

Name: Nasri Binsaleh

ID: 113429299

Email Address: nasri.binsaleh-1@ou.edu

Assignment: Individual Project

Course: CS/DSA 4513, Online Sections 995-999

Semester and Year: Fall 2022

Instructor: Dr. Le Gruenwald

SCORE:

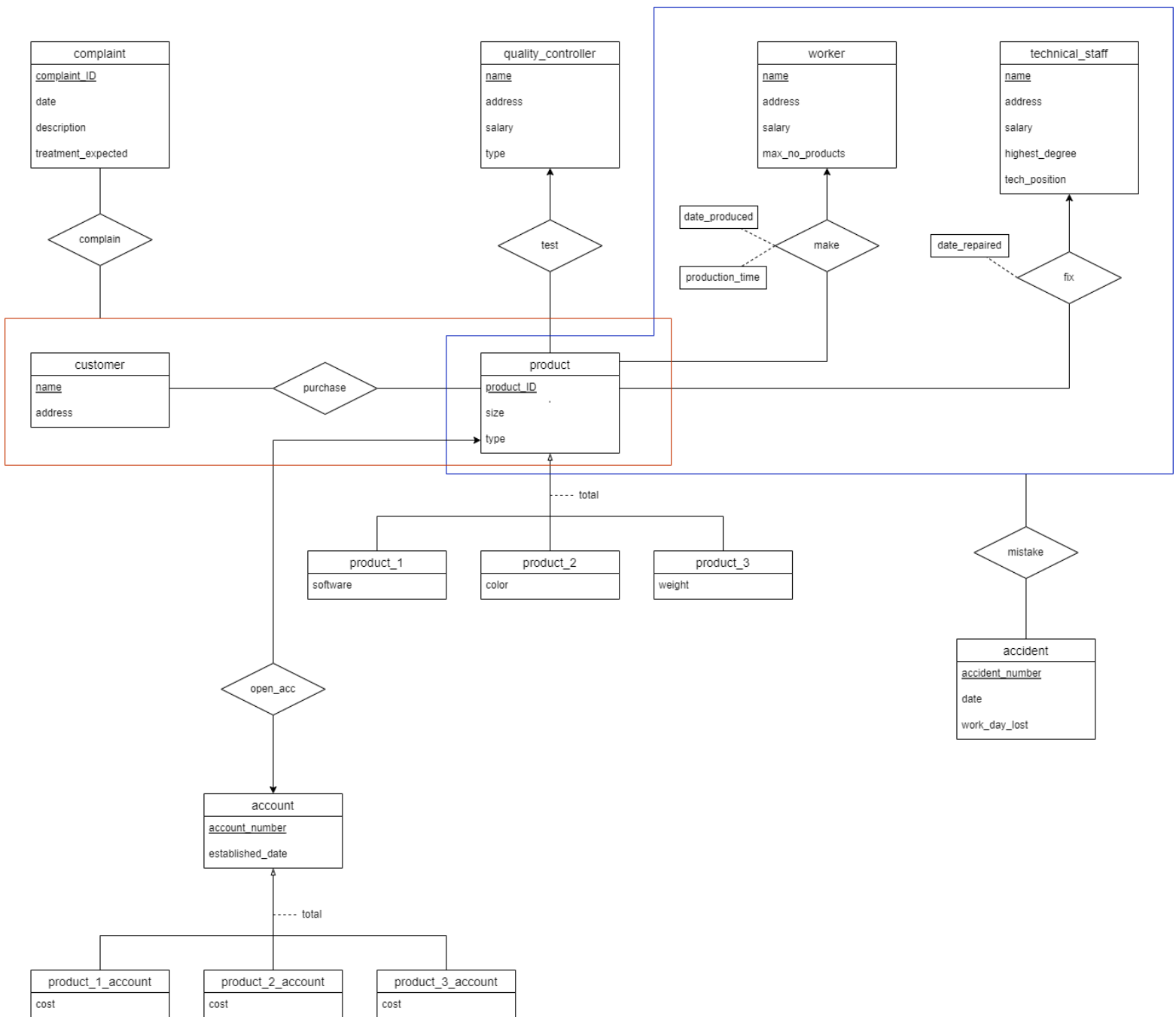
Table of Contents

Tasks Performed	Page Number
Task 1	4-8
1.1 ER Diagram	4
1.2 Relational Database Schema	5-7
Task 2 Schema Diagram	8
Task 3	9-11
3.1 Discussion of storage structures for tables	9-10
3.2 Discussion of storage structures for tables (Azure SQL Database)	10-11
Task 4 SQL statements and screenshots showing the creation of tables in Azure SQL Database	11-14
Task 5.	14-46
5.1 SQL statements (and Transact SQL stored procedures, if any) Implementing all queries (1-15 and error checking)	14-23
5.2 The Java source program and screenshots showing its successful compilation	24-49
Task 6. Java program Execution	49-62
6.1 Screenshots showing the testing of query 1	50-51
6.2 Screenshots showing the testing of query 2	51-53
6.3 Screenshots showing the testing of query 3	54
6.4 Screenshots showing the testing of query 4	55
6.5 Screenshots showing the testing of query 5	55-56
6.6 Screenshots showing the testing of query 6	56-57
6.7 Screenshots showing the testing of query 7	57
6.8 Screenshots showing the testing of query 8	58
6.9 Screenshots showing the testing of query 9	58-59
6.10 Screenshots showing the testing of query 10	59-60
6.11 Screenshots showing the testing of query 11	60-61
6.12 Screenshots showing the testing of query 12	61

6.13 Screenshots showing the testing of query 13	61
6.14 Screenshots showing the testing of query14	62
6.15 Screenshots showing the testing of query 15	62
6.16 Screenshots showing the testing of the Import option	63-64
6.17 Screenshots showing the testing of Export option	64
6.18. Screenshots showing the testing of the Quit option	64
6.19 Screenshots showing the testing of three types of errors	65-66
Task 7. Web database application and its execution	66-77
7.1. Web database application source program and screenshots showing Its successful compilation	66-76
7.2. Screenshots showing the testing of the Web database application	76-77

Task 1

ER Diagram



Relational Database

Table 1: Quality Controller Database Table Design

qname	qaddress	qsalary	qtype

Table 2: Worker Database Table Design

wname	waddress	wsalary	max_no_products

Table 3: Technical Staff Database Table Design

tname	taddress	tsalary	highest_degree	tech_position

Table 4: Product Database Table Design

product_ID	size	type

Table 5: Product 1 Database Table Design

product_ID	size	type	software	account_number	established_date	p1_cost

Table 6: Product 2 Database Table Design

product_ID	size	type	color	account_number	established_date	p2_cost

Table 7: Product 3 Database Table Design

product_ID	size	type	weight	account_number	established_date	p2_cost

Table 8: Accident Database Table Design

accident_number	date	work_day_lost

Table 9: Customer Database Table Design

cname	caddress

Table 10: Complaint Database Table Design

complaint_ID	date	description	treatment_expected

Table 11: Database of 'make' when worker produces a product

wname	product_ID	date_produced	production_time

Table 12: Database of when technical staff 'fix' a product

tname	product_ID	date_repair

Table 13: Database for 'test' Relationship when quality controller tests the product

qname	product_ID

Table 14: Database for 'mistake' Relationship when a worker or a technical staff makes a mistake

accident_number	product_ID	wname	tname

Table 15: Database for 'purchase' Relationship when a customer purchases a product

cname	product_ID

Table 16: Database for 'complain' Relationship for when a customer complains

cname	complaint_ID	product_ID

Relation Schemas

quality_controller(qname, qaddress, qsalary, qtype)

worker(wname, waddress, wsalary, max_no_products)

technical_staff(tname, taddress, tsalary, highest_degree, tech_position)

product(product_ID, size, ptype, account_number, established_date, cost)

product_1(product_ID, size, ptype, software, account_number, established_date, cost)

product_2(product_ID, size, ptype, color, account_number, established_date, cost)

product_3(product_ID, size, ptype, pweight, account_number, established_date, cost)

accident(accident_number, accident_date, work_day_lost)

customer(cname, caddress)

complaint(complaint_ID, complaint_date, complaint_description, treatment_expected)

test(qname, product_ID)

make(wname, product_ID, date_produced, production_time)

fix(tname, product_ID, date_repaired)

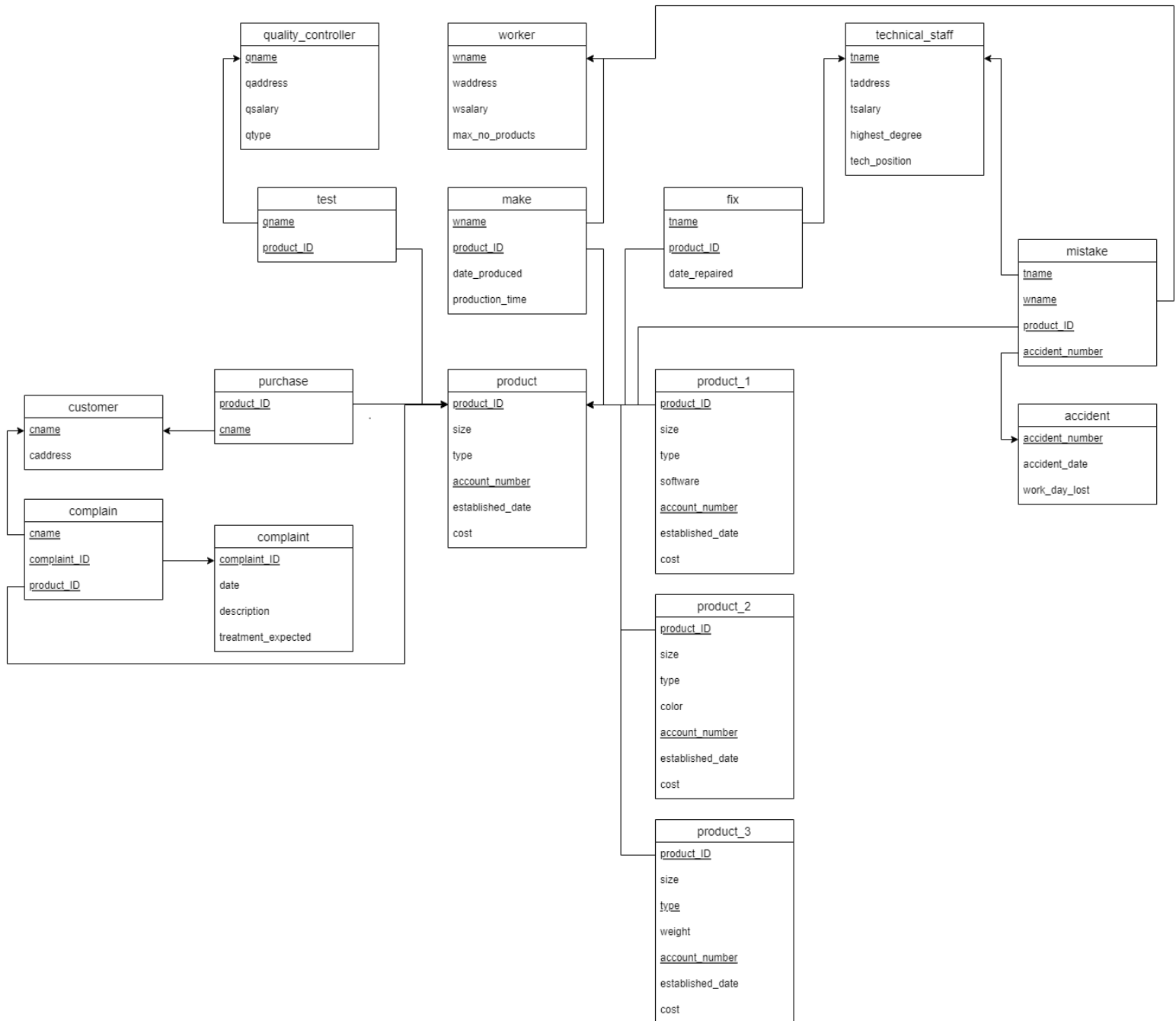
mistake(accident_number, product_ID, wname, tname)

purchase(cname, product_ID)

complain(cname, complaint_ID, product_ID)

