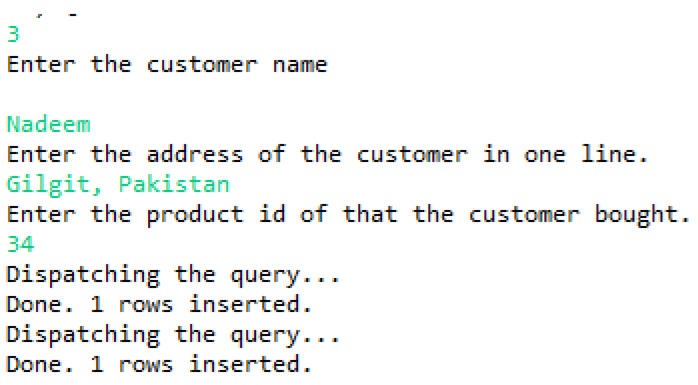
```
3
Enter the customer name
Donald
Enter the address of the customer in one line.
Washington DC
Enter the product id of that the customer bought.
36
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



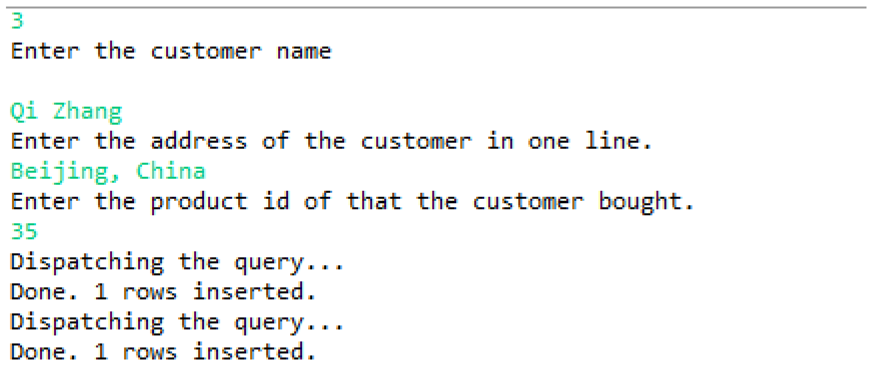
```
3
Enter the customer name
Mark
Enter the address of the customer in one line.
Edmond, Oklahoma
Enter the product id of that the customer bought.
39
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



```
3
Enter the customer name
Matthew
Enter the address of the customer in one line.
NYC, New York
Enter the product id of that the customer bought.
33
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



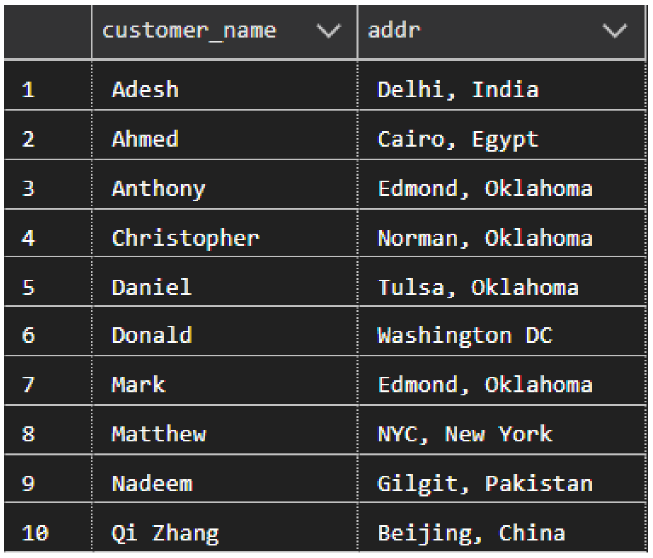
```
3
Enter the customer name

Nadeem
Enter the address of the customer in one line.
Gilgit, Pakistan
Enter the product id of that the customer bought.
34
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



```
3
Enter the customer name

Qi Zhang
Enter the address of the customer in one line.
Beijing, China
Enter the product id of that the customer bought.
35
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Customer after insertions:

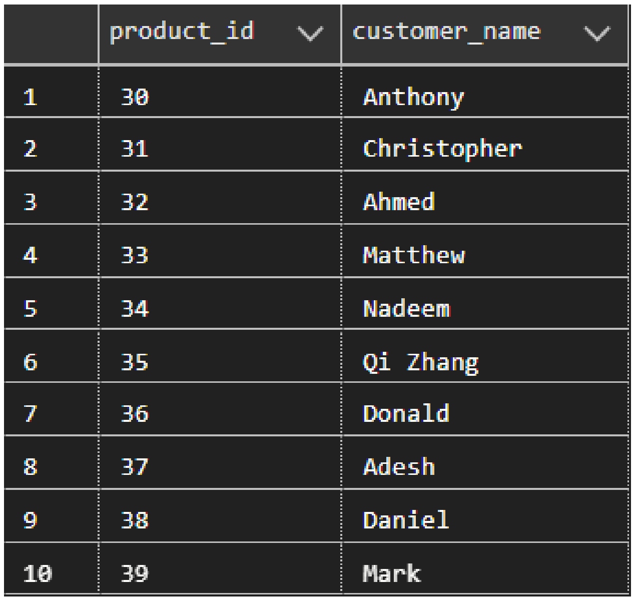
SELECT * FROM Customer;



| | customer_name | addr |
|---|---|---|
| 1 | Adesh | Delhi, India |
| 2 | Ahmed | Cairo, Egypt |
| 3 | Anthony | Edmond, Oklahoma |
| 4 | Christopher | Norman, Oklahoma |
| 5 | Daniel | Tulsa, Oklahoma |
| 6 | Donald | Washington DC |
| 7 | Mark | Edmond, Oklahoma |
| 8 | Matthew | NYC, New York |
| 9 | Nadeem | Gilgit, Pakistan |
| 10 | Qi Zhang | Beijing, China |

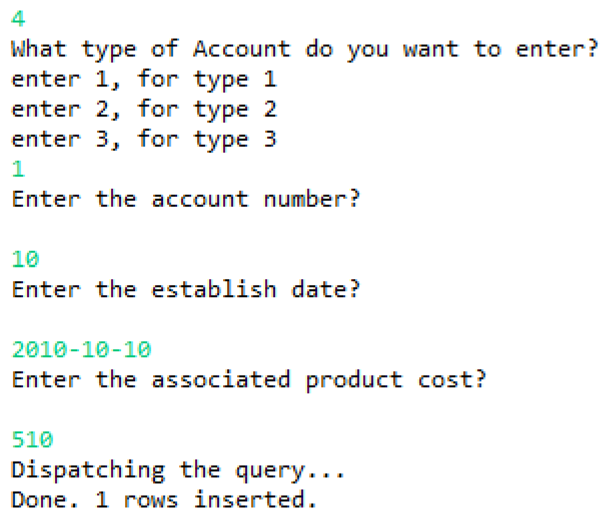
Content of the Table Purchase after insertions:

SELECT * FROM Purchase;

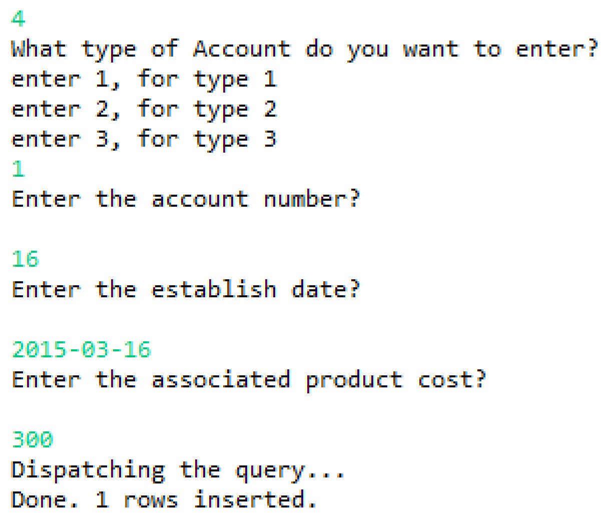


| | product_id | customer_name |
|---|---|---|
| 1 | 30 | Anthony |
| 2 | 31 | Christopher |
| 3 | 32 | Ahmed |
| 4 | 33 | Matthew |
| 5 | 34 | Nadeem |
| 6 | 35 | Qi Zhang |
| 7 | 36 | Donald |
| 8 | 37 | Adesh |
| 9 | 38 | Daniel |
| 10 | 39 | Mark |

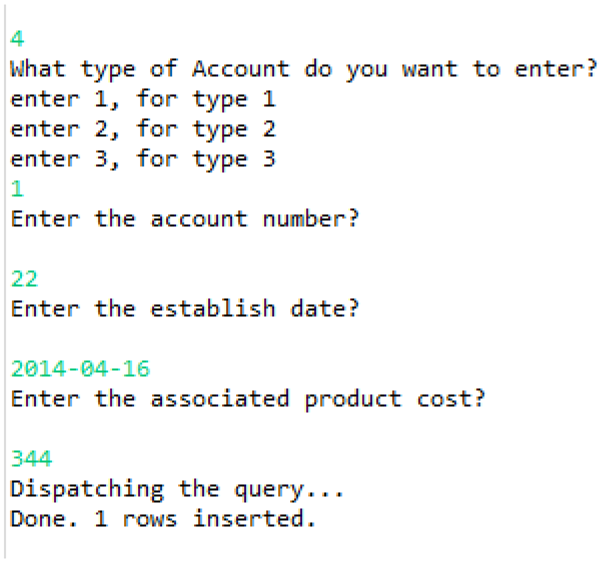
### 6.4. Screenshots showing the testing of query 4



```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
1
Enter the account number?

10
Enter the establish date?

2010-10-10
Enter the associated product cost?

510
Dispatching the query...
Done. 1 rows inserted.
```



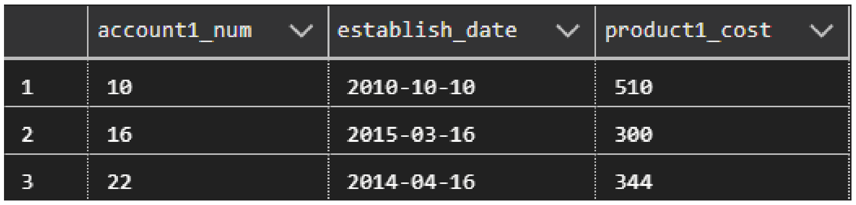
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
1
Enter the account number?

16
Enter the establish date?

2015-03-16
Enter the associated product cost?

300
Dispatching the query...
Done. 1 rows inserted.
```



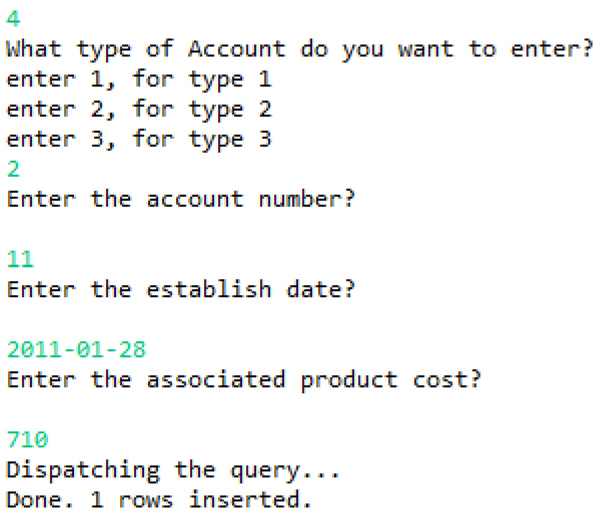
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
1
Enter the account number?

22
Enter the establish date?

2014-04-16
Enter the associated product cost?

344
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Account1 after insertions:

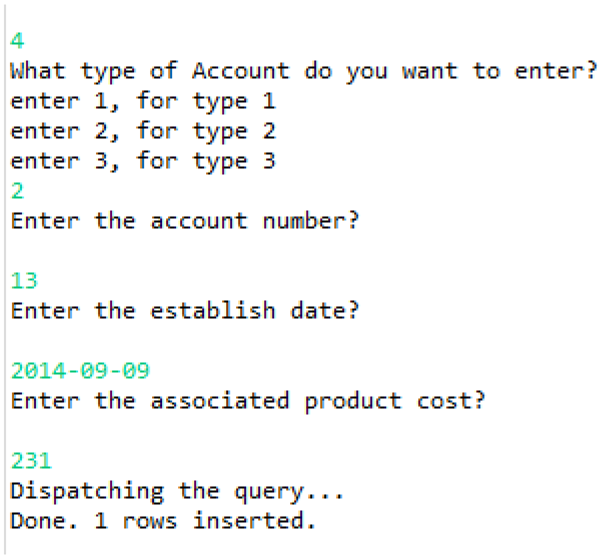
SELECT * FROM Account1;



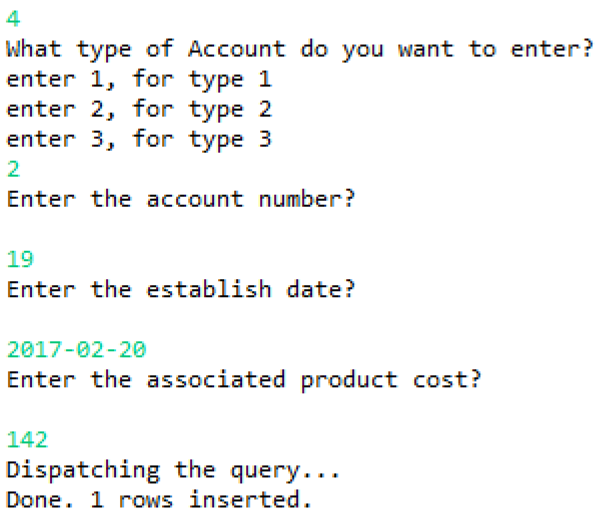
| | account1_num | establish_date | product1_cost |
|---|---|---|---|
| 1 | 10 | 2010-10-10 | 510 |
| 2 | 16 | 2015-03-16 | 300 |
| 3 | 22 | 2014-04-16 | 344 |



```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
2
Enter the account number?

11
Enter the establish date?

2011-01-28
Enter the associated product cost?

710
Dispatching the query...
Done. 1 rows inserted.
```



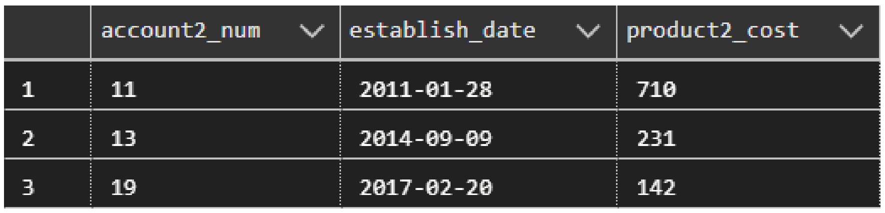
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
2
Enter the account number?

