**Wahidullah Haidari, 113523274,** [**wahid@ou.edu**](mailto:wahid@ou.edu)

**Assignment: Individual Project**

**Semester and Year: Fall 2022**

**Instructor: Dr. Le Gruenwald**

**SCORE:**

**Tasks Performed Page Number**

Task 1. 4-5

1.1. ER Diagram 4-4

1.2. Relational Database Schema 5-5

Task 2. Schema Diagram 6-6

Task 3. 7-11

3.1. Discussion of storage structures for tables 7-9

3.2. Discussion of storage structures for tables (Azure SQL Database) 10-11

Task 4. SQL statements and screenshots showing the creation of 12-18

tables in Azure SQL Database

Task 5. 19-59

5,1 SQL statements (and Transact SQL stored procedures, if any) 19-24

Implementing all queries (1-15 and error checking)

5,2 The Java source program and screenshots showing 25-59

its successful compilation

Task 6. Java program Execution 60-104

6.1. Screenshots showing the testing of query 1 60-65

6.2. Screenshots showing the testing of query 2 66-78

6.3. Screenshots showing the testing of query 3 79-82

6.4. Screenshots showing the testing of query 4 83-87

6.5. Screenshots showing the testing of query 5 88-90

6.6. Screenshots showing the testing of query 6 91-92

6.7. Screenshots showing the testing of query 7 93-93

6.8. Screenshots showing the testing of query 8 94-94

6.9. Screenshots showing the testing of query 9 95-95

6.10. Screenshots showing the testing of query 10 96-96

6.11. Screenshots showing the testing of query 11 97-97

6.12. Screenshots showing the testing of query 12 98-98

6.13. Screenshots showing the testing of query 13 98-98

6.14. Screenshots showing the testing of query 14 98-98

6.15. Screenshots showing the testing of query 15 99-99

6.16. Screenshots showing the testing of the Import and 100-101

Export options

6.17. Screenshots showing the testing of three types of errors 102-103

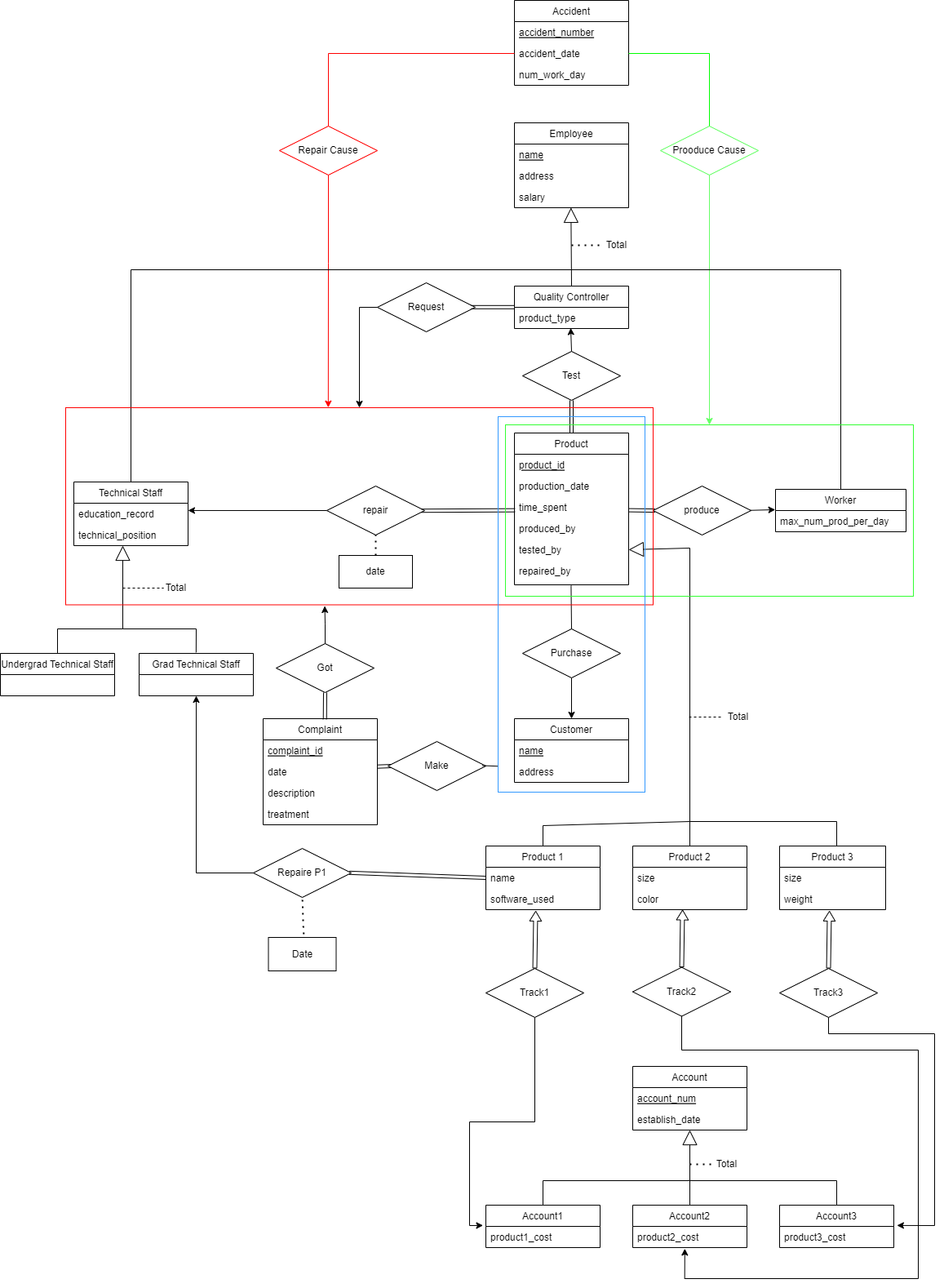
6.18. Screenshots showing the testing of the Quit option 104-104