Task 2

Schema Diagram



Task 3

3.1)

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justification
quality_controller	(1) insert (12) range search	salary	2/month 1/month	index sequential on search key salary	The most frequent query is range search on salary, and index sequential file is best for range search.
worker	(1) insert (12) range search	salary	2/month 1/month	index sequential on search key salary	The most frequent query is range search on salary, and index sequential file is best for range search.
technical_staff	(1) insert (12) range search	salary	2/month 1/month	index sequential on search key salary	The most frequent query is range search on salary, and index sequential file is best for range search.
product	(2) insert (8) random search (14) random search	product_ID product_ID	400/day 2000/day 5/day	dynamic hashing with hash key product_ID	The most frequent query is random search on product_ID, and hashing is best for random search.
product_1	(2) insert (4) insert (14) random search	product_ID	400/day 40/day 5/day	dynamic hashing with hash key product_ID	Even though insert is the most frequent query, there is still a good amount of random search on product_ID. Thus, dynamic hashing would be the most efficient for both inserting and random searching.
product_2	(2) insert (4) insert (11) random search (14) random search	color product_ID	400/day 40/day 5/month 5/day	dynamic hashing with hash key product_ID	Even though insert is the most frequent query, there is still a good amount of random search on product_ID. Thus, dynamic hashing would be the most efficient for both inserting and random searching.
product_3	(2) insert (4) insert (10) random search (14) random search	product_ID product_ID	400/day 40/day 40/day 5/day	dynamic hashing with hash key product_ID	Even though insert is the most frequent query, there is still a good amount of random search on product_ID. Thus, dynamic hashing would be the most efficient for both inserting and random searching.
accident	(6) insert (13) random search (15) range search + delete	accident_no. accident_date	1/week 1/month 1/day	index sequential on search key accident_date	The most frequent query is range search on accident_date, and index sequential file is best for range search.

customer	(3) insert		50/day	heap	Since the only query is to insert, using heap will be the most efficient as the data can be put just at the end of the heap.
complaint	(5) insert		30/day	heap	Since the only query is to insert, using heap will be the most efficient as the data can be put just at the end of the heap.
make	(2) insert (7) random search (8) random search (14) random search	product_ID name date_produced	400/day 100/day 2000/day 5/day	dynamic hashing with hash key name	The most frequent query is random search on name, and hashing is best for random search.
fix	(2) insert (10) random search (13) random search	product_ID product_ID	400/day 40/day 1/month	dynamic hashing with hash key product_ID	Even though insert is the most frequent query, there is still a good amount of random search on product_ID. Thus, dynamic hashing would be the most efficient for both inserting and random searching.
test	(2) insert (9) random search (10) random search	name product_ID	400/day 400/day 40/day	dynamic hashing with hash key name	The most frequent query is random search, and the most frequent search key is on name, and hashing is best for random search.
mistake	(6) insert (13) random search (15) range search + delete	product_ID accident_no.	1/week 1/month 1/day	index sequential on search key accident_no.	The most frequent query is range search on accident_no., and index sequential file is best for range search.
purchase	(3) insert (11) range search	name	50/day 5/month	index sequential on search key name	Even though insert is the most frequent query, there is still a range search on name. Thus, dynamic hashing would be the most efficient for both inserting and range searching.
complain	(5) insert (9) random search (13) random search	product_ID product_ID	30/day 400/day 1/month	dynamic hashing with hash key product_ID	The most frequent query is random search on product_ID, and hashing is best for random search.

3.2)

Since file organizations above, works great on SQL, there was no modification done on the structures. Most of the tables, when created, already is indexed automatically on their respective

primary keys. The table for employees needs an extra index on salary, thus the index was created on them.

Task 4 SQL Statement to Create Table

```
-- Drop tables to start fresh
DROP TABLE IF EXISTS test;
DROP TABLE IF EXISTS make;
DROP TABLE IF EXISTS fix;
DROP TABLE IF EXISTS mistake;
DROP TABLE IF EXISTS purchase;
DROP TABLE IF EXISTS complain;
DROP TABLE IF EXISTS quality_controller;
DROP TABLE IF EXISTS worker;
DROP TABLE IF EXISTS technical_staff;
DROP TABLE IF EXISTS product_1;
DROP TABLE IF EXISTS product_2;
DROP TABLE IF EXISTS product_3;
DROP TABLE IF EXISTS product;
DROP TABLE IF EXISTS accident;
DROP TABLE IF EXISTS customer;
DROP TABLE IF EXISTS complaint;

-- Create tables

-- Quality Controller
CREATE TABLE quality_controller (
    qname varchar(30) NOT NULL,
    qaddress varchar(50) NOT NULL,
    qsalary REAL NOT NULL,
    qtype INT NOT NULL,
    PRIMARY KEY (qname)
);

-- create an index on quality_controller table with salary being the serch key.
CREATE INDEX qsalary_idx
ON quality_controller(qsalary ASC)

-- Worker
CREATE TABLE worker (
    wname varchar(30) NOT NULL,
    waddress varchar(50) NOT NULL,
    wsalary REAL NOT NULL,
    max_no_products INT NOT NULL,
    PRIMARY KEY (wname)
);

-- create an index on worker table with salary being the serch key.
CREATE INDEX worker
ON worker(wsalary ASC)
```

```

-- Technical Staff
CREATE TABLE technical_staff (
    tname varchar(30) NOT NULL,
    taddress varchar(50) NOT NULL,
    tsalary REAL NOT NULL,
    highest_degree varchar(30) NOT NULL,
    tech_position varchar(30) NOT NULL,
    PRIMARY KEY (tname)
);

-- create an index on worker table with salary being the serch key.
CREATE INDEX technical_staff
ON technical_staff(tsalary ASC)

-- Product
CREATE TABLE product (
    product_ID INT NOT NULL,
    size VARCHAR(15) NOT NULL,
    ptype INT NOT NULL,
    account_number INT,
    established_date DATE,
    cost REAL,
    PRIMARY KEY (product_ID)
);

-- Product 1
CREATE TABLE product_1 (
    product_ID INT NOT NULL,
    software varchar(30) NOT NULL,
    PRIMARY KEY (product_ID),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Product 2
CREATE TABLE product_2 (
    product_ID INT NOT NULL,
    color varchar(30) NOT NULL,
    PRIMARY KEY (product_ID),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Product 3
CREATE TABLE product_3 (
    product_ID INT NOT NULL,
    pweight REAL NOT NULL,
    PRIMARY KEY (product_ID),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Accident
CREATE TABLE accident (
    accident_number INT NOT NULL,
    accident_date DATE NOT NULL,
    work_day_lost INT,
    PRIMARY KEY (accident_number));

```

```

-- Customer
CREATE TABLE customer (
    cname varchar(30) NOT NULL,
    address varchar(50) NOT NULL,
    PRIMARY KEY (cname)
);

-- Complaint
CREATE TABLE complaint (
    complaint_ID INT NOT NULL,
    complaint_date DATE NOT NULL,
    complaint_description varchar(50) NOT NULL,
    treatment_expected varchar(30) NOT NULL,
    PRIMARY KEY (complaint_ID)
);

-- Test -- Inspection from quality controller
CREATE TABLE test (
    qname varchar(30) NOT NULL,
    product_ID INT NOT NULL,
    PRIMARY KEY (qname, product_ID),
    FOREIGN KEY (qname) REFERENCES quality_controller(qname),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Make -- Production from worker
CREATE TABLE make (
    wname varchar(30) NOT NULL,
    product_ID INT NOT NULL,
    date_produced DATE NOT NULL,
    production_time REAL NOT NULL,
    PRIMARY KEY (wname, product_ID),
    FOREIGN KEY (wname) REFERENCES worker(wname),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Fix -- Repair from technical staff
CREATE TABLE fix (
    tname varchar(30) NOT NULL,
    product_ID INT NOT NULL,
    date_repaired DATE NOT NULL,
    PRIMARY KEY (tname, product_ID),
    FOREIGN KEY (tname) REFERENCES technical_staff(tname),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Mistake -- when a worker or technical staff makes an accident
CREATE TABLE mistake (
    accident_number INT NOT NULL,
    product_ID INT NOT NULL,
    wname varchar(30),
    tname varchar(30),
    PRIMARY KEY (accident_number, product_ID),
    FOREIGN KEY (accident_number) REFERENCES accident(accident_number),
    FOREIGN KEY (wname) REFERENCES worker(wname),

```

```

        FOREIGN KEY (tname) REFERENCES technical_staff(tname),
        FOREIGN KEY (product_ID) REFERENCES product(product_ID)
    );

-- Purchase -- when a customer purchases a product
CREATE TABLE purchase (
    cname varchar(30),
    product_ID INT NOT NULL,
    PRIMARY KEY (cname, product_ID),
    FOREIGN KEY (cname) REFERENCES customer(cname),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID)
);

-- Complain -- when a customer complains about a product
CREATE TABLE complain (
    cname varchar(30),
    product_ID INT NOT NULL,
    complaint_ID INT NOT NULL,
    PRIMARY KEY (cname, product_ID, complaint_ID),
    FOREIGN KEY (cname) REFERENCES customer(cname),
    FOREIGN KEY (product_ID) REFERENCES product(product_ID),
    FOREIGN KEY (complaint_ID) REFERENCES complaint(complaint_ID)
);

```

Task 5

5.1 Write SQL statements for all queries (1-18)

```

-- Queries/Procedures

-- 1. Enter a new employee

-- 1.1 Enter a quality controller
DROP PROCEDURE IF EXISTS quality_controller_insert
GO
CREATE PROCEDURE quality_controller_insert
(@qname VARCHAR(50),
@qaddress VARCHAR(70),
@qsalary INT,
@qtype INT)
AS
BEGIN
    INSERT INTO quality_controller (qname, qaddress, qsalary, qtype) VALUES (@qname,
@qaddress, @qsalary, @qtype);
END
GO

-- 1.2 Enter a worker
DROP PROCEDURE IF EXISTS worker_insert
GO
CREATE PROCEDURE worker_insert
(@wname VARCHAR(50),
@waddress VARCHAR(70),
@wsalary INT,

```

```

@max_no_products INT)
AS
BEGIN
    INSERT INTO worker (wname, waddress, wsalary, max_no_products) VALUES (@wname,
@waddress, @wsalary, @max_no_products);
END
GO

-- 1.3 Enter a technical staff
DROP PROCEDURE IF EXISTS technical_staff_insert
GO
CREATE PROCEDURE technical_staff_insert
(@tname VARCHAR(50),
@taddress VARCHAR(70),
@tsalary INT,
@highest_degree VARCHAR(15),
@tech_position VARCHAR(50))
AS
BEGIN
    INSERT INTO technical_staff (tname, taddress, tsalary, highest_degree,
tech_position)
VALUES (@tname, @taddress, @tsalary, @highest_degree, @tech_position);
END
GO

-- 2. Enter a new product
DROP PROCEDURE IF EXISTS product_insert
GO
CREATE PROCEDURE product_insert
(@product_ID INT,
@size VARCHAR(15),
@ptype INT
--@account_number INT,
--@established_date DATE,
--@cost REAL
)
AS
BEGIN
    INSERT INTO product (product_ID, size, ptype)
VALUES (@product_ID, @size, @ptype);
END
GO

-- for Product 1
DROP PROCEDURE IF EXISTS product1_insert
GO
CREATE PROCEDURE product1_insert
(@product_ID INT,
@software VARCHAR(50))
AS
BEGIN
    INSERT INTO product_1 (product_ID, software)
VALUES (@product_ID, @software);
END
GO