13
Enter the establish date?

2014-09-09
Enter the associated product cost?

231
Dispatching the query...
Done. 1 rows inserted.
```



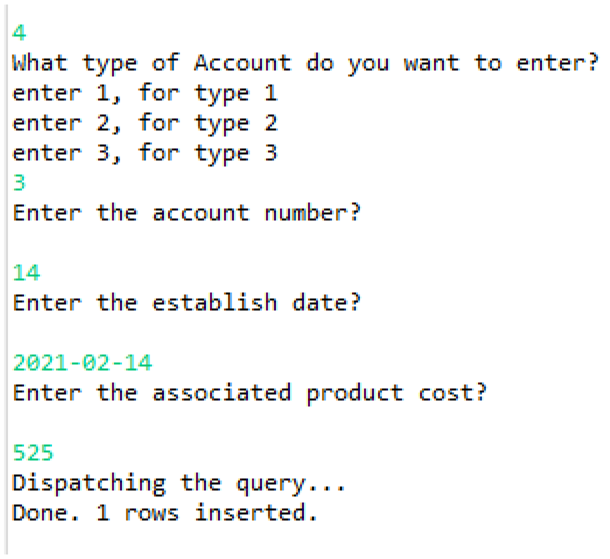
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
2
Enter the account number?

19
Enter the establish date?

2017-02-20
Enter the associated product cost?

142
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Account2 after insertions:

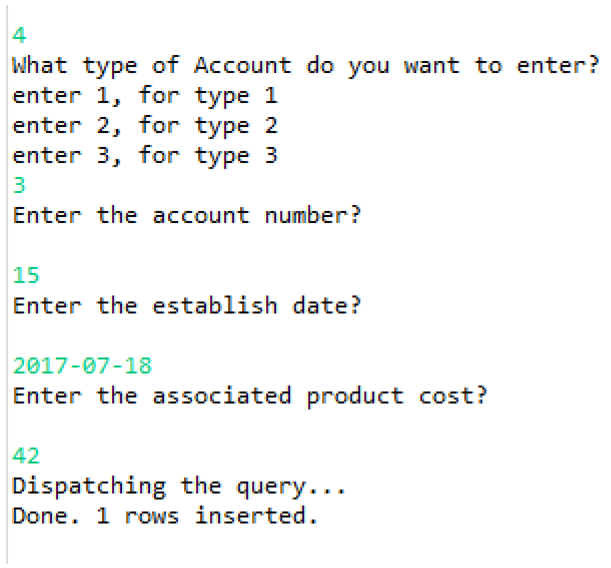
SELECT * FROM Account2;



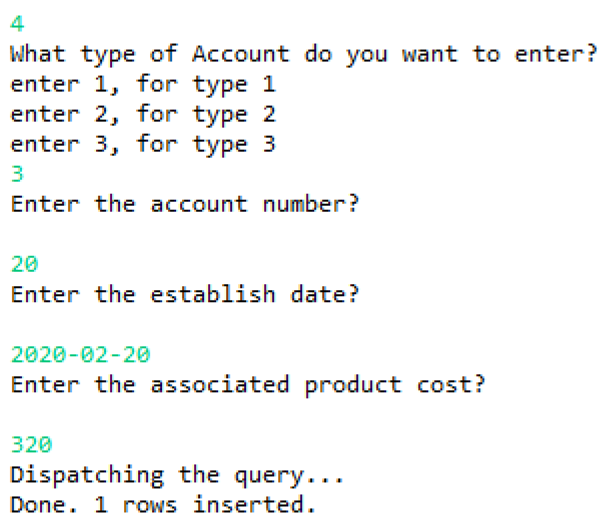
| | account2_num | establish_date | product2_cost |
|---|---|---|---|
| 1 | 11 | 2011-01-28 | 710 |
| 2 | 13 | 2014-09-09 | 231 |
| 3 | 19 | 2017-02-20 | 142 |



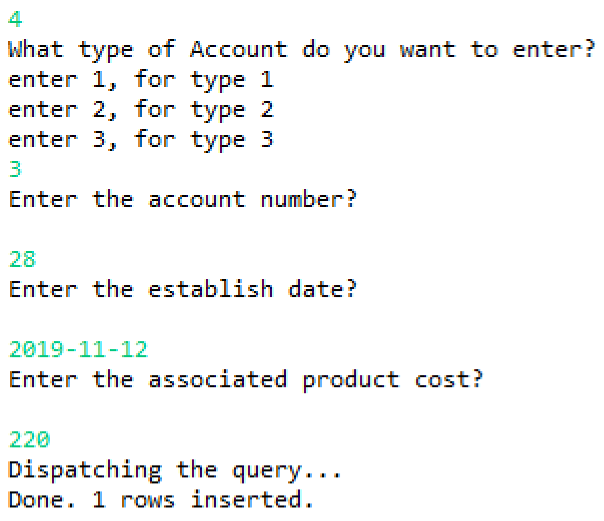
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
3
Enter the account number?

14
Enter the establish date?

2021-02-14
Enter the associated product cost?

525
Dispatching the query...
Done. 1 rows inserted.
```



```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
3
Enter the account number?

15
Enter the establish date?

2017-07-18
Enter the associated product cost?

42
Dispatching the query...
Done. 1 rows inserted.
```



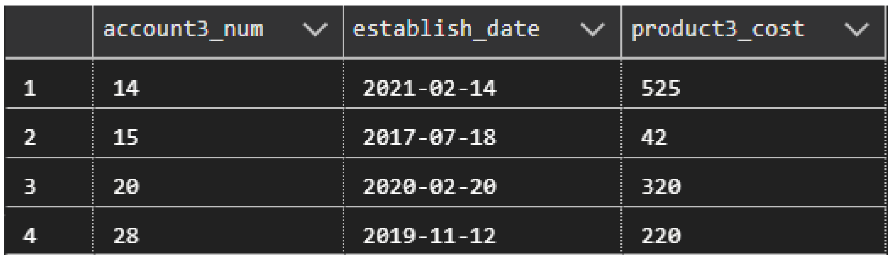
```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
3
Enter the account number?

20
Enter the establish date?

2020-02-20
Enter the associated product cost?

320
Dispatching the query...
Done. 1 rows inserted.
```



```
4
What type of Account do you want to enter?
enter 1, for type 1
enter 2, for type 2
enter 3, for type 3
3
Enter the account number?

28
Enter the establish date?

2019-11-12
Enter the associated product cost?

220
Dispatching the query...
Done. 1 rows inserted.
```

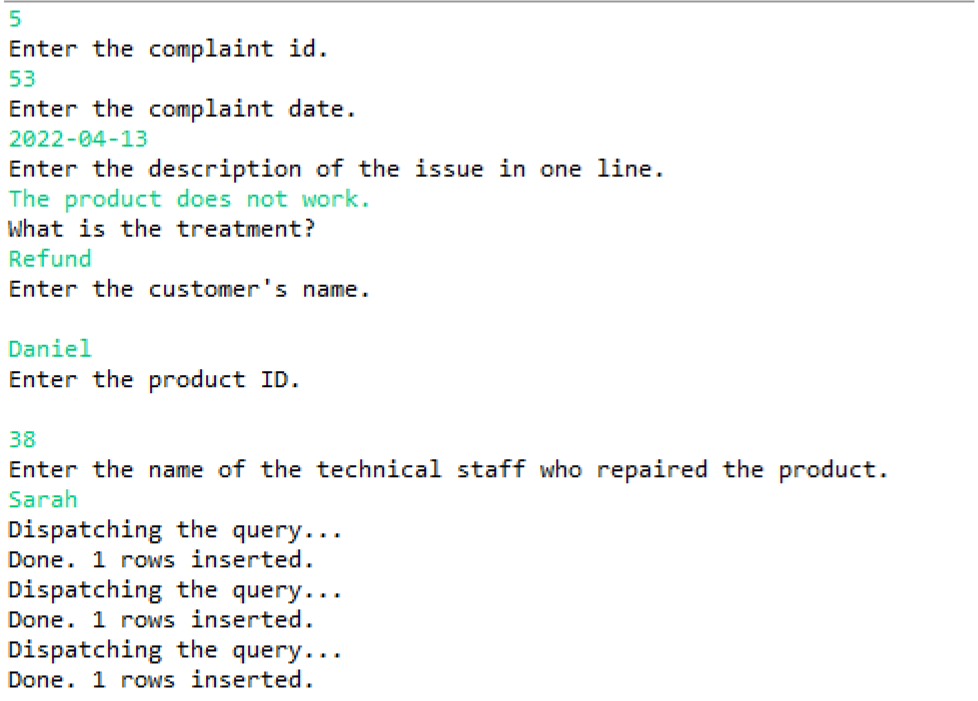
Content of the Table Account3 after insertions:

SELECT * FROM Account3;

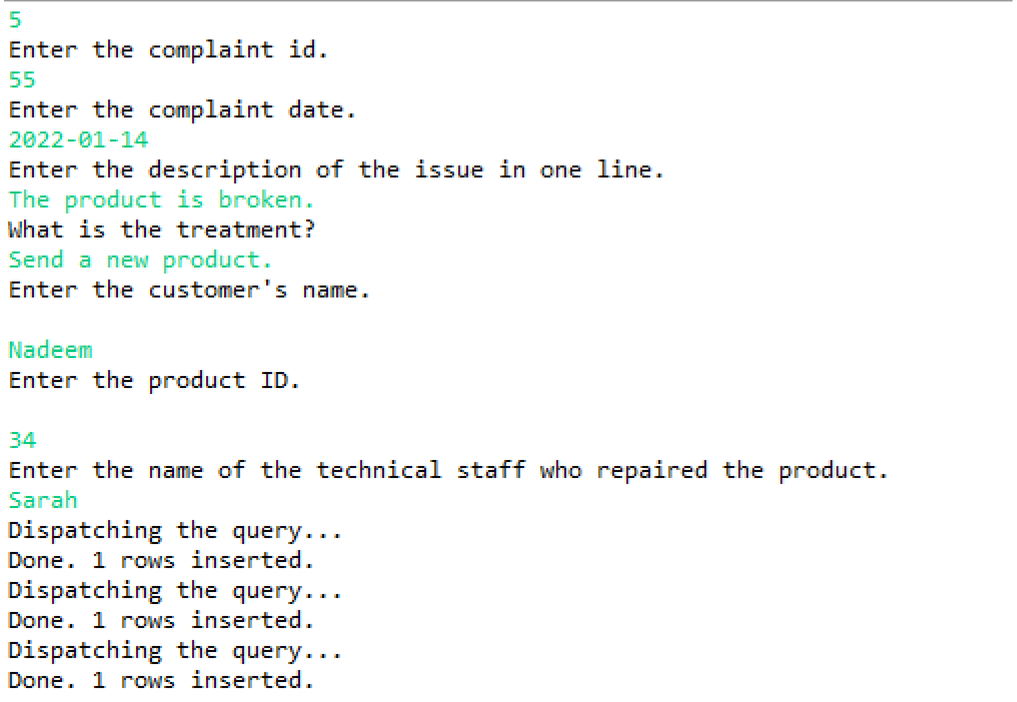


| | account3_num | establish_date | product3_cost |
|---|---|---|---|
| 1 | 14 | 2021-02-14 | 525 |
| 2 | 15 | 2017-07-18 | 42 |
| 3 | 20 | 2020-02-20 | 320 |
| 4 | 28 | 2019-11-12 | 220 |