Task 7. Web database application and its execution 105-118

7.1. Web database application source program and screenshots showing 105-116

Its successful compilation

7.2. Screenshots showing the testing of the Web database application 117-120

**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

**Diagram, schematic

Description automatically generated**

**Task 3:**

**3.1.** Appropriate Storage Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name | Query# and Type | Search Key | Query Frequency | Selected Fie Organization | Justification |
| Worker | 1 (Insertion)  12 (range search) |  | 2/month  1/month | Heap file | We insert more frequently than range search. |
| Quality Controller | 1 (Insertion)  12 (range search) |  | 2/month  1/month | Heap file | We insert more frequently than range search |
| Technical Staff | 1 (Insertion)  12 (range search) |  | 2/month  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)  7 (random search)  8 (random search)  14 (random search) | worker\_name | 400/day  100/day  2000/day  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)  14 (random search) |  | 400/day  5/day | Heap file | Insertion is done more frequently than random search. |
| Product2 | 2 (Insertion)  11 (random search)  14 (random search) |  | 400/day  5/month  5/day | Heap file | Insertion is done more frequently than random search. |
| Product3 | 2 (Insertion)  10 (random search)  14 (random search) |  | 400/day  40/day  5/day | Heap file | Insertion is done more frequently than random search. |
| Customer | 3 (Insertion)  11 (Random Search |  | 50/day  5/Month | Heap file | Insertion is done more frequently than random search. |
| Account1 | 4 (Insertion)  14 (random search) |  | 40/day  5/day | Heap file | Insertion is done more frequently than random search. |
| Account2 | 4 (Insertion)  14 (random search) |  | 40/day  5/day | Heap file | Insertion is done more frequently than random search. |
| Account3 | 4 (Insertion)  10 (random search)  14 (random search) |  | 40/day  40/day  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)  13 (random search)  15 (range search) | accident\_date | 1/week  1/month  1/day | Indexed Sequential File based on accident\_date  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)  13 (random search) |  | 1/week  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  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)  11 (random search) |  | 50/day  5/month | Heap File | Insertion is done more frequently than random search. |
| Make | 5 (Insertion)  9 (random search) | product\_id | 30/day  400/day | Sequential File based on primary key product\_id  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)  8 (random search) | worker\_name | 400/day  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)  9 (random search) | product\_id | 400/day  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)  10 (random search) |  | 400/day  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\_id | 1/month | Sequential File based on primary key product\_id  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

--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

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(

    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

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');

--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

    (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

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;encrypt=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 RepareP1 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** **Graphical user interface, text, application, email

Description automatically generated**

**Text

Description automatically generated**

Text, letter

Description automatically generated

Text

Description automatically generated

Content of the Table Worker after insertions

SELECT \* FROM Worker;

A screenshot of a computer

Description automatically generated with medium confidence

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Content of the Table QualityController after insertions

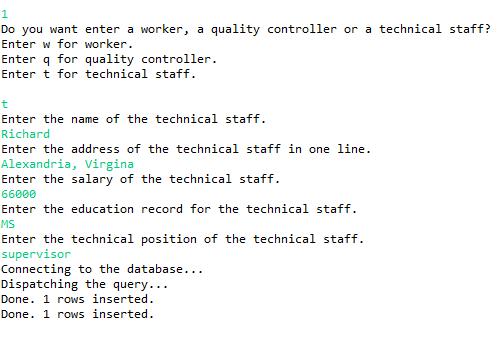
SELECT \* FROM QualityController;

A picture containing graphical user interface

Description automatically generated

Text, letter

Description automatically generated



Text, letter

Description automatically generated

Content of the Table TechnicalStaff after insertions:

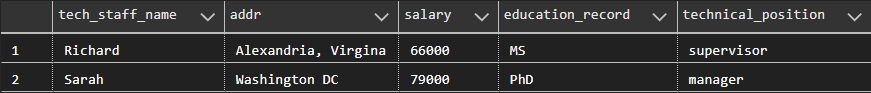
SELECT \* FROM TechnicalStaff;

A screenshot of a computer

Description automatically generated

Content of the Table GradTechnicalStaff after insertions:

SELECT \* FROM GradTechnicalStaff;