```

```

-- for Product 2
DROP PROCEDURE IF EXISTS product2_insert
GO
CREATE PROCEDURE product2_insert
(@product_ID INT,
@color VARCHAR(20))
AS
BEGIN
    INSERT INTO product_2 (product_ID, color)
    VALUES (@product_ID, @color);
END
GO

-- for Product 3
DROP PROCEDURE IF EXISTS product3_insert
GO
CREATE PROCEDURE product3_insert
(@product_ID INT,
@pweight REAL)
AS
BEGIN
    INSERT INTO product_3 (product_ID, pweight)
    VALUES (@product_ID, @pweight);
END
GO

-- Make insert -- insert the worker-production information
DROP PROCEDURE IF EXISTS make_insert
GO
CREATE PROCEDURE make_insert
(@wname VARCHAR(50),
@product_ID INT,
@date_produced DATE,
@production_time REAL)
AS
BEGIN
    INSERT INTO make (wname, product_ID, date_produced, production_time)
    VALUES (@wname, @product_ID, @date_produced, @production_time);
END
GO

-- Fix insert -- insert the technical-staff-repair information
DROP PROCEDURE IF EXISTS fix_insert
GO
CREATE PROCEDURE fix_insert
(@tname VARCHAR(50),
@product_ID INT,
@date_repaired DATE)
AS
BEGIN
    INSERT INTO fix (tname, product_ID, date_repaired)
    VALUES (@tname, @product_ID, @date_repaired);
END
GO

```



```

-- Test insert -- insert the quality-controller-testing information
DROP PROCEDURE IF EXISTS test_insert
GO
CREATE PROCEDURE test_insert
(@qname VARCHAR(50),
@product_ID INT)
AS
BEGIN
    INSERT INTO test (qname, product_ID)
    VALUES (@qname, @product_ID);
END
GO

-- 3. Enter a customer and product purchased
-- Customer insert
DROP PROCEDURE IF EXISTS customer_insert
GO
CREATE PROCEDURE customer_insert
(@cname VARCHAR(50),
@address VARCHAR(50))
AS
BEGIN
    INSERT INTO customer (cname, address)
    VALUES (@cname, @address);
END
GO

-- Purchase insert -- product that customer purchased
DROP PROCEDURE IF EXISTS purchase_insert
GO
CREATE PROCEDURE purchase_insert
(@cname VARCHAR(50),
@product_ID INT)
AS
BEGIN
    INSERT INTO purchase (cname, product_ID)
    VALUES (@cname, @product_ID);
END
GO

-- 4. Create a new account associated with a product
DROP PROCEDURE IF EXISTS account_insert
GO
CREATE PROCEDURE account_insert
(@acc_product_ID INT,
@account_number INT,
@established_date DATE,
@cost REAL)
AS
BEGIN
    UPDATE product
    SET account_number = @account_number, established_date = @established_date, cost
    = @cost
    WHERE product_ID = @acc_product_ID;

```

```

END
GO

-- 5. Enter a complaint associated with a customer and product
-- Insert in complaint
DROP PROCEDURE IF EXISTS complaint_insert
GO
CREATE PROCEDURE complaint_insert
(@complaint_ID INT,
@complaint_date DATE,
@complaint_description VARCHAR(100),
@treatment_expected VARCHAR(100))
AS
BEGIN
    INSERT INTO complaint (complaint_ID, complaint_date, complaint_description,
treatment_expected)
    VALUES (@complaint_ID, @complaint_date, @complaint_description,
@treatment_expected);
END
GO

-- Insert in complain relationship
DROP PROCEDURE IF EXISTS complain_insert
GO
CREATE PROCEDURE complain_insert
(@complain_cname VARCHAR(50),
@complaint_product_ID INT,
@complaint_ID INT)
AS
BEGIN
    INSERT INTO complain (cname, product_ID ,complaint_ID)
    VALUES (@complain_cname, @complaint_product_ID, @complaint_ID);
END
GO

-- 6. Enter an accident associated with an appropriate employee and product
-- Insert in accident
DROP PROCEDURE IF EXISTS accident_insert
GO
CREATE PROCEDURE accident_insert
(@accident_number INT,
@accident_date DATE,
@work_day_lost INT)
AS
BEGIN
    INSERT INTO accident (accident_number, accident_date, work_day_lost)
    VALUES (@accident_number, @accident_date, @work_day_lost);
END
GO

-- Insert in mistake relationship with worker
DROP PROCEDURE IF EXISTS mistake_worker_insert
GO
CREATE PROCEDURE mistake_worker_insert
(@accident_number INT,

```

```

@product_ID INT,
@name VARCHAR(50))
AS
BEGIN
    INSERT INTO mistake (accident_number, product_ID, wname)
    VALUES (@accident_number, @product_ID, @name);
END
GO

-- Insert in mistake relationship with technical staff
DROP PROCEDURE IF EXISTS mistake_technical_insert
GO
CREATE PROCEDURE mistake_technical_insert
(@accident_number INT,
@product_ID INT,
@name VARCHAR(50))
AS
BEGIN
    INSERT INTO mistake (accident_number, product_ID, tname)
    VALUES (@accident_number, @product_ID, @name);
END
GO

-- 7. Retrieve the date produced and time spent to produce a particular product
DROP PROCEDURE IF EXISTS case7
GO
CREATE PROCEDURE case7
(@product_ID INT)
AS
BEGIN
    SELECT date_produced, production_time
    FROM make
    WHERE product_ID = @product_ID
END
GO

-- 8. Retrieve all products made by a particular worker
DROP PROCEDURE IF EXISTS case8
GO
CREATE PROCEDURE case8
(@wname VARCHAR(50))
AS
BEGIN
    SELECT product_ID
    FROM make
    WHERE wname = @wname
END
GO

-- 9. Retrieve the total number of errors a particular quality controller made.
-- This is the total number of products certified by this controller and got some
complaints
DROP PROCEDURE IF EXISTS case9
GO
CREATE PROCEDURE case9

```

```

(@qname VARCHAR(50))
AS
BEGIN
    SELECT COUNT(t.product_ID)
    FROM test t, complain c
    WHERE t.qname = @qname
        AND t.product_ID IN (SELECT c.product_ID)
END
GO

-- 10. Retrieve the total costs of the products in the product3 category which were
repaired at the
-- request of a particular quality controller
DROP PROCEDURE IF EXISTS case10
GO
CREATE PROCEDURE case10
(@qname VARCHAR(50))
AS
BEGIN
    SELECT SUM(p.cost) as total_cost
    FROM test t, fix f, product_3 p3, product p
    WHERE p.product_ID = p3.product_ID
        AND f.product_ID = p3.product_ID
        AND t.product_ID = f.product_ID
        AND t.qname = @qname
END
GO

-- 11. Retrieve all customers (in name order) who purchased all products of a
particular color

DROP PROCEDURE IF EXISTS case11
GO
CREATE PROCEDURE case11
(@color VARCHAR(30))
AS
BEGIN
    SELECT p.cname
    FROM purchase p, product_2 p2
    WHERE p.product_ID = p2.product_ID
        AND p2.color = @color
END
GO

-- 12. Retrieve all employees whose salary is above a particular salary
DROP PROCEDURE IF EXISTS case12
GO
CREATE PROCEDURE case12
(@salary REAL)
AS
BEGIN
    SELECT t.tname
    FROM technical_staff t
    WHERE t.tsalary > @salary
    UNION

```

```

        SELECT q.qname
        FROM quality_controller q
        WHERE q.qsalary > @salary
        UNION
        SELECT w.wname
        FROM worker w
        WHERE w.wsalary > @salary
    END
    GO

-- 13. Retrieve the total number of work days lost due to accidents in repairing the
products which got complaints
DROP PROCEDURE IF EXISTS case13
GO
CREATE PROCEDURE case13
AS
BEGIN
    SELECT SUM(a.work_day_lost) as days_lost
    FROM accident a, mistake m, complain c, fix f
    WHERE a.accident_number = m.accident_number
        AND m.product_ID IN (SELECT c.product_ID)
        AND m.product_ID IN (SELECT f.product_ID)
END
GO

-- 14. Retrieve the average cost of all products made in a particular year
DROP PROCEDURE IF EXISTS case14
GO
CREATE PROCEDURE case14
(@year INT)
AS
BEGIN
    SELECT AVG(p.cost) as avg_cost
    FROM product p, make m
    WHERE p.product_ID = m.product_ID
        AND YEAR(m.date_produced) = @year
END
GO

-- 15. Delete all accidents whose dates are in some range
DROP PROCEDURE IF EXISTS case15
GO
CREATE PROCEDURE case15
(@date1 VARCHAR(10),
@date2 VARCHAR(10))
AS
BEGIN
    DELETE FROM mistake
    WHERE mistake.accident_number IN (SELECT accident.accident_number
        FROM accident
        WHERE [accident_date]
        BETWEEN @date1 AND @date2
    )

```

```

DELETE FROM accident
WHERE [accident_date]
BETWEEN @date1 AND @date2
END
GO

-- 16. Import: enter new employees from a data file until the file is empty (the user
must be asked to enter the input file name);
-- file location -- --
https://storageforproject.blob.core.windows.net/newcontainer/technical_staff.csv

CREATE EXTERNAL DATA SOURCE MyAzureBlobStorage
WITH ( TYPE = BLOB_STORAGE,
        LOCATION = 'https://storageforproject.blob.core.windows.net/newcontainer'
);

DROP PROCEDURE IF EXISTS case16_technical_staff
GO
CREATE PROCEDURE case16_technical_staff
(@file VARCHAR(100))
AS
BEGIN
    -- using dynamic SQL
    DECLARE @CSVfile varchar(100);
    SET @CSVfile = @file
    declare @q nvarchar(MAX);
    set @q=
        'BULK INSERT technical_staff
        FROM '+char(39)+@CSVfile+char(39)+'
        WITH
        (
        DATA_SOURCE = ''MyAzureBlobStorage'',
        FIELDTERMINATOR = ',',',
        ROWTERMINATOR = ''\n'',
        FIRSTROW = 1
        )'
    exec(@q)
END
GO

DROP PROCEDURE IF EXISTS case16_worker
GO
CREATE PROCEDURE case16_worker
(@file VARCHAR(100))
AS
BEGIN
    -- using dynamic SQL
    DECLARE @CSVfile varchar(100);
    SET @CSVfile = @file
    declare @q nvarchar(MAX);
    set @q=
        'BULK INSERT worker
        FROM '+char(39)+@CSVfile+char(39)+'
        WITH
        (

```

```

        DATA_SOURCE = 'MyAzureBlobStorage',
        FIELDTERMINATOR = ',',
        ROWTERMINATOR = '\n',
        FIRSTROW = 1
    )'
exec(@q)
END
GO