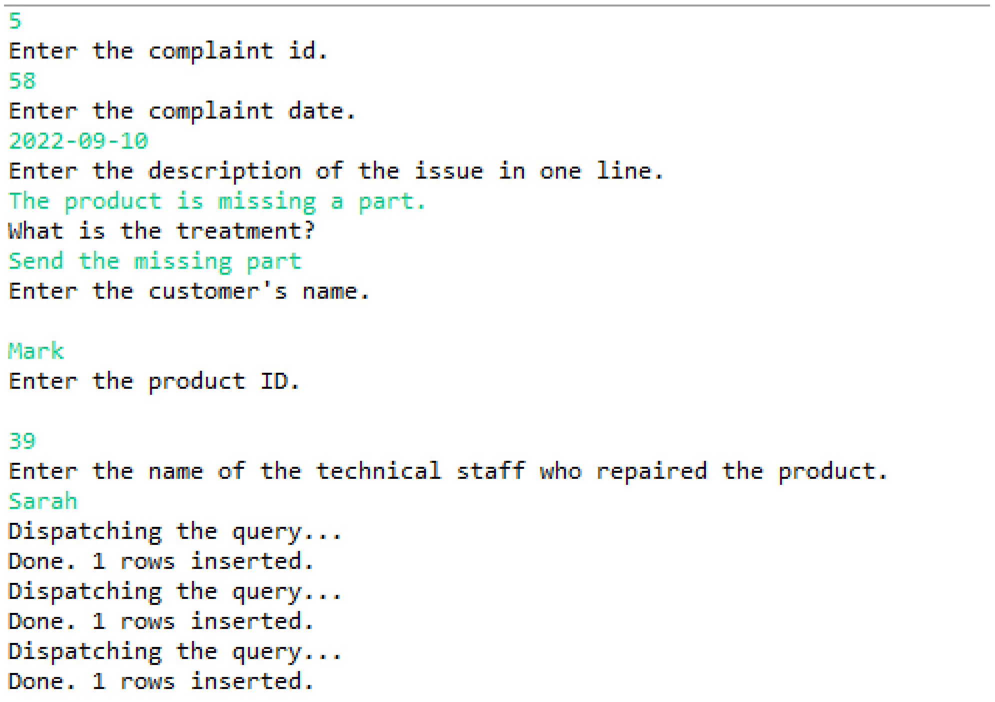
### 6.5. Screenshots showing the testing of query 5



```
5
Enter the complaint id.
53
Enter the complaint date.
2022-04-13
Enter the description of the issue in one line.
The product does not work.
What is the treatment?
Refund
Enter the customer's name.

Daniel
Enter the product ID.

38
Enter the name of the technical staff who repaired the product.
Sarah
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



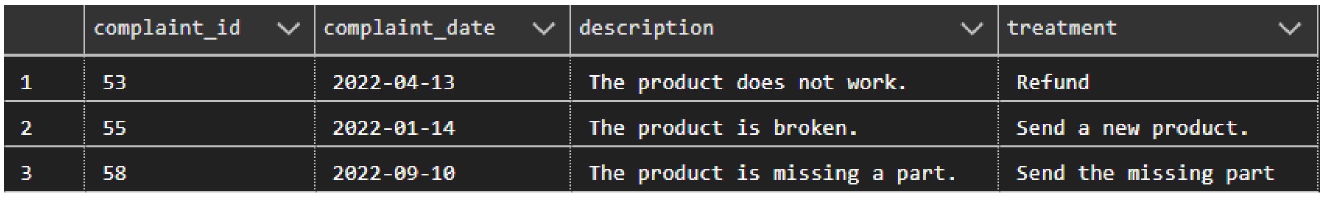
```
5
Enter the complaint id.
55
Enter the complaint date.
2022-01-14
Enter the description of the issue in one line.
The product is broken.
What is the treatment?
Send a new product.
Enter the customer's name.

Nadeem
Enter the product ID.

34
Enter the name of the technical staff who repaired the product.
Sarah
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```



```
5
Enter the complaint id.
58
Enter the complaint date.
2022-09-10
Enter the description of the issue in one line.
The product is missing a part.
What is the treatment?
Send the missing part
Enter the customer's name.

Mark
Enter the product ID.

39
Enter the name of the technical staff who repaired the product.
Sarah
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

Content of the Table Complaint after insertions:

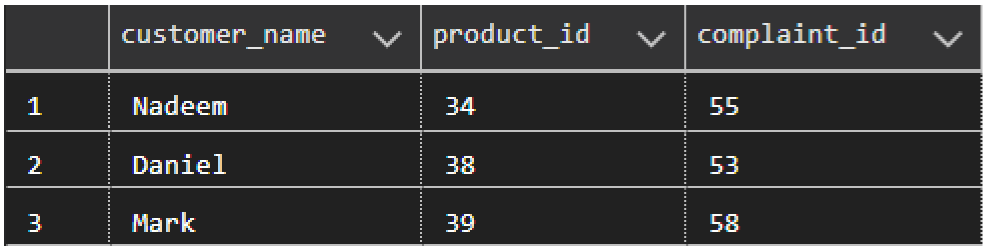
SELECT * FROM Complaint;



| | complaint_id | complaint_date | description | treatment |
|---|---|---|---|---|
| 1 | 53 | 2022-04-13 | The product does not work. | Refund |
| 2 | 55 | 2022-01-14 | The product is broken. | Send a new product. |
| 3 | 58 | 2022-09-10 | The product is missing a part. | Send the missing part |

Content of the Table Make after insertions:

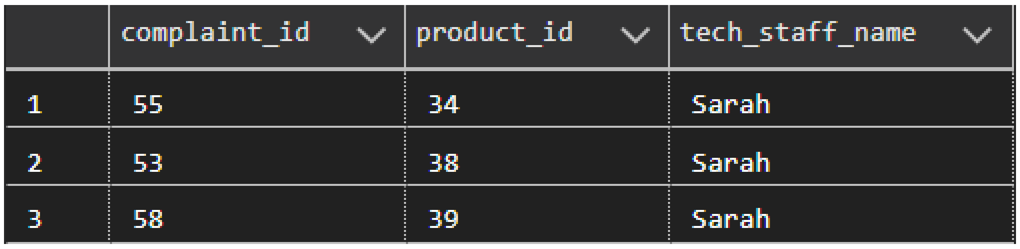
SELECT * FROM Make;



| | customer_name | product_id | complaint_id |
|---|---|---|---|
| 1 | Nadeem | 34 | 55 |
| 2 | Daniel | 38 | 53 |
| 3 | Mark | 39 | 58 |

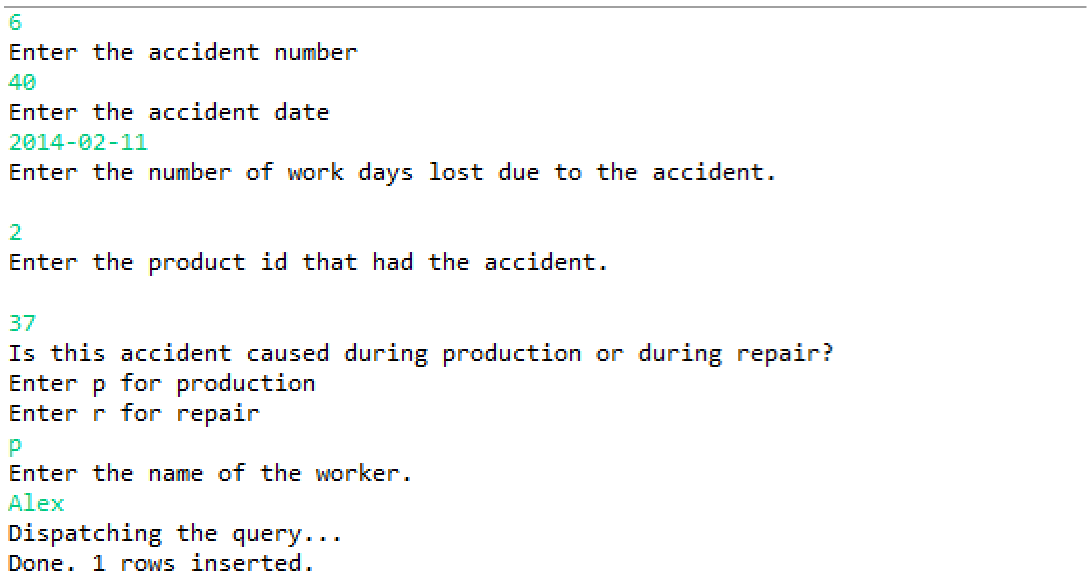
Content of the Table Cot after insertions:

SELECT * FROM Got;

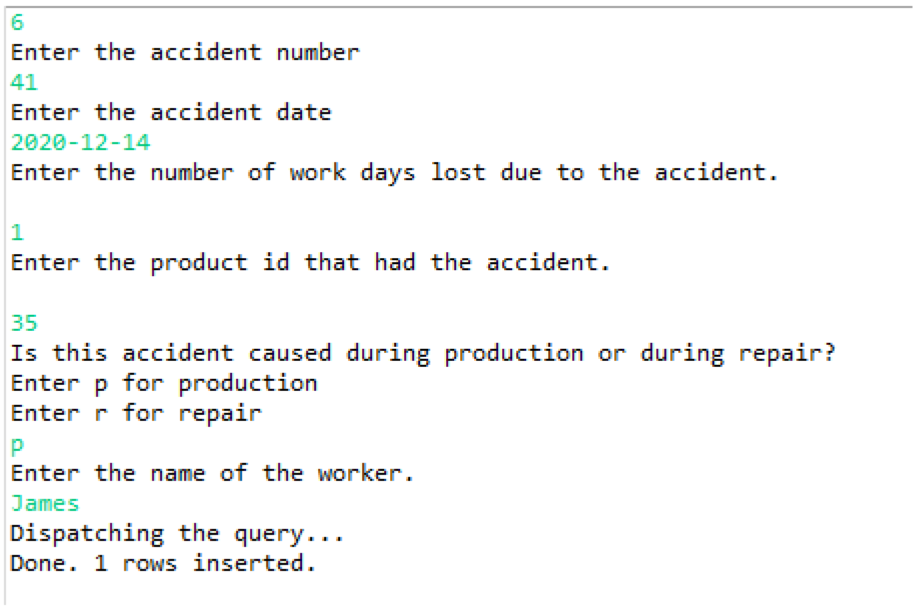


| | complaint_id | product_id | tech_staff_name |
|---|---|---|---|
| 1 | 55 | 34 | Sarah |
| 2 | 53 | 38 | Sarah |
| 3 | 58 | 39 | Sarah |

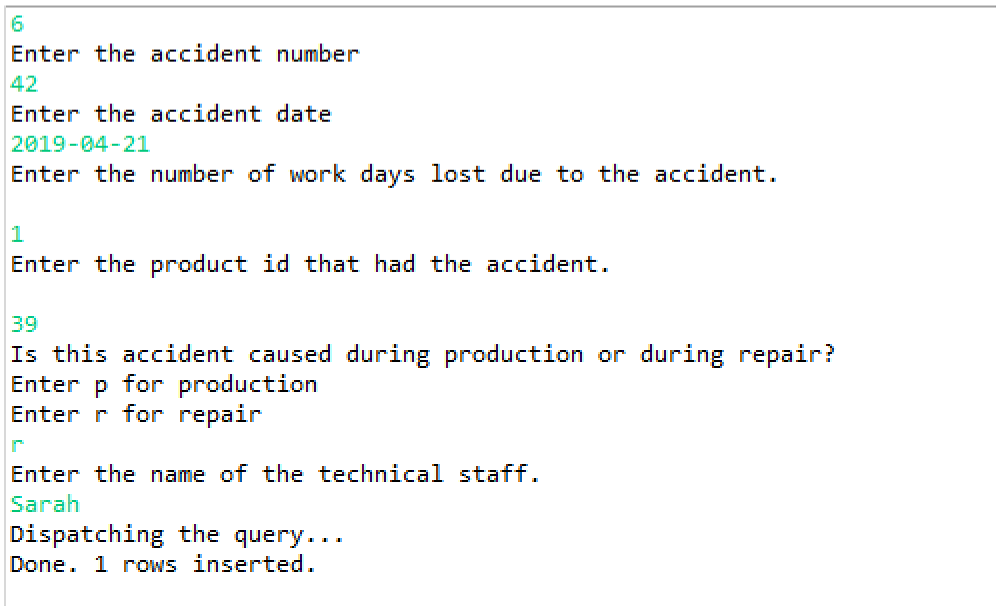
### 6.6. Screenshots showing the testing of query 6



```
6
Enter the accident number
40
Enter the accident date
2014-02-11
Enter the number of work days lost due to the accident.

2
Enter the product id that had the accident.

37
Is this accident caused during production or during repair?
Enter p for production
Enter r for repair
p
Enter the name of the worker.
Alex
Dispatching the query...
Done. 1 rows inserted.
```

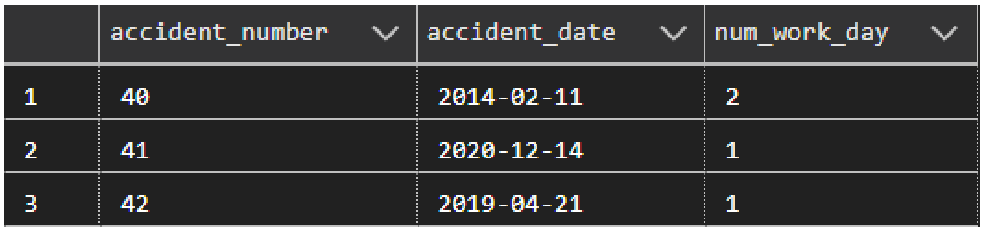


```
6
Enter the accident number
41
Enter the accident date
2020-12-14
Enter the number of work days lost due to the accident.

