NAME: DAVID NNAMDI

STUDENT ID: 113449330

EMAIL: nnamdidavid.n@ou.edu

ASSIGNMENT: INDIVIDUAL PROJECT

COURSE: CS/DSA 4513 – DATABASE MANAGEMENT

SECTION: ONLINE 995-999

SEMESTER: FALL 2022

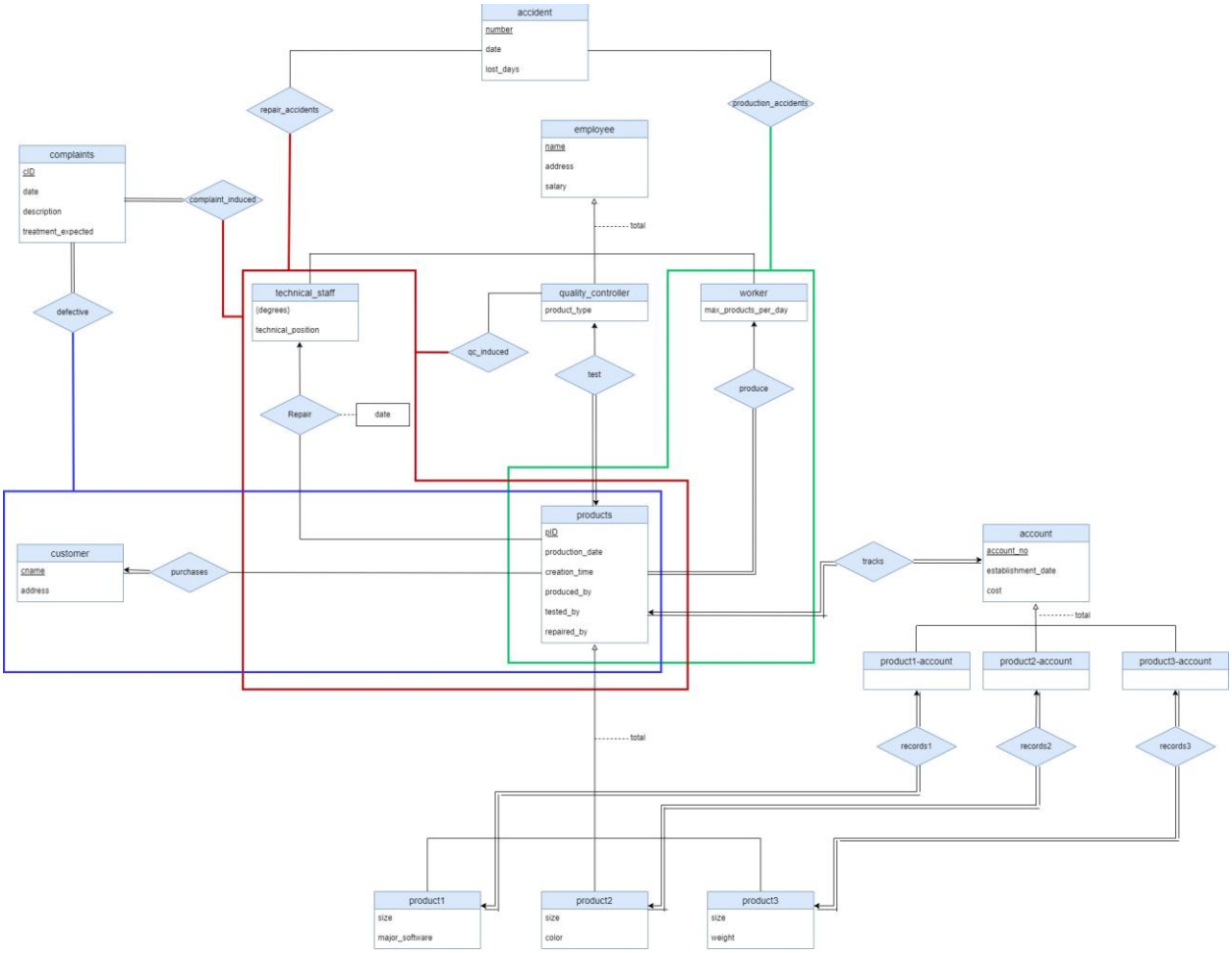
INSTRUCTOR: DR. LE GRUENWALD

SCORE:

Tasks Performed	**Page Number**

Task 1

1.1 ER Diagram



1.2 Relational Database Schema

Employee (<u>name</u>, address, salary)

Technical_staff(<u>name</u>, technical_position)

Technical_staff_degree(<u>name, degree</u>)

Quality_controller(<u>name</u>, product_type)

Worker(<u>name</u>, max_products_per_day)

Products(<u>pID</u>, production_date, creation_time, produced_by, tested_by, repaired_by)

Product1(<u>pID</u>, size, major_software)

Product2(<u>pID</u>, size, color)

Product3(<u>pID</u>, size, weight)

Customer(cname, pID, address)

Complaints(cID, date, description, treatment_expected, cname, pID)

Repair(pID, tech_staff_name, cID, qc_name, date)

Account(account_no, establishment_date, cost, pID)

Product1-account(account_no, pID)

Product2-account(account_no, pID)

Product3-account(account_no, pID)