```

```

DROP PROCEDURE IF EXISTS case16_quality_controller
GO
CREATE PROCEDURE case16_quality_controller
(@file VARCHAR(100))
AS
BEGIN
    -- using dynamic SQL
    DECLARE @CSVfile varchar(100);
    SET @CSVfile = @file
    declare @q nvarchar(MAX);
    set @q=
        'BULK INSERT quality_controller
        FROM '+char(39)+@CSVfile+char(39)+'
        WITH
        (
            DATA_SOURCE = 'MyAzureBlobStorage',
            FIELDTERMINATOR = ',',
            ROWTERMINATOR = '\n',
            FIRSTROW = 1
        )'
    exec(@q)
END
GO

```

```

-- 17. Export: Retrieve all customers (in name order) who purchased all products of a
particular color and output them to a data file instead of screen
--      (the user must be asked to enter the output file name);
DROP PROCEDURE IF EXISTS case17
GO
CREATE PROCEDURE case17
(@color VARCHAR(30),
@filename VARCHAR(100))
AS
BEGIN
    SELECT p.cname
    FROM purchase p, product_2 p2
    WHERE p.product_ID = p2.product_ID
        AND p2.color = @color
    OUTPUT TO @filename;
END
GO

```

5.2 Java application program that uses JDBC and Azure SQL Database

```
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;

public class Binsaleh_Nasri_IP {

    // Database credentials
    final static String HOSTNAME = "bins0000-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "bins0000";
    final static String PASSWORD = "databaseFall22!";

    // 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;loginTimeou
t=30;",
                HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Query templates
    final static String QUERY_TEMPLATE_1_1 = "EXEC quality_controller_insert
@qname = ?, @qaddress = ?, @qsalary = ?, @qtype = ?;";

    final static String QUERY_TEMPLATE_1_2 = "EXEC worker_insert @wname = ?,
@waddress = ?, @wsalary = ?, @max_no_products = ?;";

    final static String QUERY_TEMPLATE_1_3 = "EXEC technical_staff_insert @tname =
?, @taddress = ?, @tsalary = ?, @highest_degree = ?, @tech_position = ?;";

    final static String QUERY_TEMPLATE_2 = "EXEC product_insert @product_ID = ?,
@size = ?, @ptype = ?;";

    final static String QUERY_TEMPLATE_2_1 = "EXEC product1_insert @product_ID =
?, @software = ?;";

    final static String QUERY_TEMPLATE_2_2 = "EXEC product2_insert @product_ID =
?, @color = ?;";

    final static String QUERY_TEMPLATE_2_3 = "EXEC product3_insert @product_ID =
?, @pweight = ?;";

    final static String QUERY_TEMPLATE_2_make = "EXEC make_insert @wname = ?,
@product_ID = ?, @date_produced = ?, @production_time = ?;";

    final static String QUERY_TEMPLATE_2_fix = "EXEC fix_insert @tname = ?,
@product_ID = ?, @date_repaired = ?;";

    final static String QUERY_TEMPLATE_2_test = "EXEC test_insert @qname = ?,
@product_ID = ?;";
```



```

    final static String QUERY_TEMPLATE_3_customer = "EXEC customer_insert @cname =
?, @caddress = ?;";

    final static String QUERY_TEMPLATE_3_purchase = "EXEC purchase_insert @cname =
?, @product_ID = ?;";

    final static String QUERY_TEMPLATE_4 = "EXEC account_insert @acc_product_ID =
?, @account_number = ?, @established_date = ?, @cost = ?;";

    final static String QUERY_TEMPLATE_5_complaint = "EXEC complaint_insert
@complaint_ID = ?, @complaint_date = ?, @complaint_description = ?,
@treatment_expected = ?;";

    final static String QUERY_TEMPLATE_5_complain = "EXEC complain_insert
@complain_cname = ?, @complaint_product_ID = ?, @complaint_ID = ?;";

    final static String QUERY_TEMPLATE_6_accident = "EXEC accident_insert
@accident_number = ?, @accident_date = ?, @work_day_lost = ?;";

    final static String QUERY_TEMPLATE_6_mistake_worker = "EXEC
mistake_worker_insert @accident_number = ?, @product_ID = ?, @name = ?;";

    final static String QUERY_TEMPLATE_6_mistake_technical = "EXEC
mistake_technical_insert @accident_number = ?, @product_ID = ?, @name = ?;";

    final static String QUERY_TEMPLATE_7 = "EXEC case7 @product_ID = ?;";

    final static String QUERY_TEMPLATE_8 = "EXEC case8 @wname = ?;";

    final static String QUERY_TEMPLATE_9 = "EXEC case9 @qname = ?;";

    final static String QUERY_TEMPLATE_10 = "EXEC case10 @qname = ?;";

    final static String QUERY_TEMPLATE_11 = "EXEC case11 @color = ?;";

    final static String QUERY_TEMPLATE_12 = "EXEC case12 @salary = ?;";

    final static String QUERY_TEMPLATE_13 = "EXEC case13;";

    final static String QUERY_TEMPLATE_14 = "EXEC case14 @year = ?;";

    final static String QUERY_TEMPLATE_15 = "EXEC case15 @date1 = ?, @date2 = ?;";

    final static String QUERY_TEMPLATE_16_quality_controller = "EXEC
case16_quality_controller @file = ?;";

    final static String QUERY_TEMPLATE_16_worker = "EXEC case16_worker @file =
?;";

    final static String QUERY_TEMPLATE_16_technical_staff = "EXEC
case16_technical_staff @file = ?;";

    final static String QUERY_TEMPLATE_17 = "EXEC case17 @color = ?;";

```

```

// User input prompt//
final static String PROMPT =
    "\n\nPlease select one of the options below: \n" +
    "1) Enter a new employee; \n" +
    "2) Enter a new product; \n" +
    "3) Enter a customer; \n" +
    "4) Create a new account; \n" +
    "5) Enter a complaint; \n" +
    "6) Enter an accident; \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" +
    "10) Retrieve the total costs of the products in the product3 category
which were repaired at the request of a particular quality controller; \n" +
    "11) Retrieve all customers 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 work days lost due to accidents in
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: enter new employees from a data file until the file is
empty; \n" +
    "17) Export: Retrieve all customers (in name order) who purchased all
products of a particular color and output them to a data file instead of screen; \n"
+
    "18) Exit! \n \nYour Option? ";

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

    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")) { // Ask user for options until option 4 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 option
                // see what type of employee
                System.out.println("Please choose the type of employee
(enter either 1, 2, or 3): \n1) Quality Controller \n2) Worker \n3) Technical
Staff");

```

```

of employee type        final int etype = sc.nextInt(); // Read in the user input

switch (etype) {
case 1:
    // Collect quality controller data from the user
    System.out.println("Please enter employee's name:");
    // Preceding nextInt, nextFloat, etc. do not consume
    // the user input.
    // We call nextLine to consume that newline
    // nextLine doesn't return anything.
    sc.nextLine();
    final String qname = sc.nextLine(); // Read in the
    user input of q-controller name

    System.out.println("Please enter employee's
address:");
    final String qaddress = sc.nextLine(); // Read in
    user input of employees's address (white-spaces allowed).

    System.out.println("Please enter employee's
salary:");
    final float qsalary = sc.nextFloat(); // Read in
    user input of salary

    System.out.println("Please enter quality
controller's type (which type of product is the controller allowed to inspect):");
    final int qtype = sc.nextInt(); // Read in user
    input of product type associated with the quality controller

    // INSERT QUALITY CONTROLLER
    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(QUERY_TEMPLATE_1_1)) {
            // Populate the query template with the
            data collected from the user

            statement.setString(1, qname);
            statement.setString(2, qaddress);
            statement.setFloat(3, qsalary);
            statement.setInt(4, qtype);

            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 2:
        // Collect worker data from the user
        System.out.println("Please enter employee's name:");
        sc.nextLine();
        final String wname = sc.nextLine(); // Read in the
user input of name

        System.out.println("Please enter employee's
address:");
        final String waddress = sc.nextLine(); // Read in
user input of employees's address (white-spaces allowed).

        System.out.println("Please enter employee's
salary:");
        final float wsalary = sc.nextFloat(); // Read in
user input of salary

        System.out.println("Please enter maximum number of
product a worker and produce per day:");
        final int max_no_products = sc.nextInt(); // Read in
user input of maximum number of product a worker and produce per day

        // INSERT WORKER
        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(QUERY_TEMPLATE_1_2)) {
                // Populate the query template with the
data collected from the user

                statement.setString(1, wname);
                statement.setString(2, waddress);
                statement.setFloat(3, wsalary);
                statement.setInt(4, max_no_products);

                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 3:
        // Collect technical staff data from the user
        System.out.println("Please enter employee's name:");
        sc.nextLine();

```

```

        final String tname = sc.nextLine(); // Read in the
user input of name

        System.out.println("Please enter employee's
address:");

        final String taddress = sc.nextLine(); // Read in
user input of employees's address (white-spaces allowed).

        System.out.println("Please enter employee's
salary:");

        final float tsalary = sc.nextFloat(); // Read in
user input of salary

        System.out.println("Please enter employee's highest
degree (BS, MS, PhD):");
        sc.nextLine();
        final String highest_degree = sc.nextLine(); // Read
in user input of employee's highest degree

        System.out.println("Please enter employee's
technical position:");

        final String tech_position = sc.nextLine(); // Read
in user input of employee's technical position

        // INSERT TECHNICAL_STAFF
        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(QUERY_TEMPLATE_1_3)) {
                // Populate the query template with the
data collected from the user

                statement.setString(1, tname);
                statement.setString(2, taddress);
                statement.setFloat(3, tsalary);
                statement.setString(4, highest_degree);
                statement.setString(5, tech_position);

                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;

        default: // Unrecognized option, re-prompt the user for
the correct one
            System.out.println(String.format(
                "Unrecognized option: %s\n" +

```

```

        "Please try again!",
        etype));
    break;
}

// break out of option 1
break;

case "2": // Insert a product
    System.out.println("Please enter integer product ID:");
    final int product_ID = sc.nextInt(); // Read in the user
input of product ID

    System.out.println("Please enter product size (small,
medium, or large):");
    sc.nextLine();
    final String size = sc.nextLine(); // Read in user input
of product size.

    System.out.println("Please enter product type (1, 2, or
3):");
    final int ptype = sc.nextInt(); // Read in user input of
product type

    System.out.println("Please enter worker's name that
produced the product:");
    sc.nextLine();
    final String wname = sc.nextLine(); // Read in user input
of associated worker.

    System.out.println("Please enter product's production date
(YYYY-MM-DD):");
    final String date_produced = sc.nextLine(); // Read in
user input of production date

    System.out.println("Please enter time spent to produce this
product (in hour):");
    final float production_time = sc.nextFloat(); // Read in
user input of time spent to produce this product

    // INSERT PRODUCT
    //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(QUERY_TEMPLATE_2)) {
            // Populate the query template with the data
            collected from the user

            statement.setInt(1, product_ID);
            statement.setString(2, size);
            statement.setInt(3, ptype);

```

```

        //statement.setInt(4, account_number);
        //statement.setDate(5, established_date);
        //statement.setFloat(6, cost);

        //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 MAKE - production data
try (final Connection connection =
DriverManager.getConnection(URL)) {
    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_2_make)) {
        // Populate the query template with the data
        collected from the user

        statement.setString(1, wname);
        statement.setInt(2, product_ID);
        statement.setDate(3,
java.sql.Date.valueOf(date_produced));
        statement.setFloat(4, production_time);

        //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));
    }
}

// Also collect the information if the product is repaired
or inspected

// collect technical staff
System.out.println("Please enter technical staff's name
that repaired the product if applicable (if the product was not repaired then just
press Enter):");
sc.nextLine();
final String tname = sc.nextLine(); // Read in user input
of associated worker.

// if user input technical staff, then do the followings
if (tname != null && !tname.trim().isEmpty()) {
    System.out.println("Please enter product's repair
date (YYYY-MM-DD):");
    final String date_repaired = sc.nextLine(); // Read
in user input of repair date