1
Enter the product id that had the accident.

35
Is this accident caused during production or during repair?
Enter p for production
Enter r for repair
p
Enter the name of the worker.
James
Dispatching the query...
Done. 1 rows inserted.
```



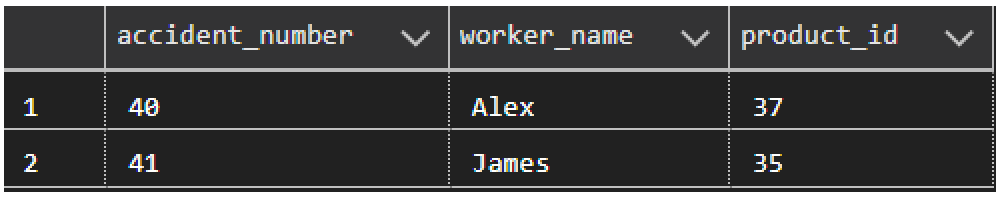
```
6
Enter the accident number
42
Enter the accident date
2019-04-21
Enter the number of work days lost due to the accident.

1
Enter the product id that had the accident.

39
Is this accident caused during production or during repair?
Enter p for production
Enter r for repair
r
Enter the name of the technical staff.
Sarah
Dispatching the query...
Done. 1 rows inserted.
```

SELECT * FROM Accident;



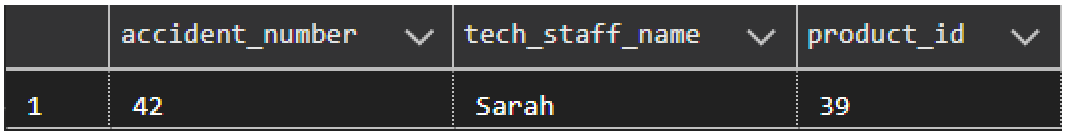
| | accident_number | accident_date | num_work_day |
|---|---|---|---|
| 1 | 40 | 2014-02-11 | 2 |
| 2 | 41 | 2020-12-14 | 1 |
| 3 | 42 | 2019-04-21 | 1 |

SELECT * FROM ProduceCause;



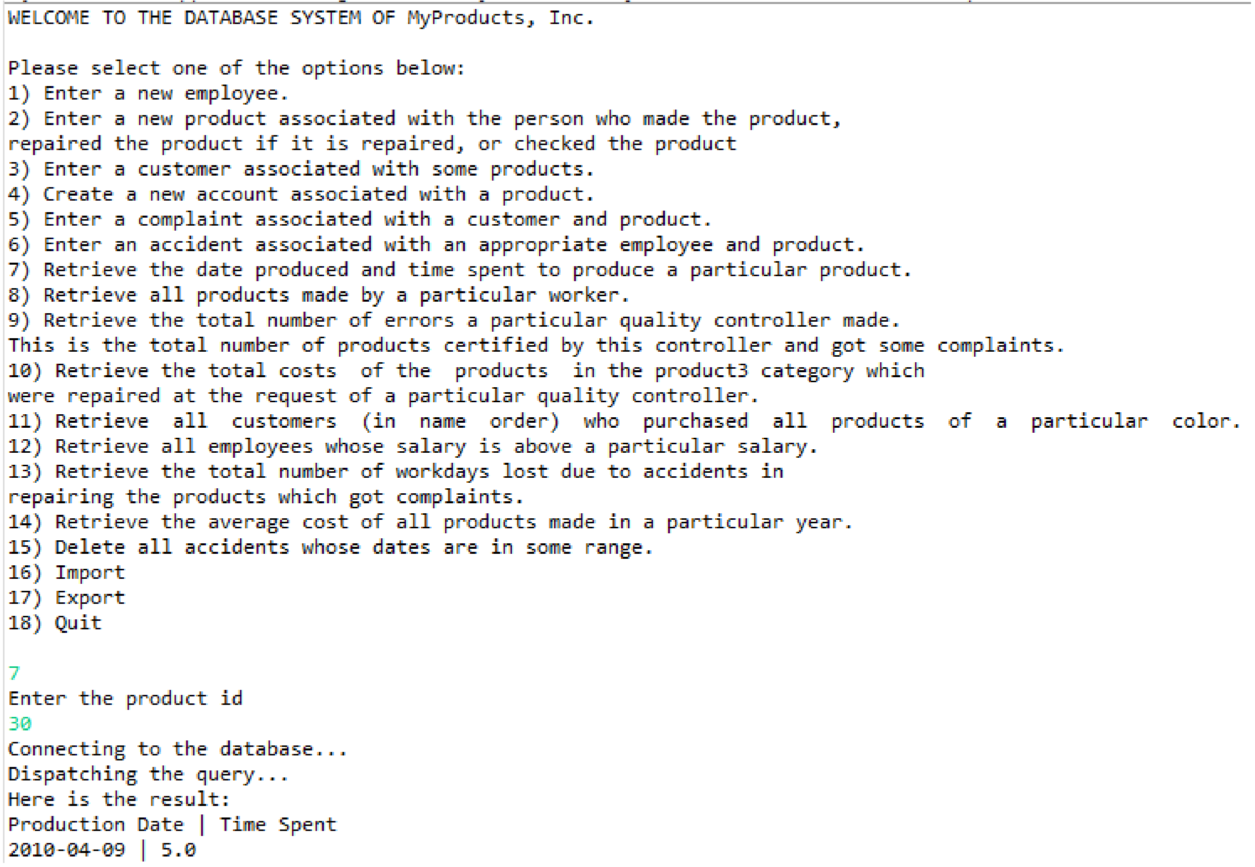
| | accident_number | worker_name | product_id |
|---|---|---|---|
| 1 | 40 | Alex | 37 |
| 2 | 41 | James | 35 |

SELECT * FROM RepairCause;

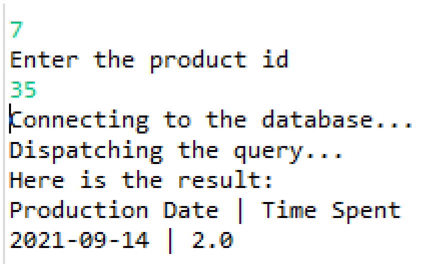


| | accident_number | tech_staff_name | product_id |
|---|---|---|---|
| 1 | 42 | Sarah | 39 |

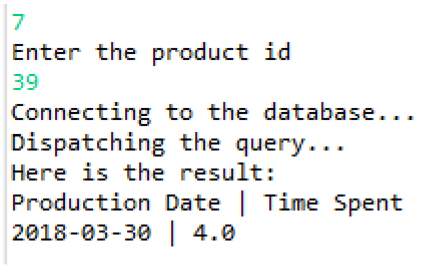
### 6.7. Screenshots showing the testing of query 7



```
WELCOME TO THE DATABASE SYSTEM OF MyProducts, Inc.

Please select one of the options below:
1) Enter a new employee.
2) Enter a new product associated with the person who made the product,
repaired the product if it is repaired, or checked the product
3) Enter a customer associated with some products.
4) Create a new account associated with a product.
5) Enter a complaint associated with a customer and product.
6) Enter an accident associated with an appropriate employee and product.
7) Retrieve the date produced and time spent to produce a particular product.
8) Retrieve all products made by a particular worker.
9) Retrieve the total number of errors a particular quality controller made.
This is the total number of products certified by this controller and got some complaints.
10) Retrieve the total costs  of the  products  in the product3 category which
were repaired at the request of a particular quality controller.
11) Retrieve  all  customers  (in  name  order)  who  purchased  all  products  of  a  particular  color.
12) Retrieve all employees whose salary is above a particular salary.
13) Retrieve the total number of workdays lost due to accidents in
repairing the products which got complaints.
14) Retrieve the average cost of all products made in a particular year.
15) Delete all accidents whose dates are in some range.
16) Import
17) Export
18) Quit

7
Enter the product id
30
Connecting to the database...
Dispatching the query...
Here is the result:
Production Date | Time Spent
2010-04-09 | 5.0
```

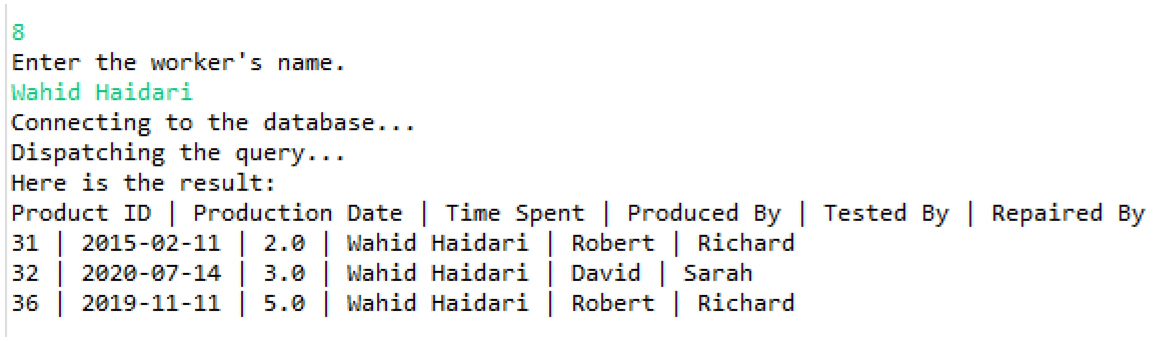


```
7
Enter the product id
35
Connecting to the database...
Dispatching the query...
Here is the result:
Production Date | Time Spent
2021-09-14 | 2.0
```

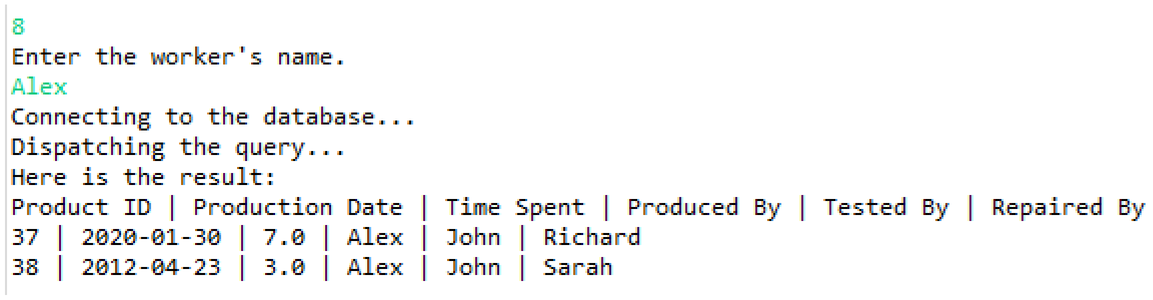


```
7
Enter the product id
39
Connecting to the database...
Dispatching the query...
Here is the result:
Production Date | Time Spent
2018-03-30 | 4.0
```

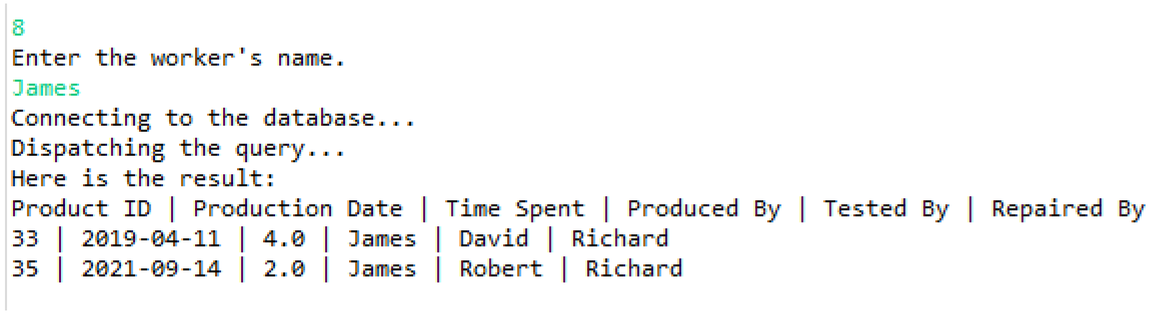
### 6.8. Screenshots showing the testing of query 8



```
8
Enter the worker's name.
Wahid Haidari
Connecting to the database...
Dispatching the query...
Here is the result:
Product ID | Production Date | Time Spent | Produced By | Tested By | Repaired By
31 | 2015-02-11 | 2.0 | Wahid Haidari | Robert | Richard
32 | 2020-07-14 | 3.0 | Wahid Haidari | David | Sarah
36 | 2019-11-11 | 5.0 | Wahid Haidari | Robert | Richard
```

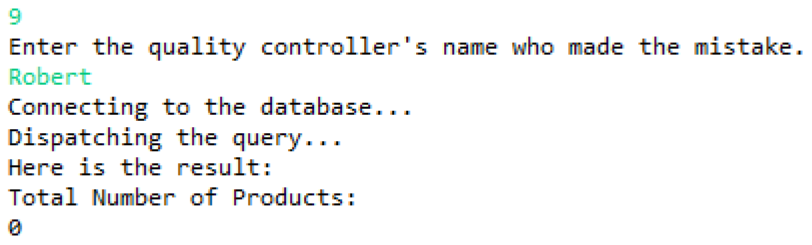