Content of the Table UndergradTechnicalStaff after insertions:

SELECT \* FROM UndergradTechnicalStaff;



**6.2. Screenshots showing the testing of query 2** Text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Content of the Table Product1 after insertions:

SELECT \* FROM Product1;

Graphical user interface, application

Description automatically generated

Table

Description automatically generated with medium confidence

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Content of the Table Product2 after insertions:

SELECT \* FROM Product2;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Content of the Table Product3 after insertions:

SELECT \* FROM Product3;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Content of the Table Product after insertions:

SELECT \* FROM Product;

A screenshot of a computer

Description automatically generated

Content of the Table Produce after insertions:

SELECT \* FROM Produce;

A picture containing table

Description automatically generated

Content of the Table Test after insertions:

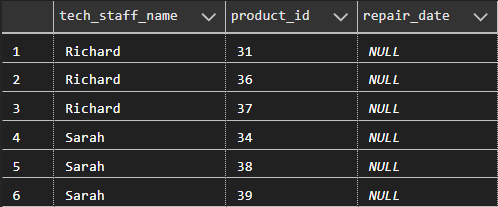
SELECT \* FROM Test;

Table

Description automatically generated with medium confidence

Content of the Table Repair after insertions:

SELECT \* FROM Repair;



Content of the Table RepairP1 after insertions:

SELECT \* FROM RepairP1;

A picture containing table

Description automatically generated

Content of the Table Request after insertions:

SELECT \* FROM Request;

A screenshot of a computer

Description automatically generated with medium confidence

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

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Graphical user interface, text

Description automatically generated

Content of the Table Customer after insertions:

SELECT \* FROM Customer;

A screenshot of a computer

Description automatically generated with medium confidence

Content of the Table Purchase after insertions:

SELECT \* FROM Purchase;

Funnel chart

Description automatically generated with low confidence

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

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Content of the Table Account1 after insertions:

SELECT \* FROM Account1;

Table

Description automatically generated with medium confidence

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Content of the Table Account2 after insertions:

SELECT \* FROM Account2;

Table

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Content of the Table Account3 after insertions:

SELECT \* FROM Account3;

Table

Description automatically generated

**6.5. Screenshots showing the testing of query 5** Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

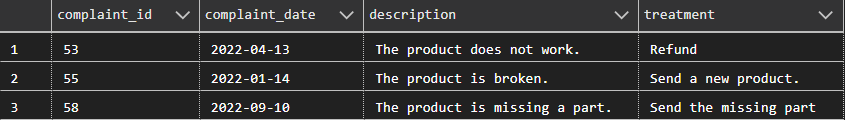
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Content of the Table Complaint after insertions:

SELECT \* FROM Complaint;



Content of the Table Make after insertions:

SELECT \* FROM Make;

Table

Description automatically generated

Content of the Table Cot after insertions:

SELECT \* FROM Got;

Table

Description automatically generated with medium confidence

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

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

SELECT \* FROM Accident;

Table

Description automatically generated

SELECT \* FROM ProduceCause;

A picture containing table

Description automatically generated

SELECT \* FROM RepairCause;

Graphical user interface

Description automatically generated with low confidence

**6.7. Screenshots showing the testing of query 7** Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

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

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

**6.9. Screenshots showing the testing of query 9** Graphical user interface, text

Description automatically generated

Text

Description automatically generated

**Text

Description automatically generated**

**6.10. Screenshots showing the testing of query 10** Text

Description automatically generated

**Chart, text

Description automatically generated with medium confidence**

Text

Description automatically generated with low confidence

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

**Text

Description automatically generated**

Text, letter

Description automatically generated

Text, letter

Description automatically generated

**6.12. Screenshots showing the testing of query 12**  Text, letter

Description automatically generated

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

**Text

Description automatically generated**

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

**Text, letter

Description automatically generated**

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

Text, letter

Description automatically generated

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

Content of the file employees.csv

**Graphical user interface, application

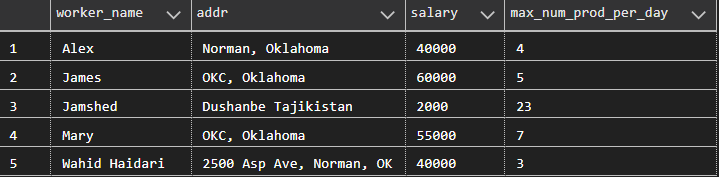
Description automatically generated**

**Graphical user interface, text

Description automatically generated with medium confidence**

Content of the table after import:

SELECT \* FROM Worker



SELECT \* FROM QualityController

A screenshot of a computer

Description automatically generated with medium confidence

SELECT \* FROM TechnicalStaff

A screenshot of a computer

Description automatically generated

**Testing Export**

**Text

Description automatically generated**

Content of the exported file:

Graphical user interface, text, application

Description automatically generated

**6.17. Screenshots showing the testing of three types of errors** Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

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

**Text, letter

Description automatically generated**

**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;loginTimeout=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** Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

**Result of Query 1**

Graphical user interface, application, table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

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

We can see that Samuel is added to the table.

Table

Description automatically generated