```

```

        // INSERT FIX - repair data
        // Get a database connection and prepare a query
statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_2_fix)) {
                // Populate the query template with the
data collected from the user
                statement.setString(1, tname);
                statement.setInt(2, product_ID);
                statement.setDate(3,
java.sql.Date.valueOf(date_repaired));

                // 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));
            }
        }

        // collect quality controller
        System.out.println("Please enter quality controller's name
that inspected the product if applicable (if the product was not inspected then just
press Enter):");

        final String qname = sc.nextLine(); // Read in user input
of associated worker.

        // if user input quality controller, then do the
followings
        if (qname != null && !qname.trim().isEmpty()) {
            // INSERT TEST - inspection data
            // Get a database connection and prepare a query
statement
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_2_test)) {
                    // Populate the query template with the
data collected from the user
                    statement.setString(1, qname);
                    statement.setInt(2, product_ID);

                    // 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));
                }
            }
        }

```



```

    }
}

switch (ptype) {
case 1:
    System.out.println("Please enter the main software
used for the product:");
    final String software = sc.nextLine(); // Read in
the user input of software used for the product

    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(QUERY_TEMPLATE_2_1)) {
            // Populate the query template with the
data collected from the user

            statement.setInt(1, product_ID);
            statement.setString(2, software);

            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 2:
    System.out.println("Please enter product's color:");
    final String color = sc.nextLine(); // Read in the
user input of product's color

    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(QUERY_TEMPLATE_2_2)) {
            // Populate the query template with the
data collected from the user

            statement.setInt(1, product_ID);
            statement.setString(2, color);

            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 3:
    System.out.println("Please enter product's weight
(in pound):");
    final Float pweight = sc.nextFloat(); // Read in the
user input of product's weight

    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(QUERY_TEMPLATE_2_3)) {
            // Populate the query template with the
data collected from the user

            statement.setInt(1, product_ID);
            statement.setFloat(2, pweight);

            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;

break;

case "3": // Insert a Customer
    // Collect customer data from the user
    System.out.println("Please enter customer's name:");
    sc.nextLine();
    final String cname = sc.nextLine(); // Read in the user
input of name

    // ask if the customer is an existing customer (if yes, we
don't need to insert in customer table, only in purchase table)
    System.out.println("Is this an existing customer?
(yes/no):");

    final String cexist = sc.nextLine();

    if (ceexist.equals("no")) {
        System.out.println("Please enter customer's
address:");
    }
}
}
}

```

```

        final String caddress = sc.nextLine(); // Read in
user input of customer's address

        // INSERT CUSTOMER
        System.out.println("Inserting customer 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(QUERY_TEMPLATE_3_customer)) {
                // Populate the query template with the
data collected from the user

                statement.setString(1, cname);
                statement.setString(2, caddress);

                // Actually execute the populated query
                final int rows_inserted =

statement.executeUpdate();
                System.out.println(String.format("Done.
%d rows inserted.", rows_inserted));
            }
        }

        // then add purchase data associated with a product
        System.out.println("\nPlease enter product ID of the
purchased product:");

        final int purchase_product_ID = sc.nextInt(); // Read in
user input of product_ID

        // INSERT PURCHASE DATA
        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(QUERY_TEMPLATE_3_purchase)) {
                // Populate the query template with the data
collected from the user

                statement.setString(1, cname);
                statement.setInt(2, purchase_product_ID);

                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 "4": // Create a new account associated with a product

            System.out.println("Please enter integer product ID:");
            final int acc_product_ID = sc.nextInt(); // Read in the
user input of product ID

            System.out.println("Please enter product account
number:");
            final int account_number = sc.nextInt(); // Read in user
input of product account number

            long millis=System.currentTimeMillis();
            java.sql.Date established_date = new java.sql.Date(millis); //
get the established time (input date basically)

            System.out.println("Please enter product cost (just enter number
without currency):");
            final float cost = sc.nextFloat(); // Read in user input
of product cost

            // INSERT ACCOUNT DATA
            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(QUERY_TEMPLATE_4)) {
                    // Populate the query template with the data
collected from the user

                    statement.setInt(1, acc_product_ID);
                    statement.setInt(2, account_number);
                    statement.setDate(3, established_date);
                    statement.setFloat(4, cost);

                    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 "5": // Enter a complaint associated with a customer and product

            // get complaint information
            System.out.println("Please enter integer complaint ID:");
            final int complaint_ID = sc.nextInt(); // Read in the user
input of complaint ID

```

```

        System.out.println("Please enter customer's name:");
        sc.nextLine();
        final String complain_cname = sc.nextLine(); // Read in
the user input of customer that complains

        System.out.println("Please enter integer product ID in the
complaint:");
        final int complaint_product_ID = sc.nextInt(); // Read in
the user input of product ID

        System.out.println("Please enter complaint date (YYYY-MM-
DD):");
        sc.nextLine();
        final String complaint_date = sc.nextLine(); // Read in
user input of complaint date

        System.out.println("Please enter complaint description:");
        final String complaint_description = sc.nextLine(); //
Read in user input of complaint description

        System.out.println("Please enter treatment expected for this
complaint:");
        final String treatment_expected = sc.nextLine(); // Read
in user input of treatment expected

        // INSERT complaint DATA
        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(QUERY_TEMPLATE_5_complaint)) {
                // Populate the query template with the data
                collected from the user
                statement.setInt(1, complaint_ID);
                statement.setDate(2,
java.sql.Date.valueOf(complaint_date));
                statement.setString(3,
complaint_description);
                statement.setString(4, treatment_expected);

                // 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 COMPLAIN
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {

```

```

        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_5_complain)) {
        // Populate the query template with the data
collected from the user

        statement.setString(1, complain_cname);
        statement.setInt(2, complaint_product_ID);
        statement.setInt(3, complaint_ID);

        // 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 "6": // Enter an accident associated with an appropriate employee
and product

        // get accident information
        System.out.println("Please enter integer accident number:");
        sc.nextLine();
        final int accident_number = sc.nextInt(); // Read in the
user input of accident number

        System.out.println("Please enter accident date (YYYY-MM-
DD):");
        sc.nextLine();
        final String accident_date = sc.nextLine(); // Read in
user input of accident date

        System.out.println("Please enter number of work-day
lost:");
        final int work_day_lost = sc.nextInt(); // Read in the
user input of work-day lost

        System.out.println("Which product is it? Please enter
integer product ID:");
        sc.nextLine();
        final int accident_product_ID = sc.nextInt(); // Read in
the user input of product ID

        System.out.println("Was the accident caused by a worker or
a technical staff? Enter 1 for worker or 2 for technical staff:");
        final int atype = sc.nextInt(); // Read in user input of
accident type

        // we have two cases, one for when worker is the caused,
and the other for technical staff
        switch(atype) {
        case 1:

```

```

// get worker's name
System.out.println("Please enter worker's name:");
sc.nextLine();
final String accident_wname = sc.nextLine();

// INSERT ACCIDENT
// Get a database connection and prepare a query
statement
try (final Connection connection =
DriverManager.getConnection(URL)) {
    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_6_accident)) {
        // Populate the query template with the
data collected from the user
        statement.setInt(1, accident_number);
        statement.setDate(2,
java.sql.Date.valueOf(accident_date));
        statement.setInt(3, work_day_lost);

        // 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 MISTAKE WITH WORKER
// Get a database connection and prepare a query
statement
try (final Connection connection =
DriverManager.getConnection(URL)) {
    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_6_mistake_worker)) {
        // Populate the query template with the
data collected from the user
        statement.setInt(1, accident_number);
        statement.setInt(2,
accident_product_ID);
        statement.setString(3, accident_wname);

        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 2:

```

```

        // get technical staff's name
        System.out.println("Please enter technical staff's
name:");

        sc.nextLine();
        final String accident_tname = sc.nextLine();

        // INSERT ACCIDENT
        // Get a database connection and prepare a query
statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_6_accident)) {
                // Populate the query template with the
data collected from the user

                statement.setInt(1, accident_number);
                statement.setDate(2,
java.sql.Date.valueOf(accident_date));

                statement.setInt(3, work_day_lost);

                // 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 MISTAKE WITH TECHNICAL STAFF
        // Get a database connection and prepare a query
statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_6_mistake_technical)) {
                // Populate the query template with the
data collected from the user

                statement.setInt(1, accident_number);
                statement.setInt(2,
accident_product_ID);

                statement.setString(3, accident_tname);

                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;

```



```

        default: // Unrecognized option, re-prompt the user for
the correct one
        System.out.println(String.format(
            "Unrecognized option: %s\n" +
            "Please try again!",
            atype));
        break;
    }

    break;

    case "7": // Retrieve the date produced and time spent to produce a
particular product
        // ask the user for the product ID
        System.out.println("Please enter integer product ID:");
        final int c7_product_ID = sc.nextInt(); // Read in the
user input of product ID

        // Run the SQL Procedure
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_7)) {
                // Populate the query template with the data
collected from the user

                statement.setInt(1, c7_product_ID);

                // call the stored procedure
                ResultSet resultSet =

statement.executeQuery();

                System.out.println(String.format("Result for
product %d:", c7_product_ID));
                System.out.println("Date Produced |
Production Time ");

                while (resultSet.next()) {
                    System.out.println(String.format("%s
| %s Hours", resultSet.getString(1),
resultSet.getString(2)));
                }
            }
        }

        break;

    case "8": // Retrieve all products made by a particular worker

        // ask the user for worker's name
        System.out.println("Please enter worker's name:");
        sc.nextLine();
        final String c8_wname = sc.nextLine();

```

```

        // Run the SQL Procedure
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_8)) {
                // Populate the query template with the data
                collected from the user

                statement.setString(1, c8_wname);

                // call the stored procedure
                ResultSet resultSet =

statement.executeQuery();

                System.out.println(String.format("Product IDs
made by %s:", c8_wname));

                while (resultSet.next()) {
                    System.out.println(String.format("%s",
resultSet.getString(1)));
                }
            }
        }

        break;

    case "9": // Retrieve the total number of errors a particular quality
controller made.
        // ask the user for quality controller's name
        System.out.println("Please enter quality controller's name:");
        sc.nextLine();
        final String c9_qname = sc.nextLine();

        // Run the SQL Procedure
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_9)) {
                // Populate the query template with the data
                collected from the user

                statement.setString(1, c9_qname);

                // call the stored procedure
                ResultSet resultSet =

statement.executeQuery();

                System.out.println(String.format("Total
number of errors made by %s: ", c9_qname));

                while (resultSet.next()) {
                    System.out.print(String.format("%s",
resultSet.getString(1)));

```

```

        }
    }
}

    break;

    case "10": // Retrieve the total costs of the products in the
product3 category which were repaired at the
                // request of a particular quality controller
                // ask the user for quality controller's name
                System.out.println("Please enter quality controller's name:");
                sc.nextLine();
                final String c10_qname = sc.nextLine();

                // Run the SQL Procedure
                // Get a database connection and prepare a query statement
                try (final Connection connection =
DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_10)) {
                        // Populate the query template with the data
                        collected from the user

                        statement.setString(1, c10_qname);

                        // call the stored procedure
                        ResultSet resultSet =

statement.executeQuery();

                        System.out.println(String.format("Total cost
of the repaired products inspected by %s is:", c10_qname));

                        while (resultSet.next()) {
                            System.out.print(String.format("%s",
resultSet.getString(1)));
                        }
                    }
                }

                break;

    case "11": // Retrieve all customers (in name order) who purchased all
products of a particular color

                // ask the user for a color
                System.out.println("Please enter product color:");
                sc.nextLine();
                final String c11_color = sc.nextLine();

                // Run the SQL Procedure
                // Get a database connection and prepare a query statement
                try (final Connection connection =
DriverManager.getConnection(URL)) {