```
8
Enter the worker's name.
Alex
Connecting to the database...
Dispatching the query...
Here is the result:
Product ID | Production Date | Time Spent | Produced By | Tested By | Repaired By
37 | 2020-01-30 | 7.0 | Alex | John | Richard
38 | 2012-04-23 | 3.0 | Alex | John | Sarah
```

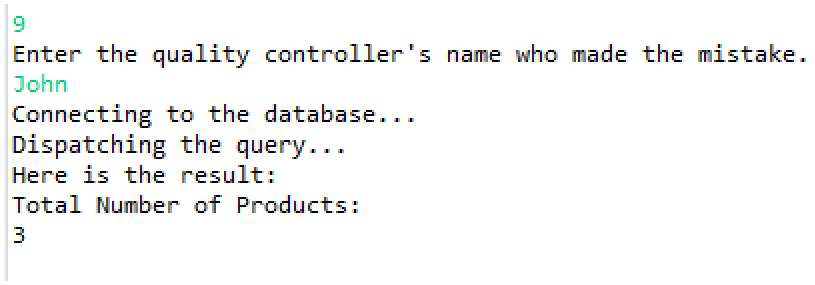


```
8
Enter the worker's name.
James
Connecting to the database...
Dispatching the query...
Here is the result:
Product ID | Production Date | Time Spent | Produced By | Tested By | Repaired By
33 | 2019-04-11 | 4.0 | James | David | Richard
35 | 2021-09-14 | 2.0 | James | Robert | Richard
```

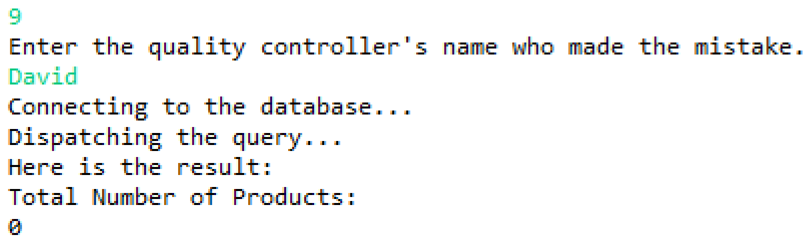
### 6.9. Screenshots showing the testing of query 9



```
9
Enter the quality controller's name who made the mistake.
Robert
Connecting to the database...
Dispatching the query...
Here is the result:
Total Number of Products:
0
```

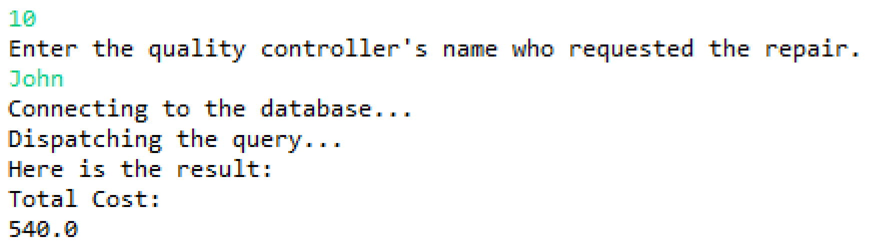


```
9
Enter the quality controller's name who made the mistake.
John
Connecting to the database...
Dispatching the query...
Here is the result:
Total Number of Products:
3
```

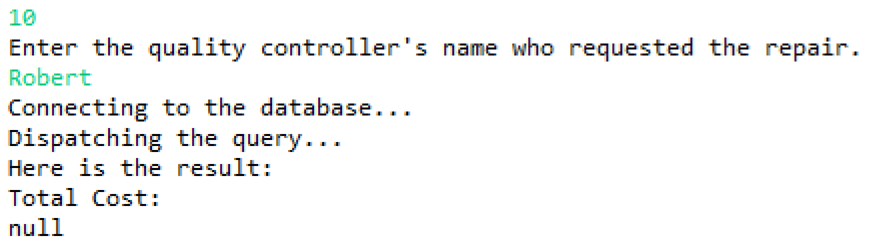


```
9
Enter the quality controller's name who made the mistake.
David
Connecting to the database...
Dispatching the query...
Here is the result:
Total Number of Products:
0
```

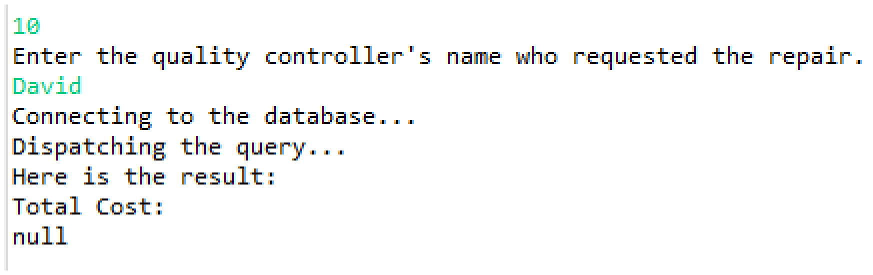
### 6.10. Screenshots showing the testing of query 10



```
10
Enter the quality controller's name who requested the repair.
John
Connecting to the database...
Dispatching the query...
Here is the result:
Total Cost:
540.0
```

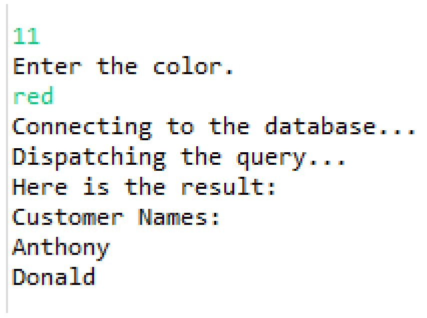


```
10
Enter the quality controller's name who requested the repair.
Robert
Connecting to the database...
Dispatching the query...
Here is the result:
Total Cost:
null
```

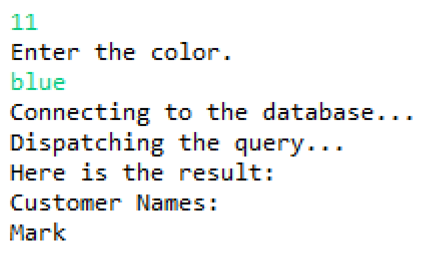


```
10
Enter the quality controller's name who requested the repair.
David
Connecting to the database...
Dispatching the query...
Here is the result:
Total Cost:
null
```

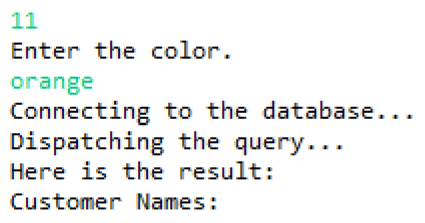
**6.11. Screenshots showing the testing of query 11**



```
11
Enter the color.
red
Connecting to the database...
Dispatching the query...
Here is the result:
Customer Names:
Anthony
Donald
```

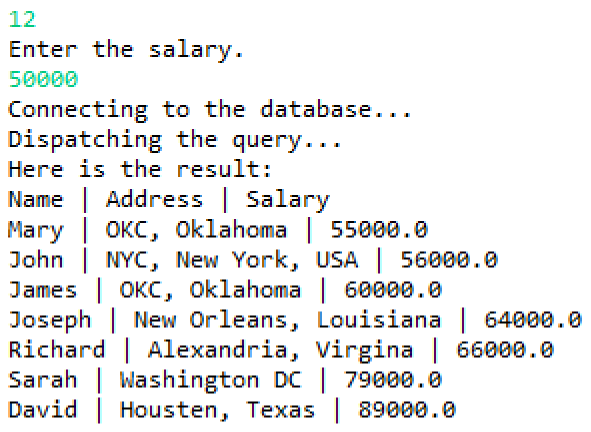


```
11
Enter the color.
blue
Connecting to the database...
Dispatching the query...
Here is the result:
Customer Names:
Mark
```



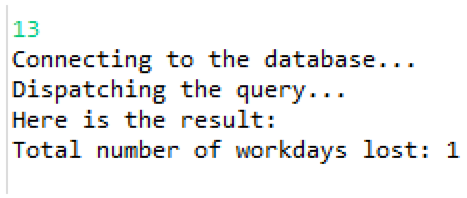
```
11
Enter the color.
orange
Connecting to the database...
Dispatching the query...
Here is the result:
Customer Names:
```

### 6.12. Screenshots showing the testing of query 12



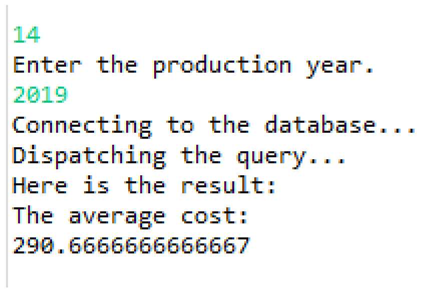
```
12
Enter the salary.
50000
Connecting to the database...
Dispatching the query...
Here is the result:
Name | Address | Salary
Mary | OKC, Oklahoma | 55000.0
John | NYC, New York, USA | 56000.0
James | OKC, Oklahoma | 60000.0
Joseph | New Orleans, Louisiana | 64000.0
Richard | Alexandria, Virgina | 66000.0
Sarah | Washington DC | 79000.0
David | Housten, Texas | 89000.0
```

### 6.13. Screenshots showing the testing of query 13



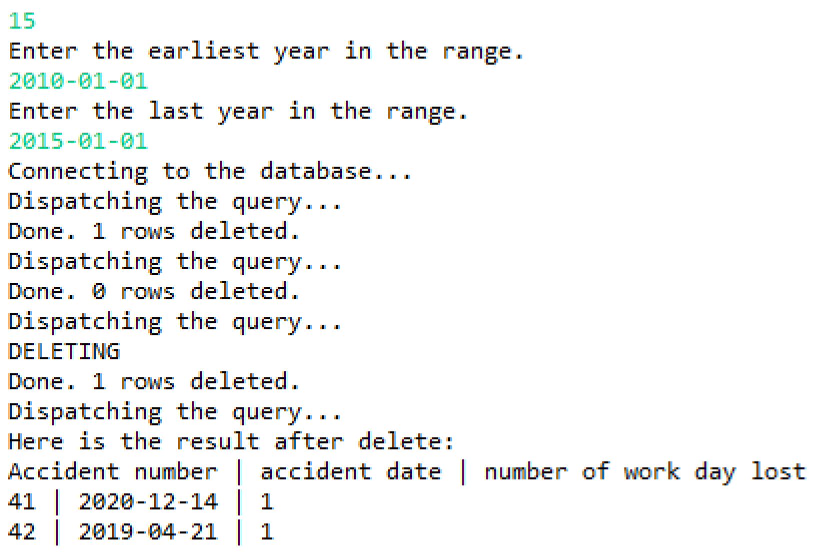
```
13
Connecting to the database...
Dispatching the query...
Here is the result:
Total number of workdays lost: 1
```

### 6.14. Screenshots showing the testing of query 14



```
14
Enter the production year.
2019
Connecting to the database...
Dispatching the query...
Here is the result:
The average cost:
290.6666666666667
```

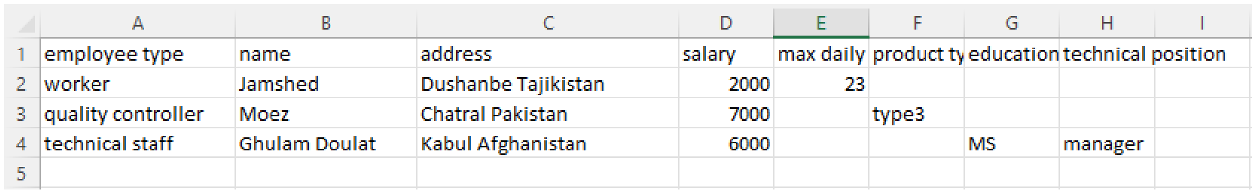
**6.15. Screenshots showing the testing of query 15**



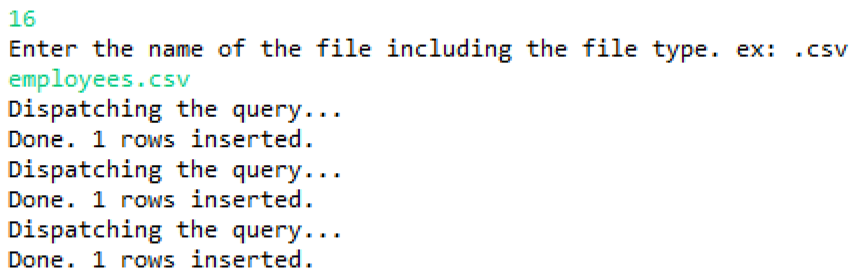
```
15
Enter the earliest year in the range.
2010-01-01
Enter the last year in the range.
2015-01-01
Connecting to the database...
Dispatching the query...
Done. 1 rows deleted.
Dispatching the query...
Done. 0 rows deleted.
Dispatching the query...
DELETING
Done. 1 rows deleted.
Dispatching the query...
Here is the result after delete:
Accident number | accident date | number of work day lost
41 | 2020-12-14 | 1
42 | 2019-04-21 | 1
```

**Screenshots showing the testing of the Import and Export options**

Content of the file employees.csv



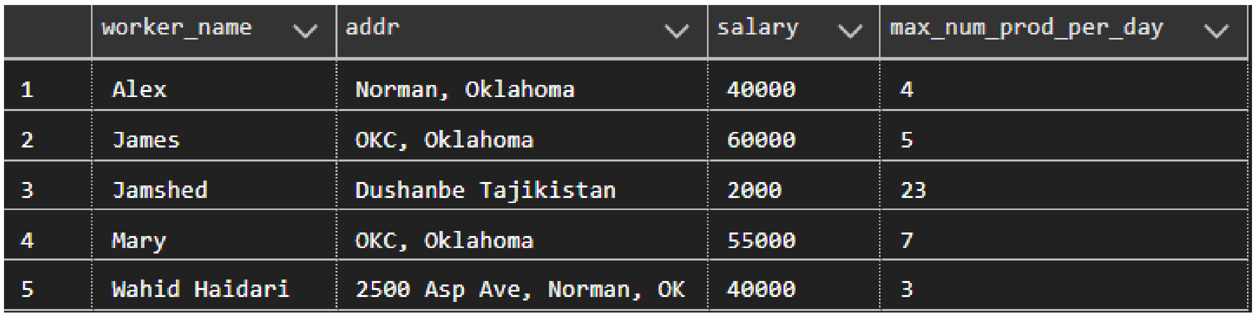
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | employee type | name | address | salary | max daily | product ty | education | technical | position |
| 2 | worker | Jamshed | Dushanbe Tajikistan | 2000 | 23 | | | | |
| 3 | quality controller | Moez | Chatral Pakistan | 7000 | | type3 | | | |
| 4 | technical staff | Ghulam Doulat | Kabul Afghanistan | 6000 | | | MS | manager | |
| 5 | | | | | | | | | |