```

```

        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_11)) {
            // Populate the query template with the data
            collected from the user
            statement.setString(1, c11_color);

            // call the stored procedure
            ResultSet resultSet =

statement.executeQuery();

            System.out.println(String.format("All
customers who purchased all products of a %s color:", c11_color));

            while (resultSet.next()) {
                System.out.println(String.format("%s",
resultSet.getString(1)));
            }
        }

        break;

    case "12": // Retrieve all employees whose salary is above a particular
salary

        // ask the user for a salary
        System.out.println("Please enter a salary of your choice (just number
without $ sign):");

        final float c12_salary = sc.nextFloat();

        // Run the SQL Procedure
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_12)) {
                // Populate the query template with the data
                collected from the user
                statement.setFloat(1, c12_salary);

                // call the stored procedure
                ResultSet resultSet =

statement.executeQuery();

                System.out.println(String.format("All
employees whose salary is above $%f:", c12_salary));

                while (resultSet.next()) {
                    System.out.println(String.format("%s",
resultSet.getString(1)));
                }
            }
        }
    }
}

```

```

        break;

        case "13": // Retrieve the total number of work days lost due to
accidents in repairing the products which got complaints

            System.out.println("Connecting to the database...");
            // Get the database connection, create statement and
execute it right away, as no user input need be collected
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                System.out.println("Dispatching the query...");
                try (final Statement statement =
connection.createStatement());
                    final ResultSet resultSet =
statement.executeQuery(QUERY_TEMPLATE_13)) {

                        System.out.println("total number of work days
lost due to accidents in repairing the products which got complaints:");

                        // Unpack the tuples returned by the database
and print them out to the user
                        while (resultSet.next()) {
                            System.out.println(String.format("%s
days", resultSet.getString(1)));
                        }
                    }
                }
            }

        break;

        case "14": // Retrieve the average cost of all products made in a
particular year

            // ask the user for a year
            System.out.println("Please enter a year of your choice:");
            final int c14_year = sc.nextInt();

            // Run the SQL Procedure
            // Get a database connection and prepare a query statement
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_14)) {
                    // Populate the query template with the data
collected from the user

                    statement.setInt(1, c14_year);

                    // call the stored procedure
                    ResultSet resultSet =
statement.executeQuery();

```

```

        System.out.println(String.format("The average
cost of all products made in a year %d:", c14_year));

        while (resultSet.next()) {
            System.out.println(String.format("%s",
resultSet.getString(1)));
        }
    }

    break;

case "15": // Delete all accidents whose dates are in some range
    // ask the user for date range
    System.out.println("Please enter lower boundary date (YYYY-MM-DD):");
    sc.nextLine();
    final String c15_date1 = sc.nextLine();

    System.out.println("Please enter upper boundary date
(YYYY-MM-DD):");
    final String c15_date2 = sc.nextLine();

    // Run the SQL Procedure
    // Get a database connection and prepare a query statement
    try (final Connection connection =
DriverManager.getConnection(URL)) {
        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_15)) {
            // Populate the query template with the data
            collected from the user

            statement.setString(1, c15_date1);
            statement.setString(2, c15_date2);

            System.out.println(String.format("Deleting
Accidents from %s to %s:", c15_date1, c15_date2));
            // Actually execute the populated query
            final int rows_deleted =
statement.executeUpdate();

            System.out.println(String.format("Done. %d
rows deleted.", rows_deleted));
        }
    }

    break;

case "16": // Import: enter new employees from a data file until the file
is empty (the user must be asked to enter the input file name);

    // see what type of employee

```

```

        System.out.println("Please choose the type of employee
(enter either 1, 2, or 3): \n1) Quality Controller \n2) Worker \n3) Technical
Staff");
        final int c16type = sc.nextInt(); // Read in the user
input of employee type

        // ask the user for file name
        System.out.println("Please enter file name to import:");
        sc.nextLine();
        final String c16_file_name = sc.nextLine();

        switch (c16type) {
        case 1:
            // Run the SQL Procedure
            // Get a database connection and prepare a query
statement
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_16_quality_controller)) {
                    // Populate the query template with the
data collected from the user
                    statement.setString(1, c16_file_name);

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

        case 2:
            // Run the SQL Procedure
            // Get a database connection and prepare a query
statement
            try (final Connection connection =
DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_16_worker)) {
                    // Populate the query template with the
data collected from the user
                    statement.setString(1, c16_file_name);

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

```

```

        break;

    case 3:
        // Run the SQL Procedure
        // Get a database connection and prepare a query
statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {
            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_16_technical_staff)) {
                // Populate the query template with the
data collected from the user
                statement.setString(1, c16_file_name);

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

    default: // Unrecognized option, re-prompt the user for
the correct one
        System.out.println(String.format(
            "Unrecognized option: %s\n" +
            "Please try again!",
            c16type));
        break;
    }

    break;

    case "17": // Export: Retrieve all customers (in name order) who
purchased all products of a particular color and output them to a data file instead
of screen
        //(the user must be asked to enter the output file name);

        // ask the user for a color
        System.out.println("Please enter product color:");
        sc.nextLine();
        final String c17_color = sc.nextLine();

        // ask the user for file name
        System.out.println("Please enter file name to export:");
        final String c17_file_name = sc.nextLine();
        /*
        // Run the SQL Procedure
        // Get a database connection and prepare a query statement
        try (final Connection connection =
DriverManager.getConnection(URL)) {

```



```

        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_17)) {
            // Populate the query template with the data
collected from the user

            statement.setString(1, c17_color);
            statement.setString(1, c17_file_name);

            // call the stored procedure
            //final int rows_inserted =

statement.executeUpdate();

            System.out.println(String.format("Exported
all customers who purchased all products of a %s color to %s:", c17_color,
c17_file_name));
        }
    }
    */
    System.out.println(String.format("Exported all customers
who purchased all products of a %s color to %s", c17_color, c17_file_name));
    break;

    case "18": // Exit option -- Do nothing, the while loop will terminate
upon the next iteration
        System.out.println("Exiting... See you next time!"); //
HAHA good-BUY XD
        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
}
}

```

Task 6 Test the program in Task 5.

To populate the database, perform 10 queries for each type (1-4) and 3 queries for each type (5-6) and show the contents of the affected tables after the 10 queries of each type (1-4) are completed and after the 3 queries for each type (5-6) are completed.

```

WELCOME TO THE DATABASE SYSTEM OF MyProducts, Inc.

Please select one of the options below:
1) Enter a new employee;
2) Enter a new product;
3) Enter a customer;
4) Create a new account;
5) Enter a complaint;
6) Enter an accident;
7) Retrieve the date produced and time spent to produce a particular product;
8) Retrieve all products made by a particular worker;
9) Retrieve the total number of errors a particular quality controller made;
10) Retrieve the total costs of the products in the product3 category which were repaired at the request of a particular quality controller;
11) Retrieve all customers who purchased all products of a particular color;
12) Retrieve all employees whose salary is above a particular salary;
13) Retrieve the total number of work days lost due to accidents in repairing the products which got complaints;
14) Retrieve the average cost of all products made in a particular year;
15) Delete all accidents whose dates are in some range;
16) Import: enter new employees from a data file until the file is empty;
17) Export: Retrieve all customers (in name order) who purchased all products of a particular color and output them to a data file instead of screen;
18) Exit!

```

Query 1

Enter new employee example:

```

Your Option?
1
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
1
Please enter employee's name:
Art
Please enter employee's address:
1 Lane
Please enter employee's salary:
89000
Please enter quality controller's type (which type of product is the controller allowed to inspect):
1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.

```

After 10 queries of Query 1: Inserting employee

Quality Controller:

	qname	qaddress	qsalary	qtype
1	Art	1 Lane	89000	1
2	Gun	55 St	89000	2

Worker:

	wname	waddress	wsalary	max_no_products
1	Douglas	43 Dr	78000	6
2	Four	34 Rd	90000	10
3	Nobita	76 St	88000	9

Technical Staff:

	tname ▼	taddress ▼	tsalary ▼	highest_de... ▼	tech_posit... ▼
1	Game	22 Ave	79000	BS	Manager
2	James	490 Norman Road	95200	MS	Director

Query 2

Enter new product examples:

```
Your Option?
2
Please enter integer product ID:
1
Please enter product size (small, medium, or large):
small
Please enter product type (1, 2, or 3):
1
Please enter worker's name that produced the product:
Four
Please enter product's production date (YYYY-MM-DD):
2022-11-20
Please enter time spent to produce this product (in hour):
89
Please enter technical staff's name that repaired the product if applicable (if the product was not repaired then just press Enter):
Please enter quality controller's name that inspected the product if applicable (if the product was not inspected then just press Enter):
Please enter the main software used for the product:
GM
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
Your Option?
2
Please enter integer product ID:
3
Please enter product size (small, medium, or large):
small
Please enter product type (1, 2, or 3):
2
Please enter worker's name that produced the product:
Nobita
Please enter product's production date (YYYY-MM-DD):
2022-09-12
Please enter time spent to produce this product (in hour):
75
Please enter technical staff's name that repaired the product if applicable (if the product was not repaired then just press Enter):
James
Please enter product's repair date (YYYY-MM-DD):
2022-10-10
Please enter quality controller's name that inspected the product if applicable (if the product was not inspected then just press Enter):
Gun
Please enter product's color:
red
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Tables after inserting products:

product:

	prod... ▼	size ▼	ptype ▼	account_nu... ▼	establishe... ▼	cost ▼
1	1	small	1	NULL	NULL	NULL
2	2	medium	2	NULL	NULL	NULL
3	3	small	2	NULL	NULL	NULL
4	4	large	3	NULL	NULL	NULL
5	5	medium	2	NULL	NULL	NULL
6	6	small	1	NULL	NULL	NULL
7	7	medium	3	NULL	NULL	NULL
8	8	small	3	NULL	NULL	NULL
9	9	medium	2	NULL	NULL	NULL
1...	10	small	1	NULL	NULL	NULL

product_1:

	product_ID ▼	software ▼
1	1	GM
2	6	SA
3	10	LS

product_2:

	product_ID ▼	color ▼
1	2	green
2	3	red
3	5	blue
4	9	red

product_3:

	product_ID ▼	pweight ▼
1	4	56
2	7	21
3	8	9

make:

	wname ▼	product_ID ▼	date_produced ▼	production_time ▼
1	Douglas	4	2022-10-10	58
2	Douglas	8	2012-10-11	13
3	Four	1	2022-11-20	89
4	Four	2	2022-10-05	97
5	Four	5	2022-08-24	75
6	Four	6	2012-12-12	10
7	Four	9	2012-12-25	13
8	Nobita	3	2022-09-12	75
9	Nobita	7	2012-11-12	12
1...	Nobita	10	2015-05-11	12

fix:

	tname ▼	product_ID ▼	date_repaired ▼
1	Game	2	2022-10-15
2	Game	8	2012-11-11
3	Game	9	2012-12-29
4	James	3	2022-10-10
5	James	5	2022-11-12
6	James	10	2016-02-14

test:

	qname ▼	product_ID ▼
1	Art	2
2	Art	10
3	Gun	3
4	Gun	9

Query 3

Example of inserting customer:

```
Your Option?
3
Please enter customer's name:
One
Is this an existing customer? (yes/no):
no
Please enter customer's address:
33 Free Way
Inserting customer to the database...
Done. 1 rows inserted.

Please enter product ID of the purchased product:
1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Tables after inserting 10 customers:

customer:

	cname ▼	caddress ▼
1	Eight	56 Dr
2	Five	22 Rd
3	Four	99 Ln
4	Nine	86 St
5	One	33 Free Way
6	Seven	44 Tree
7	Six	59 Ln
8	Ten	99 St
9	Three	49 Rd
10	Two	45 Dr

purchase:

	cname ▼	product_ID ▼
1	Eight	9
2	Five	7
3	Four	2
4	Nine	1
5	One	1
6	Seven	4
7	Six	2
8	Ten	6
9	Three	5
10	Two	3

Query 4

Example of creating a new account associated with a product:

```
Your Option?
4
Please enter integer product ID:
1
Please enter product account number:
1
Please enter product cost (just enter number without currency):
12
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Tables after creating 10 new accounts:

product:

	product_ID ▾	size ▾	ptype ▾	account_number ▾	established_date ▾	cost ▾
1	1	small	1	1	2022-11-20	12
2	2	medium	2	2	2022-11-20	15
3	3	small	2	3	2022-11-20	14
4	4	large	3	4	2022-11-20	18
5	5	medium	2	5	2022-11-20	14
6	6	small	1	6	2022-11-20	16
7	7	medium	3	7	2022-11-20	17
8	8	small	3	8	2022-11-20	15
9	9	medium	2	9	2022-11-20	13
10	10	small	1	10	2022-11-20	14

Query 5

Example of entering a complaint:

```
Your Option?
5
Please enter integer complaint ID:
1
Please enter customer's name:
Eight
Please enter integer product ID in the complaint:
9
Please enter complaint date (YYYY-MM-DD):
2022-10-10
Please enter complaint description:
Broken
Please enter treatment expected for this complaint:
repair
Connecting to the database...
Done. 1 rows inserted.
```

Tables after entering three complaints:

complaint:

	complaint...	com...	comp...	treatment_exp...
1	1	2022-1...	Broken	repair
2	2	2022-1...	mulfunc...	money back
3	3	2012-1...	not wor...	money back

complain:

	cname	product_ID	complaint_ID
1	Eight	9	1
2	Six	2	2
3	Two	3	3

Query 6

Example of entering an accident:

```
Your Option?
6
Please enter integer accident number:
1
Please enter accident date (YYYY-MM-DD):
2016-02-14
Please enter number of work-day lost:
5
Which product is it? Please enter integer product ID:
10
Was the accident caused by a worker or a technical staff? Enter 1 for worker or 2 for technical staff:
1
Please enter worker's name:
Nobita
Dispatching the query...
Done. 1 rows inserted.
```

Tables after entering 3 accidents:

accident:

	accident_number	accident_date	work_day_lost
1	1	2016-02-14	5
2	2	2012-12-28	4
3	3	2022-09-18	3

mistake:

	accident_number ▼	product_ID ▼	wname ▼	tname ▼
1	1	10	Nobita	NULL
2	2	9	NULL	Game
3	3	3	NULL	James

To show database access is possible, perform 3 queries for each type (7-11), and 1 query for each type (12-15).

Query 7

Examples of query 7:

```
Your Option?
7
Please enter integer product ID:
1
Result for product 1:
Date Produced | Production Time
2022-11-20    | 89.0 Hours
```

```
Your Option?
7
Please enter integer product ID:
3
Result for product 3:
Date Produced | Production Time
2022-09-12    | 75.0 Hours
```

```
Your Option?
7
Please enter integer product ID:
6
Result for product 6:
Date Produced | Production Time
2012-12-12    | 10.0 Hours
```

Query 8

Examples of query 8:

```
Your Option?  
8  
Please enter worker's name:  
Douglas  
Product IDs made by Douglas:  
4  
8
```

```
Your Option?  
8  
Please enter worker's name:  
Nobita  
Product IDs made by Nobita:  
3  
7  
10
```

```
Your Option?  
8  
Please enter worker's name:  
Four  
Product IDs made by Four:  
1  
2  
5  
6  
9
```

Query 9

Examples of query 9:

```
Your Option?  
9  
Please enter quality controller's name:  
Art  
Total number of errors made by Art:  
1
```

```
Your Option?  
9  
Please enter quality controller's name:  
Gun  
Total number of errors made by Gun:  
2
```

We only have two quality controllers, so let's add more quality controller in,

```

Your Option?
1
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
1
Please enter employee's name:
Tim
Please enter employee's address:
34 Tr
Please enter employee's salary:
78222
Please enter quality controller's type (which type of product is the controller allowed
to inspect):
3
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.

```

Now let's check this person,

```

Your Option?
9
Please enter quality controller's name:
Tim
Total number of errors made by Tim:
0

```

0! Because this controller never checked anything yet, so the person had no error.

Query 10

Let's add product 3 that was requested to be repaired by some controller first,

```

Your Option?
2
Please enter integer product ID:
11
Please enter product size (small, medium, or large):
small
Please enter product type (1, 2, or 3):
3
Please enter worker's name that produced the product:
Four
Please enter product's production date (YYYY-MM-DD):
2020-12-14
Please enter time spent to produce this product (in hour):
12
Please enter technical staff's name that repaired the product if applicable (if the
product was not repaired then just press Enter):
James
Please enter product's repair date (YYYY-MM-DD):
2020-12-15
Please enter quality controller's name that inspected the product if applicable (if the
product was not inspected then just press Enter):
Tim
Please enter product's weight (in pound):
13
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.

```

And create its corresponding account:

```
Your Option?
4
Please enter integer product ID:
11
Please enter product account number:
11
Please enter product cost (just enter number without currency):
18
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Examples of query 10:

```
Your Option?
10
Please enter quality controller's name:
Art
Total cost of the repaired products inspected by Art is:
null
```

```
Your Option?
10
Please enter quality controller's name:
Gun
Total cost of the repaired products inspected by Gun is:
null
```

```
Your Option?
10
Please enter quality controller's name:
Tim
Total cost of the repaired products inspected by Tim is:
18.0
```

Query 11

Examples:

```
Your Option?
11
Please enter product color:
red
All customers who purchased all products of a red color:
Eight
Two
```

```
Your Option?
11
Please enter product color:
blue
All customers who purchased all products of a blue color:
Three
```

```
Your Option?  
11  
Please enter product color:  
green  
All customers who purchased all products of a green color:  
Four  
Six
```

Query 12

Examples:

```
Your Option?  
12  
Please enter a salary of your choice (just number without $ sign):  
50000  
All employees whose salary is above $50000.000000:  
Art  
Douglas  
Four  
Game  
Gun  
James  
Nobita  
Tim
```

```
Your Option?  
12  
Please enter a salary of your choice (just number without $ sign):  
80000  
All employees whose salary is above $80000.000000:  
Art  
Four  
Gun  
James  
Nobita
```

Query 13

Example:

```
Your Option?  
13  
Connecting to the database...  
Dispatching the query...  
total number of work days lost due to accidents in repairing the products which got complaints:  
7 days
```

Query 14

Example:

```
Your Option?  
14  
Please enter a year of your choice:  
2022  
The average cost of all products made in a year 2022:  
$14.6
```

Query 15

Example:

```
Your Option?  
15  
Please enter lower boundary date (YYYY-MM-DD):  
2016-01-01  
Please enter upper boundary date (YYYY-MM-DD):  
2016-12-30  
Deleting Accidents from 2016-01-01 to 2016-12-30:  
Done. 1 rows deleted.
```

Affected tables:

accident:

	accident_...	accident_d...	work...
1	2	2012-12-28	4
2	3	2022-09-18	3

mistake:

	accident_...	product_ID	wname	tname
1	2	9	NULL	Game
2	3	3	NULL	James

To show the Import and Export facilities are available, run each option (16-17) once.

Query 16

Import:

```

Your Option?
16
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
1
Please enter file name to import:
quality_controller.csv
Done. 2 rows inserted.

Your Option?
16
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
2
Please enter file name to import:
worker.csv
Done. 2 rows inserted.

Your Option?
16
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
3
Please enter file name to import:
technical_staff.csv
Done. 2 rows inserted.

```

	qname	qaddress	qsal...	qtype
1	Art	1 Lane	89000	1
2	Daf	123 Lane	111001	1
3	Gun	55 St	89000	2
4	Jane	224 rd	111000	3
5	Tim	34 Tr	78222	3

	wname	waddress	wsalary	max_no_products
1	Bob	123 Lane	111001	5
2	Douglas	43 Dr	78000	6
3	Four	34 Rd	90000	10
4	Hick	224 rd	111000	8
5	Nobita	76 St	88000	9


	tname ▼	taddress ▼	tsalary ▼	highest_degree ▼	tech_position ▼
1	Game	22 Ave	79000	BS	Manager
2	James	490 Norman Road	95200	MS	Director
3	Kool	123 Lane	111001	BS	lead
4	Rice	224 rd	111000	BS	staff

Query 17

Export:

```
Your Option?
17
Please enter product color:
red
Please enter file name to export:
customer_red.csv
Exported all customers who purchased all products of a red color to customer_red.csv
```

Exported file:

```
D: > OneDrive - University of Oklahoma > Nasri > Database Management Systems > Project >  customer_red.csv
1  cname
2  Eight
3  Two
```

To show the Quit option is available, run option (18) at least once

Query 18

```
Your Option?
18
Exiting... See you next time!
```


To demonstrate that Azure SQL Database can detect errors, perform 3 queries of different types that contain some errors

Errors

```
Your Option?
1
Please choose the type of employee (enter either 1, 2, or 3):
1) Quality Controller
2) Worker
3) Technical Staff
1
Please enter employee's name:
Jane
Please enter employee's address:
34 Rd
Please enter employee's salary:
68000
Please enter quality controller's type (which type of product is the controller allowed to inspect):
1
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__quality___1FF303F1758000D5'. Cannot insert duplicate key in object 'dbo.quality_controller'. The duplicate key value is (Jane).
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedStatement.java:620)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStatExecCmd.doExecute(SQLServerPreparedStatement.java:540)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.TDSCCommand.execute(IOBuffer.java:7627)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3912)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:268)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:242)
    at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.java:486)
    at Binsaleh.Nasri_IP.main(Binsaleh_Nasri_IP.java:157)
```

The above error showed primary key constraint error when existing employee is being inserted to the database.

```
Your Option?
3
Please enter customer's name:
Frost
Is this an existing customer? (yes/no):
no
Please enter customer's address:
445
Inserting customer to the database...
Done. 1 rows inserted.

Please enter product ID of the purchased product:
50
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: The INSERT statement conflicted with the FOREIGN KEY constraint "FK__purchase__produc__2512604C". The conflict occurred in database "cs-dsa-4513-sql-db", table "dbo.product", column 'product_ID'.
```

The above error is the foreign key error, where the customer is purchasing a non-existing product.

```

Your Option?
2
Please enter integer product ID:
15
Please enter product size (small, medium, or large):
small
Please enter product type (1, 2, or 3):
1
Please enter worker's name that produced the product:
Douglas
Please enter product's production date (YYYY-MM-DD):
2022-05-18
Please enter time spent to produce this product (in hour):
14
Please enter technical staff's name that repaired the product if applicable (if the product was not repaired then just press Enter):

Please enter quality controller's name that inspected the product if applicable (if the product was not inspected then just press Enter):

Please enter the main software used for the product:
123456789123456789123456789123456789123456789123456789
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: String or binary data would be truncated in table 'cs-dsa-4513-sql-db.dbo.product_1', column 'software'. Truncated value: '123456789123456789123456789123'.