```
16
Enter the name of the file including the file type. ex: .csv
employees.csv
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

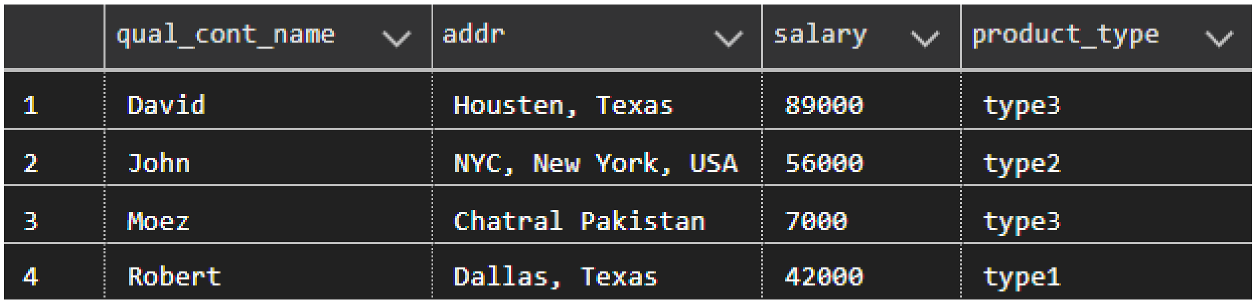
Content of the table after import:

SELECT * FROM Worker



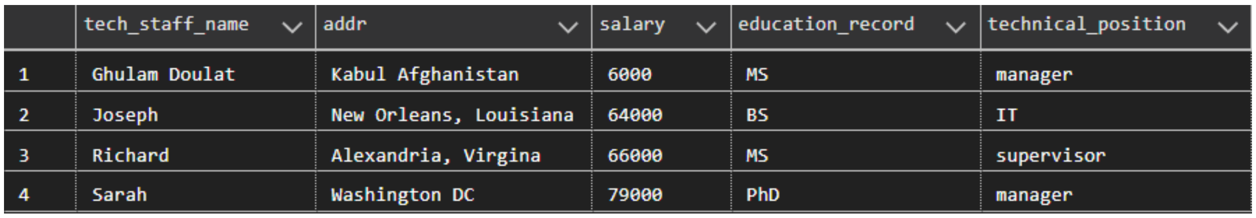
| | worker_name | addr | salary | max_num_prod_per_day |
|---|---|---|---|---|
| 1 | Alex | Norman, Oklahoma | 40000 | 4 |
| 2 | James | OKC, Oklahoma | 60000 | 5 |
| 3 | Jamshed | Dushanbe Tajikistan | 2000 | 23 |
| 4 | Mary | OKC, Oklahoma | 55000 | 7 |
| 5 | Wahid Haidari | 2500 Asp Ave, Norman, OK | 40000 | 3 |

SELECT * FROM QualityController



| | qual_cont_name | addr | salary | product_type |
|---|---|---|---|---|
| 1 | David | Housten, Texas | 89000 | type3 |
| 2 | John | NYC, New York, USA | 56000 | type2 |
| 3 | Moez | Chatral Pakistan | 7000 | type3 |
| 4 | Robert | Dallas, Texas | 42000 | type1 |

SELECT * FROM TechnicalStaff

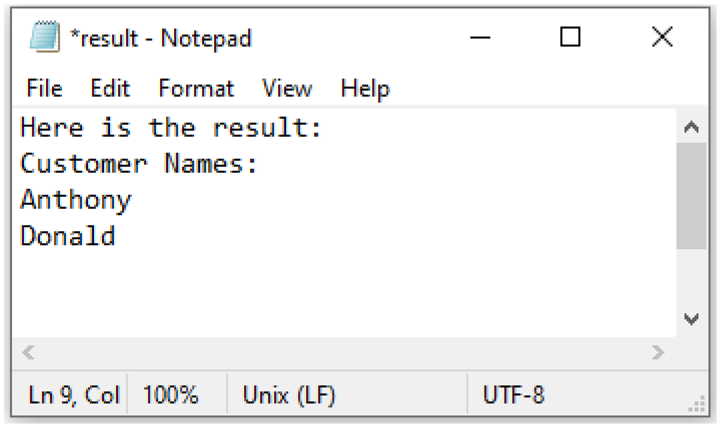


| | tech_staff_name | addr | salary | education_record | technical_position |
|---|---|---|---|---|---|
| 1 | Ghulam Doulat | Kabul Afghanistan | 6000 | MS | manager |
| 2 | Joseph | New Orleans, Louisiana | 64000 | BS | IT |
| 3 | Richard | Alexandria, Virgina | 66000 | MS | supervisor |
| 4 | Sarah | Washington DC | 79000 | PhD | manager |

**Testing Export**

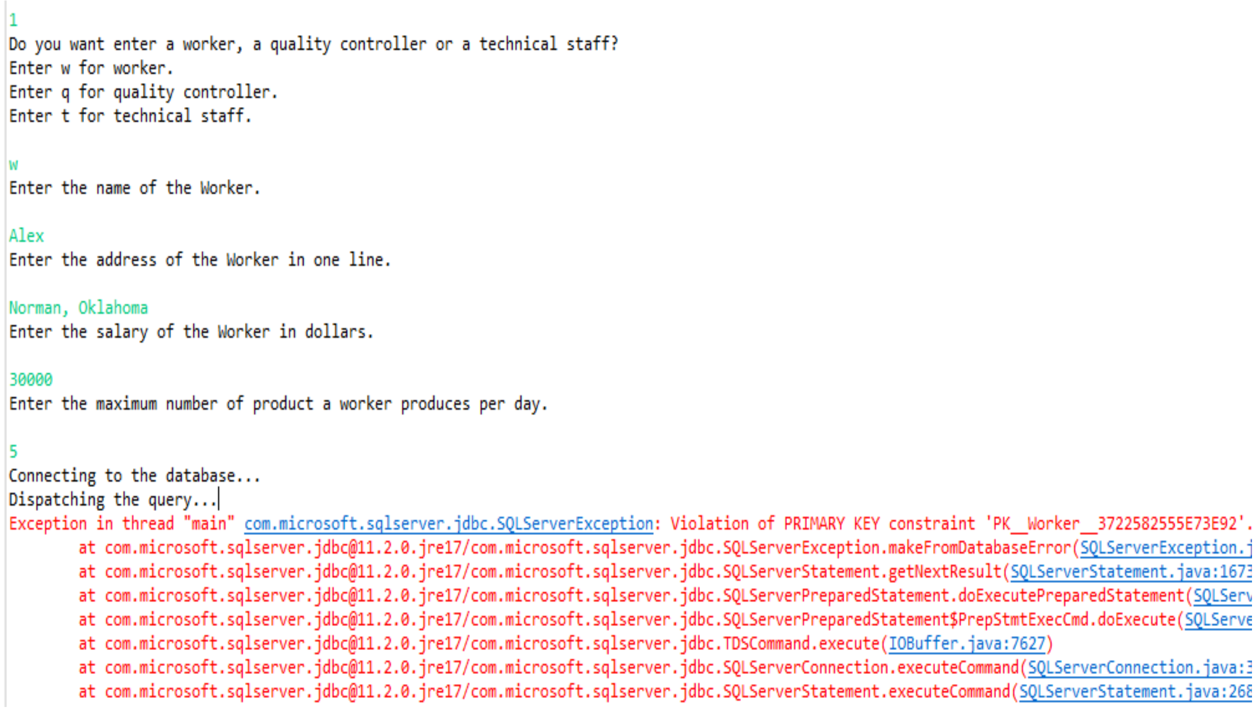
```
17
Enter the name of the file including the format.
result.txt
Enter the color.
red
Connecting to the database...
Dispatching the query...
```

Content of the exported file:

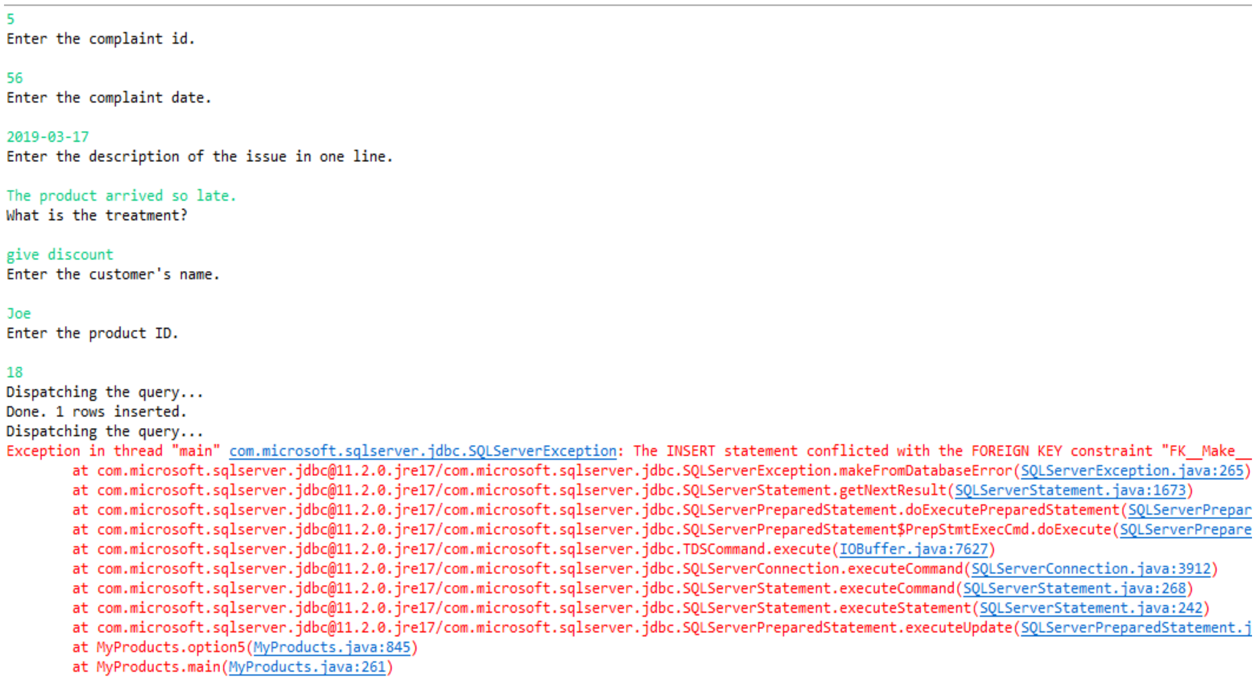


```
Here is the result:
Customer Names:
Anthony
Donald
```

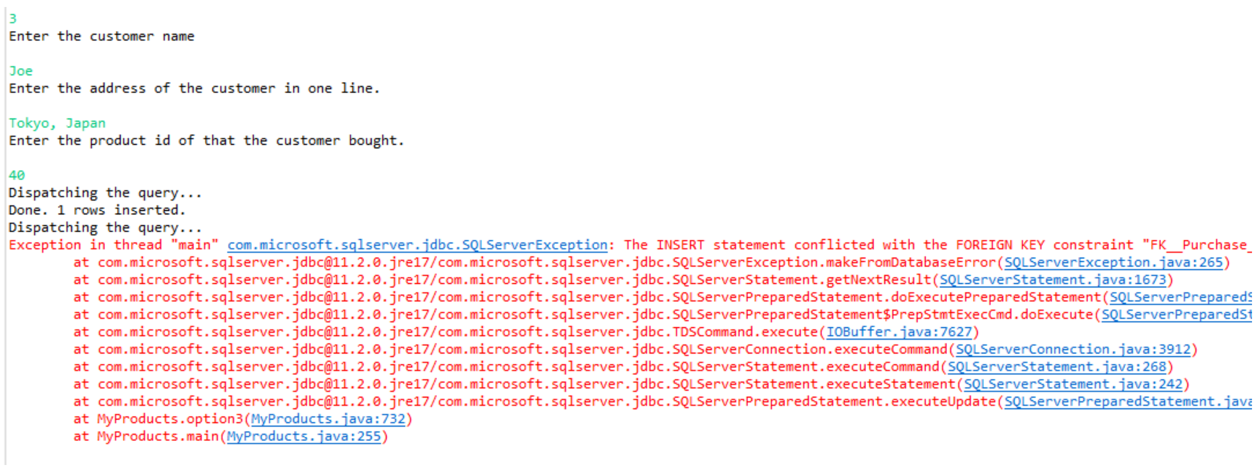
## 6.17. Screenshots showing the testing of three types of errors



```
1
Do you want enter a worker, a quality controller or a technical staff?
Enter w for worker.
Enter q for quality controller.
Enter t for technical staff.

w
Enter the name of the Worker.

Alex
Enter the address of the Worker in one line.

Norman, Oklahoma
Enter the salary of the Worker in dollars.

30000
Enter the maximum number of product a worker produces per day.

5
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__Worker__3722582555E73E92'.
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.j
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServ
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServe
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7627)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:268
```