```

The above error is when the input program's name is longer than what is restricted.

Task 7. Web database application

7.1 Source Codes and Application

Data Handler:

```

package jsp_azure_test;
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 = "bins000-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "bins000";
    private String password = "<databaseFall22!>";

    // Resulting connection string
    final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;
trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeou
t=30;",

                server, database, username, password);

    // Initialize and save the database connection
    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;

```

```

    }

    this.conn = DriverManager.getConnection(url);
}

// Return the result of selecting everything from the movie_night table
public ResultSet query12() throws SQLException {
    getDBConnection();

    final String sqlQuery = "EXEC case12 @salary = ?;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    return stmt.executeQuery();
}

// Inserts a record into the movie_night table with the given attribute values
public ResultSet getEmployee(Float salary) throws SQLException {

    getDBConnection(); // Prepare the database connection

    // Prepare the SQL statement
    final String sqlQuery = "EXEC case12 @salary = ?;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

    // Replace the '?' in the above statement with the given attribute values
    stmt.setFloat(1, salary);

    return stmt.executeQuery();
}

// Return the result of selecting everything from the employees table
public ResultSet getAllEmployee() throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM worker;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    return stmt.executeQuery();
}

// Inserts a record into the movie_night table with the given attribute values
public boolean addWorker(
    String wname, String waddress, String wsalary, String max_no_products)
throws SQLException {

    getDBConnection(); // Prepare the database connection

    // Prepare the SQL statement
    final String sqlQuery =
        "INSERT INTO worker " +
        "(wname, waddress, wsalary, max_no_products) " +
        "VALUES " +
        "(?, ?, ?, ?)";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

    // Replace the '?' in the above statement with the given attribute values

```

```

        stmt.setString(1, wname);
        stmt.setString(2, waddress);
        stmt.setString(3, wsalary);
        stmt.setString(4, max_no_products);

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

    // Inserts a record into the movie_night table with the given attribute values
    public boolean addQualityController(
        String qname, String qaddress, String qsalary, String qtype) throws
        SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
            "INSERT INTO quality_controller " +
            "(qname, qaddress, qsalary, qtype) " +
            "VALUES " +
            "(?, ?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, qname);
        stmt.setString(2, qaddress);
        stmt.setString(3, qsalary);
        stmt.setString(4, qtype);

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

    // Inserts a record into the movie_night table with the given attribute values
    public boolean addTechnicalStaff(
        String tname, String taddress, String tsalary, String highest_degree,
        String tech_position) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
            "INSERT INTO technical_staff " +
            "(tname, taddress, tsalary, highest_degree, tech_position) " +
            "VALUES " +
            "(?, ?, ?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, tname);
        stmt.setString(2, taddress);
        stmt.setString(3, tsalary);

```

```

        stmt.setString(4, highest_degree);
        stmt.setString(5, tech_position);

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

```

Adding Employee Form code:

```

package jsp_azure_test;
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 = "bins0000-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "bins0000";
    private String password = "<databaseFall22!>";

    // Resulting connection string
    final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;
trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeou
t=30;",

                server, database, username, password);

    // Initialize and save the database connection
    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;
        }

        this.conn = DriverManager.getConnection(url);
    }

    // Return the result of selecting everything from the movie_night table
    public ResultSet query12() throws SQLException {
        getDBConnection();

        final String sqlQuery = "EXEC case12 @salary = ?;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
        return stmt.executeQuery();
    }
}

```

```

// Inserts a record into the movie_night table with the given attribute values
public ResultSet getEmployee(Float salary) throws SQLException {

    getDBConnection(); // Prepare the database connection

    // Prepare the SQL statement
    final String sqlQuery = "EXEC case12 @salary = ?;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

    // Replace the '?' in the above statement with the given attribute values
    stmt.setFloat(1, salary);

    return stmt.executeQuery();
}

// Return the result of selecting everything from the employees table
public ResultSet getAllEmployee() throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM worker;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    return stmt.executeQuery();
}

// Inserts a record into the movie_night table with the given attribute values
public boolean addWorker(
    String wname, String waddress, String wsalary, String max_no_products)
throws SQLException {

    getDBConnection(); // Prepare the database connection

    // Prepare the SQL statement
    final String sqlQuery =
        "INSERT INTO worker " +
        "(wname, waddress, wsalary, max_no_products) " +
        "VALUES " +
        "(?, ?, ?, ?)";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

    // Replace the '?' in the above statement with the given attribute values
    stmt.setString(1, wname);
    stmt.setString(2, waddress);
    stmt.setString(3, wsalary);
    stmt.setString(4, max_no_products);

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

// Inserts a record into the movie_night table with the given attribute values
public boolean addQualityController(
    String qname, String qaddress, String qsalary, String qtype) throws
SQLException {

```

```

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
            "INSERT INTO quality_controller " +
            "(qname, qaddress, qsalary, qtype) " +
            "VALUES " +
            "(?, ?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, qname);
        stmt.setString(2, qaddress);
        stmt.setString(3, qsalary);
        stmt.setString(4, qtype);

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

    // Inserts a record into the movie_night table with the given attribute values
    public boolean addTechnicalStaff(
        String tname, String taddress, String tsalary, String highest_degree,
        String tech_position) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
            "INSERT INTO technical_staff " +
            "(tname, taddress, tsalary, highest_degree, tech_position) " +
            "VALUES " +
            "(?, ?, ?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, tname);
        stmt.setString(2, taddress);
        stmt.setString(3, tsalary);
        stmt.setString(4, highest_degree);
        stmt.setString(5, tech_position);

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

```

Adding Quality Controller code:

```

<%@ 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>Add Quality Controller</title>
</head>
<body>
<%@page import="jsp_azure_test.DataHandler"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Array"%>
<%
// The handler is the one in charge of establishing the connection.
DataHandler handler = new DataHandler();

// Get the attribute values passed from the input form.
String qname = request.getParameter("qname");
String qaddress = request.getParameter("qaddress");
String qsalary = request.getParameter("qsalary");
String qtype = request.getParameter("qtype");

/*
 * If the user hasn't filled out all the time, movie name and duration. This is
 very simple checking.
 */

// Now perform the query with the data from the form.
boolean success = handler.addQualityController(qname, qaddress, qsalary,
qtype);
if (!success) { // Something went wrong
    %>
        <h2>There was a problem inserting the employee</h2>
    <%
} else { // Confirm success to the user
    %>
        <h2>Employee was successfully inserted.</h2>
    <%
}

%>
</body>
</html>

```

Adding worker code:

```

<%@ 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>Add Worker</title>
</head>
<body>
  <%@page import="jsp_azure_test.DataHandler"%>
  <%@page import="java.sql.ResultSet"%>
  <%@page import="java.sql.Array"%>
  <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

    // Get the attribute values passed from the input form.
    String wname = request.getParameter("wname");
    String waddress = request.getParameter("waddress");
    String wsalary = request.getParameter("wsalary");
    String max_no_products = request.getParameter("max_no_products");

    /*
     * If the user hasn't filled out all the time, movie name and duration. This is
     very simple checking.
     */

    // Now perform the query with the data from the form.
    boolean success = handler.addWorker(wname, waddress, wsalary,
max_no_products);
    if (!success) { // Something went wrong
      %>
        <h2>There was a problem inserting the employee</h2>
      <%
    } else { // Confirm success to the user
      %>
        <h2>Employee was successfully inserted.</h2>
      <%
    }
  %>
</body>
</html>

```

Adding Technical Staff code:

```

<%@ 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>Add Quality Controller</title>
</head>

```

```

<body>
<%@page import="jsp_azure_test.DataHandler"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Array"%>
<%
// The handler is the one in charge of establishing the connection.
DataHandler handler = new DataHandler();

// Get the attribute values passed from the input form.
String tname = request.getParameter("tname");
String taddress = request.getParameter("taddress");
String tsalary = request.getParameter("tsalary");
String highest_degree = request.getParameter("highest_degree");
String tech_position = request.getParameter("tech_position");

/*
 * If the user hasn't filled out all the time, movie name and duration. This is
 very simple checking.
 */

// Now perform the query with the data from the form.
boolean success = handler.addTechnicalStaff(tname, taddress, tsalary,
highest_degree, tech_position);
if (!success) { // Something went wrong
    %>
        <h2>There was a problem inserting the employee</h2>
    <%
} else { // Confirm success to the user
    %>
        <h2>Employee was successfully inserted.</h2>
    <%
}
%>
</body>
</html>

```

Query 12 Form code:

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Retrieve Employee</title>
</head>
<body>
<h2>Retrieve all employees whose salary is above a particular salary</h2>
<!--
    Form for collecting user input for the new employee record.
    Upon form submission, addEmployee.jsp file will be invoked.
-->
<form action="query12.jsp">
    <!-- The form organized in an HTML table for better clarity. -->

```

```

        <table border=1>
            <tr>
                <th colspan="2">Enter Salary</th>
            </tr>
            <tr>
                <td>Salary (without '$' sign):</td>
                <td><div style="text-align: center;">
                    <input type=text name=q12_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>

```

Query 12 code:

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Employees With High Salary</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.DataHandler"%>
        <%@page import="java.sql.ResultSet"%>

        <%

// 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 str_salary = request.getParameter("q12_salary");

float salary = Integer.parseInt(str_salary);

ResultSet resultSet = handler.getEmployee(salary);

%>

```

```

<!-- The table for displaying all the movie records -->
<table cellspacing="2" cellpadding="2" border="1">
    <tr> <!-- The table headers row -->
        <td align="center">
            <h4>Employee Name</h4>
        </td>
    </tr>
    <%
        while(resultSet.next()) { // For each movie_night record returned...
            // Extract the attribute values for every row returned
            final String name = resultSet.getString("name");

            out.println("<tr>"); // Start printing out the new table row
            out.println( // Print each attribute value
                "<td align=\"center\">" + name) ;
            out.println("</tr>");

        }
    %>
</table>
</body>
</html>

```

7.2 Web Application:

Add Employee Form:

← → ↻ ⓘ localhost:8585/jsp_azure_test/AddEmployeeForm.jsp

Add Employee

Enter Worker's Data:	
Worker's Name:	<input type="text"/>
Address:	<input type="text"/>
Salary:	<input type="text"/>
Max number of product this worker can produce in a day:	<input type="text"/>
<input type="button" value="Clear"/>	<input type="button" value="Insert"/>

Enter Quality Controller's Data:	
Quality Controller's Name:	<input type="text"/>
Address:	<input type="text"/>
Salary:	<input type="text"/>
Type of product this quality controller can inspect:	<input type="text"/>
<input type="button" value="Clear"/>	<input type="button" value="Insert"/>

Enter Technical Staff's Data:	
Technical Staff's Name:	<input type="text"/>
Address:	<input type="text"/>
Salary:	<input type="text"/>
Highest Degree (BS, MS, PhD):	<input type="text"/>
Technical Position:	<input type="text"/>
<input type="button" value="Clear"/>	<input type="button" value="Insert"/>

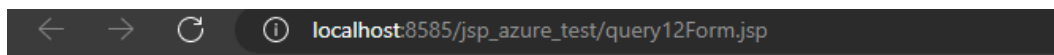
In the adding employee interface, the user can choose to enter information of any type of employee of their choice. Each insert button is independent of each other.

Once the employee is added, the confirmation page can be seen as follow,



Employee Added

Query 12 Form:



Retrieve all employees whose salary is above a particular salary

Enter Salary	
Salary (without '\$' sign):	<input type="text"/>
<input type="button" value="Clear"/>	<input type="button" value="Insert"/>

Here the user can input a salary and retrieve the names of the employees that earn more than the input salary.