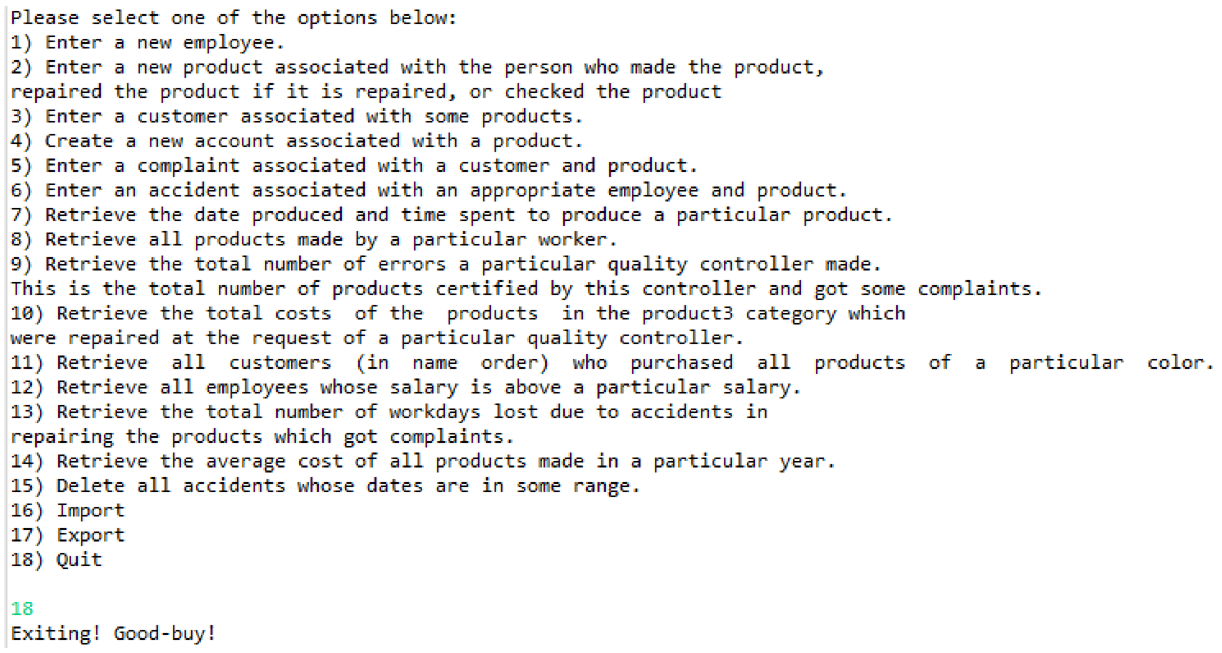


```
5
Enter the complaint id.

56
Enter the complaint date.

2019-03-17
Enter the description of the issue in one line.

The product arrived so late.
What is the treatment?

give discount
Enter the customer's name.

Joe
Enter the product ID.

18
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Make__
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPrepar
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPrepare
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7627)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3912)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:268)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:242)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.j
	at MyProducts.option5(MyProducts.java:845)
	at MyProducts.main(MyProducts.java:261)
```



```
3
Enter the customer name

Joe
Enter the address of the customer in one line.

Tokyo, Japan
Enter the product id of that the customer bought.

40
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Purchase_
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPrepared
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedSt
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7627)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3912)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:268)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:242)
	at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.java
	at MyProducts.option3(MyProducts.java:732)
	at MyProducts.main(MyProducts.java:255)
```

## 6.18. Screenshots showing the testing of the Quit option



```
Please select one of the options below:
1) Enter a new employee.
2) Enter a new product associated with the person who made the product,
repaired the product if it is repaired, or checked the product
3) Enter a customer associated with some products.
4) Create a new account associated with a product.
5) Enter a complaint associated with a customer and product.
6) Enter an accident associated with an appropriate employee and product.
7) Retrieve the date produced and time spent to produce a particular product.
8) Retrieve all products made by a particular worker.
9) Retrieve the total number of errors a particular quality controller made.
This is the total number of products certified by this controller and got some complaints.
10) Retrieve the total costs  of the  products  in the product3 category which
were repaired at the request of a particular quality controller.
11) Retrieve  all  customers  (in  name  order)  who  purchased  all  products  of  a  particular  color.
12) Retrieve all employees whose salary is above a particular salary.
13) Retrieve the total number of workdays lost due to accidents in
repairing the products which got complaints.
14) Retrieve the average cost of all products made in a particular year.
15) Delete all accidents whose dates are in some range.
16) Import
17) Export
18) Quit

18
Exiting! Good-buy!
```

**Task 7**

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

**DataHandler.java**

```
package Individual_Project;

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

public class DataHandler {

      private Connection conn;
    // Azure SQL connection credentials
    private String server = "haid0000-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "haid0000";
    private String password = "Changquan2023";
    static String inputSalary = "0";

    // Resulting connection string
    final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;
trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeou
t=30;",
                    server, database, username, password);
    // Initialize and save the database connection
    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;
        }
        this.conn = DriverManager.getConnection(url);
    }

    public ResultSet getEmployees() throws SQLException {
        getDBConnection();

        final String QUERY12 = "WITH Employee as ( " +
                "SELECT salary,addr, worker_name as employee_name FROM Worker " +
                "UNION " +
                "SELECT salary, addr, qual_cont_name as employee_name FROM
QualityController " +
                "UNION " +
                "SELECT salary, addr, tech_staff_name as employee_name " +
                "FROM TechnicalStaff ) " +
                "SELECT employee_name, addr AS address, salary " +
                "FROM Employee Where salary > ?;";

        final PreparedStatement stmt = conn.prepareStatement(QUERY12);
        stmt.setString(1, inputSalary);
        return stmt.executeQuery();
    }

    public static boolean addSalary(String salary) {
      inputSalary = salary;

      return true;
    }

    public boolean addWorker(String name, String address, String salary, String
maxNumProduct) throws SQLException {
        getDBConnection(); // Prepare the database connection
        // Prepare the SQL statement
        final String QUERY1_WORKER = "INSERT INTO Worker " +
                    "VALUES (?, ?, ?, ?);";
        final PreparedStatement stmt = conn.prepareStatement(QUERY1_WORKER);
        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, name);
        stmt.setString(2, address);
        stmt.setString(3, salary);
        stmt.setString(4, maxNumProduct);

        // Execute the query, if only one record is updated, then we indicate success
by returning true
        return stmt.executeUpdate() == 1;
    }

    public boolean addQualityController(String name, String address, String salary,
String productType) throws SQLException {
        getDBConnection(); // Prepare the database connection
        // Prepare the SQL statement
        final String QUERY1_QUALITY_CONTROLLER = "INSERT INTO QualityController " +
                    "VALUES (?, ?, ?, ?);";
        final PreparedStatement stmt =
conn.prepareStatement(QUERY1_QUALITY_CONTROLLER);
        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, name);
        stmt.setString(2, address);
        stmt.setString(3, salary);
        stmt.setString(4, productType);

        // Execute the query, if only one record is updated, then we indicate success
by returning true
        return stmt.executeUpdate() == 1;
    }

    public boolean addTechnicalStaff(String name, String address, String salary,
String educationRecord, String technicalPosition) throws SQLException {
        getDBConnection(); // Prepare the database connection
        // Prepare the SQL statement
        final String QUERY1_TECHNICAL_STAFF = "INSERT INTO TechnicalStaff " +
                    "VALUES (?, ?, ?, ?, ?);";
        final PreparedStatement stmt = conn.prepareStatement(QUERY1_TECHNICAL_STAFF);
        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, name);
        stmt.setString(2, address);
        stmt.setString(3, salary);
        stmt.setString(4, educationRecord);
        stmt.setString(5, technicalPosition);

        // Execute the query, if only one record is updated, then we indicate success
by returning true
        return stmt.executeUpdate() == 1;
    }

}
```

**add_employee_form.jsp**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Add employee</title>
    </head>
    <body>
        <h2>Use this form for entering a worker.</h2>
        <!--
            Form for collecting user input for the new employee.
            Upon form submission, add_employee.jsp file will be invoked.
        -->

        <form action="add_employee.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Worker</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>Salary</td>
                    <td><div style="text-align: center;">
                    <input type=text name=salary>
                    </div></td>
                </tr>
                <tr>
                    <td>Maximum # of product per day</td>
                    <td><div style="text-align: center;">
                    <input type=text name=maxNumProduct>
                    </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>
        <h2>Use this form for entering a quality controller.</h2>
        <form action="add_employee.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Quality Controller</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>Salary</td>
                    <td><div style="text-align: center;">
                    <input type=text name=salary>
                    </div></td>
                </tr>
                <tr>
                    <td>Product type</td>
                    <td><div style="text-align: center;">
                    <input type=text name=productType>
                    </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>
       <h2>Use this form for entering a technical staff.</h2>

        <form action="add_employee.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Technical Staff</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>Salary</td>
                    <td><div style="text-align: center;">
                    <input type=text name=salary>
                    </div></td>
                </tr>
                <tr>
                    <td>Education Record</td>
                    <td><div style="text-align: center;">
                    <input type=text name=educationRecord>
                    </div></td>
                </tr>
                <tr>
                    <td>Technical Position</td>
                    <td><div style="text-align: center;">
                    <input type=text name=technicalPosition>
                    </div></td>
                </tr>
                <tr>
                    <td><div style="text-align: center;">
```

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

**add_employee_jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="Individual_Project.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 salary = request.getParameter("salary");
    String maxNumProduct = request.getParameter("maxNumProduct");
    String productType = request.getParameter("productType");
    String educationRecord = request.getParameter("educationRecord");
    String technicalPosition = request.getParameter("technicalPosition");


    //If the user is filling out the form for Worker.
    if (maxNumProduct != null) {

        //response.sendRedirect("add_employee_form.jsp");
        // Now perform the query with the data from the form.
        boolean success = handler.addWorker(name, address, salary, 
maxNumProduct);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The New Employee:</h2>

            <ul>
                <li>Name: <%=name%></li>
                <li>Address: <%=address%></li>
                <li>Salary: <%=salary%></li>
                <li>Maximum Number of Products per Day:<%=maxNumProduct%></li>
            </ul>

            <h2>Was successfully inserted.</h2>

            <%
        }
    }

  //If the user is filling out the form for Quality Controller.
    else if (productType != null) {

        // Now perform the query with the data from the form.
        boolean success = handler.addQualityController(name, address, salary, 
productType);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The New Employee:</h2>

            <ul>
                <li>Name: <%=name%></li>
                <li>Address: <%=address%></li>
                <li>Salary: <%=salary%></li>
                <li>Product Type:<%=productType%></li>
            </ul>

            <h2>Was successfully inserted.</h2>
            <%
        }
    }
  //If the user is filling out the form for Technical Staff.
    else if (educationRecord != null) {

        // Now perform the query with the data from the form.
        boolean success = handler.addTechnicalStaff(name, address, salary,
educationRecord, technicalPosition);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The New Employee:</h2>

            <ul>
                <li>Name: <%=name%></li>
                <li>Address: <%=address%></li>
                <li>Salary: <%=salary%></li>
                <li>Education Record:<%=productType%></li>
                <li>Technical Position:<%=productType%></li>
            </ul>

            <h2>Was successfully inserted.</h2>

            <%
        }
    }
    %>
    </body>
</html>
```

**add_salary_form.jsp**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Enter the salary</title>
    </head>
    <body>
        <h2>Retrieve all employees whose salary is above a particular
salary</h2>
        <!--
            Form for collecting the salary of the employees.
            Upon form submission, add_salary.jsp file will be invoked.
        -->
        <form action="add_salary.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Enter the salary:</th>
                </tr>
                <tr>
                    <td>Salary</td>
                    <td><div style="text-align: center;">
                    <input type=text name=salary>
                    </div></td>
                </tr>

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

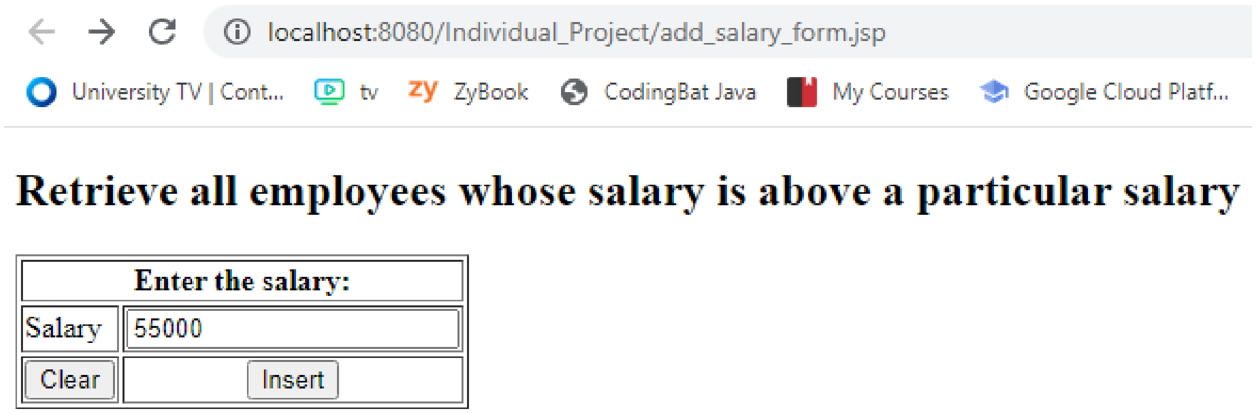
**add_salary.jsp**

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

    <body>
    <%@page import="Individual_Project.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 salary = request.getParameter("salary");

    /*
     * If the user hasn't filled out the salary. This is very simple checking.
     */
    if (salary.equals("")) {
        response.sendRedirect("add_salary_form.jsp");
    } else {

        // Now perform the query with the data from the form.
        boolean success = DataHandler.addSalary(salary);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
           <h2>Request is successfully submitted</h2>
                <li>salary: <%=salary%></li>


            <a href="get_some_employees.jsp">Get the list of employees.</a>
            <%
        }
    }
    %>
    </body>
</html>
```

**get_some_employees.jsp**

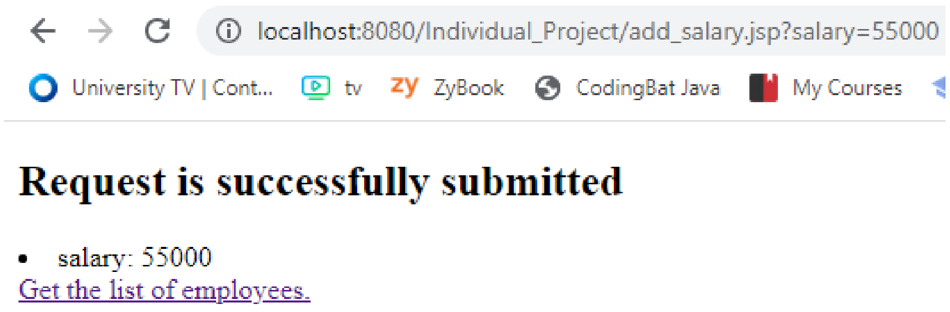
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>MyProducts</title>
    </head>
    <body>
        <%@page import="Individual_Project.DataHandler"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get the employees
from the database
            final DataHandler handler = new DataHandler();
            final ResultSet employees = handler.getEmployees();
        %>
        <!-- The table for displaying all the employees who has more than a
particular salary -->
        <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>Salary</h4>
              </td>
            </tr>
            <%
               while(employees.next()) { // For each employee record
returned...
                   // Extract the attribute values for every row returned
                   final String name = employees.getString("employee_name");
                   final String address = employees.getString("address");
                   final String salary = employees.getString("salary");
                   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\"> " + salary + "</td>");
                   out.println("</tr>");
               }
               %>

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

**Result of query 12, before query 1**



## Retrieve all employees whose salary is above a particular salary

| Enter the salary: | |
|---|---|
| Salary | 55000 |
| Clear | Insert |

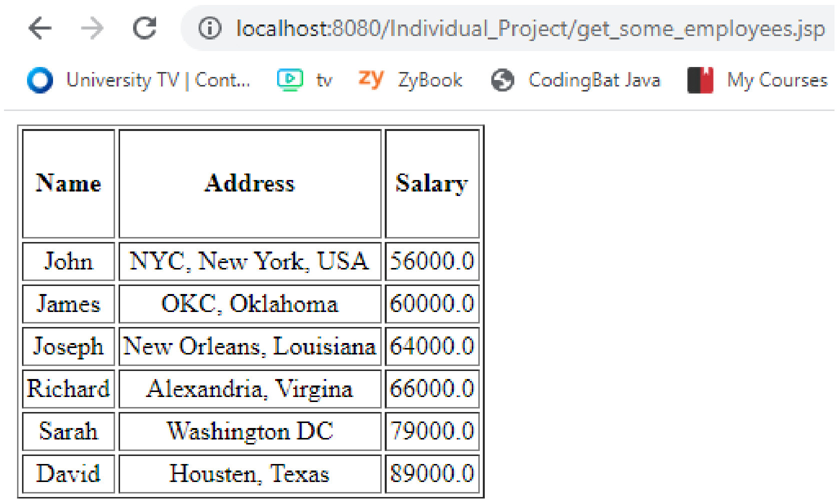


localhost:8080/Individual_Project/add_salary.jsp?salary=55000
University TV | Cont...
tv
ZyBook
CodingBat Java
My Courses

## Request is successfully submitted

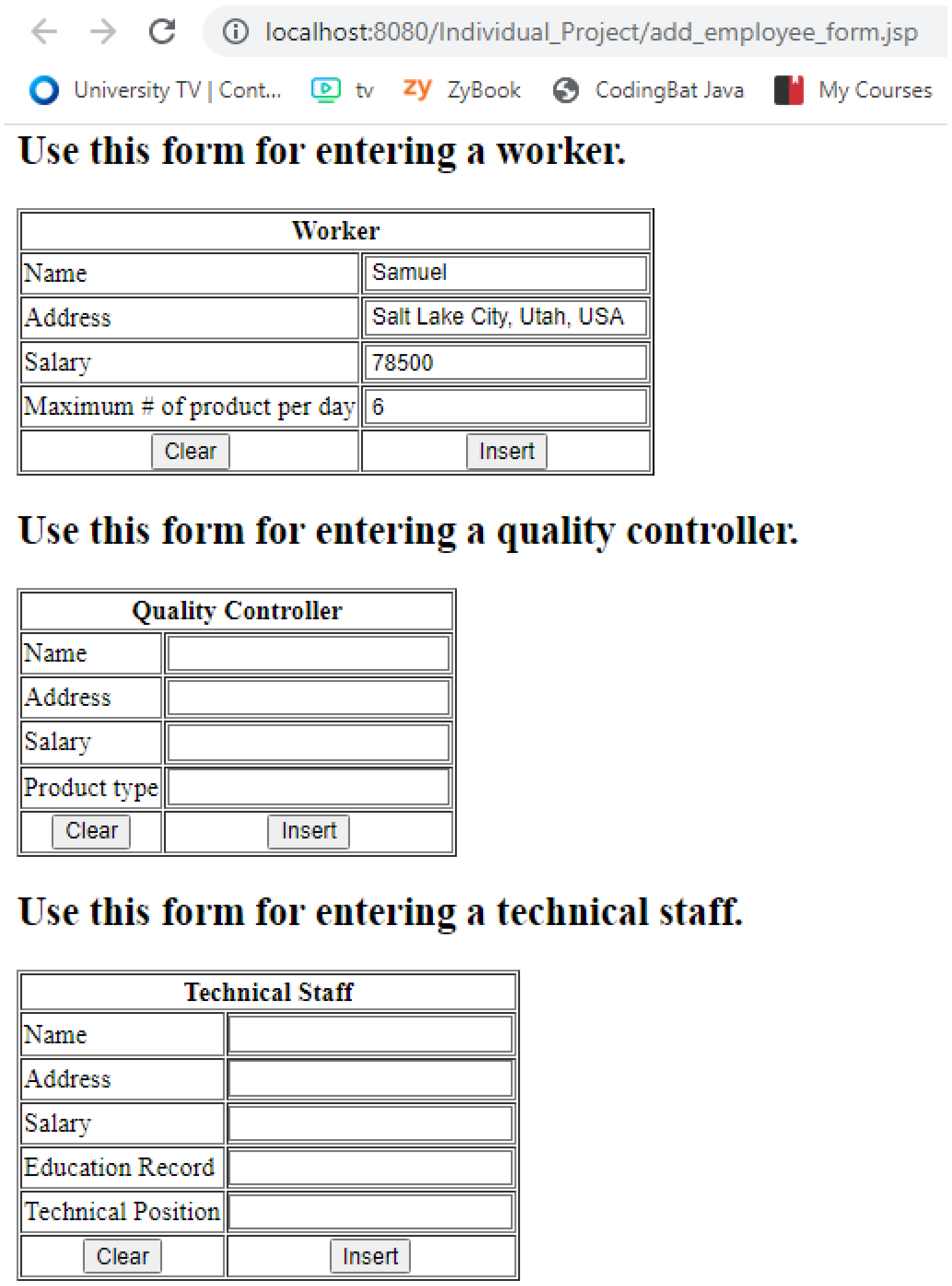
- salary: 55000

Get the list of employees.



| Name | Address | Salary |
|---|---|---|
| John | NYC, New York, USA | 56000.0 |
| James | OKC, Oklahoma | 60000.0 |
| Joseph | New Orleans, Louisiana | 64000.0 |
| Richard | Alexandria, Virgina | 66000.0 |
| Sarah | Washington DC | 79000.0 |
| David | Housten, Texas | 89000.0 |

**Result of Query 1**



Use this form for entering a worker.

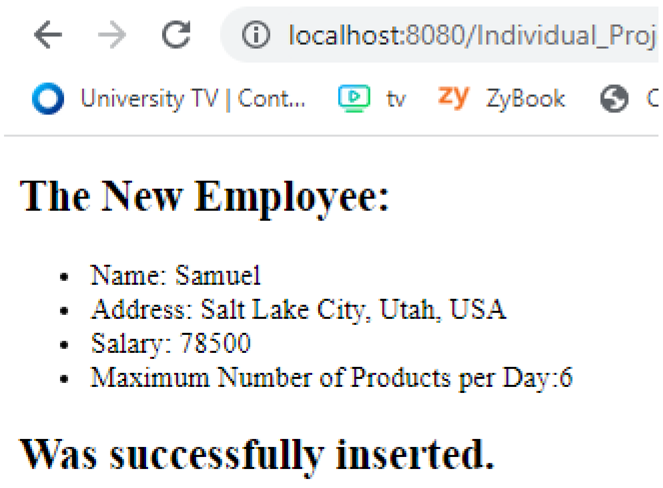
| Worker | |
|---|---|
| Name | Samuel |
| Address | Salt Lake City, Utah, USA |
| Salary | 78500 |
| Maximum # of product per day | 6 |
| Clear | Insert |

Use this form for entering a quality controller.

| Quality Controller | |
|---|---|
| Name | |
| Address | |
| Salary | |
| Product type | |
| Clear | Insert |

Use this form for entering a technical staff.

| Technical Staff | |
|---|---|
| Name | |
| Address | |
| Salary | |
| Education Record | |
| Technical Position | |
| Clear | Insert |



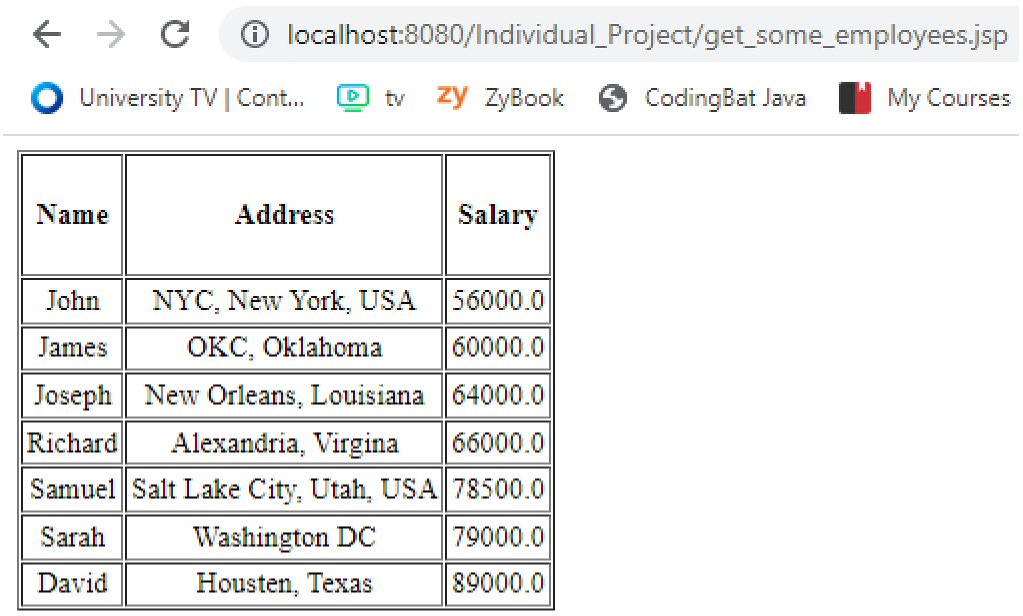
## The New Employee:

- Name: Samuel
- Address: Salt Lake City, Utah, USA
- Salary: 78500
- Maximum Number of Products per Day:6

## Was successfully inserted.

**Result of query 12, after query 1**

We can see that Samuel is added to the table.



localhost:8080/Individual_Project/get_some_employees.jsp

| Name | Address | Salary |
|---|---|---|
| John | NYC, New York, USA | 56000.0 |
| James | OKC, Oklahoma | 60000.0 |
| Joseph | New Orleans, Louisiana | 64000.0 |
| Richard | Alexandria, Virgina | 66000.0 |
| Samuel | Salt Lake City, Utah, USA | 78500.0 |
| Sarah | Washington DC | 79000.0 |
| David | Housten, Texas | 89000.0 |