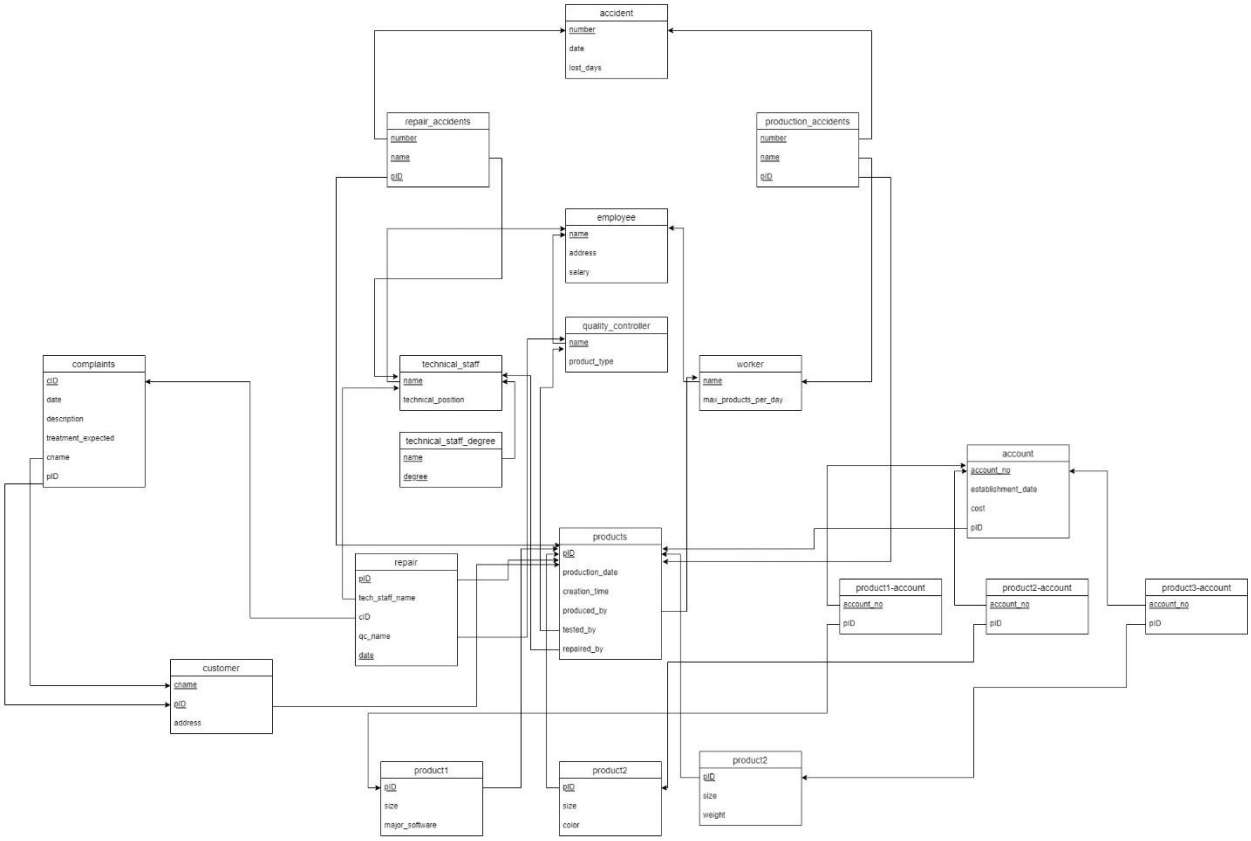
Accident(number, date, lost_days)

Repair_accidents(number, name, pID)

Production_accidents(number, name, pID)

Task 2

Schema Diagram



Task 3

3.1 Discussion of storage structures for tables

Table Name	Query # & Type	Search Key	Query Frequency	Selected File Organization	Justifications
employee	(1) Insert, (12) Range Search	salary	(2/month) (1/month)	Indexed sequential file on search key salary	It is a range search and the frequency of search is low
technical_staff	(1) Insert		(2/month)	Heap	files are only inserted, no other operation perfomed
technical_staff_degree	(1) Insert, (2) Random Search	degree	(2/month) (400/d)	Dynamic hashing with hash key as degree	high frequency random search
quality_controller	(1) Insert, (2) Random Search	name	(2/month) (400/d)	Dynamic hashing with hash key as name	The primary key of the quality controller table is "name", due to high frequency of searh, using a good has function will enable fast searching
worker	(1) Insert		(2/month)	Heap	files are only inserted, no other operation perfomed
products	(2) Insert, (5) Update Random Search, (7) Random Search, (8) Random Search, (9) Random Search, (14) Random Search	pID produced_by tested_by production_date	(400/d) (30/d) (100/d) (2000/d) (400/d) (5/d)	Dynamic hashing with hash key as produced_by	The frequency of running query 8 is very high, having the selected file organization structure allows faster access of this query
product1	(2) Insert, (4) Random Search, (5) Random Search	pID	(400/d) (40/d) (30/d)	Dynamic hashing with hash key as pID	Moderate frequency random search on product ID. Note that Azure SQL already creates a clustered index on pID since it is a primary key
product2	(2) Insert, (4) Random Search, (11) Random Search	pID color	(400/d) (40/d) (5/month)	Dynamic hashing with hash key as pID	Moderate frequency random search on product ID. Note that Azure SQL already creates a clustered index on pID since it is a primary key
product3	(2) Insert		(400/d)	Heap	files are only inserted, no other operation perfomed
customer	(3) Insert		(50/d) (5/month)	Heap	other operations that use customer table involve a join, aside from that no search is performed
complaints	(5) Insert		(30/d)	Heap	files are only inserted, no other operation perfomed
repair	(2) Insert, (5) Insert, (9) Random Search, (10) Random Search, (13) Random Search	pID qc_name cID	(400/d) (30/d) (400/d) (40/d) (1/month)	Dynamic hashing with hash key as pID	High frequency random search on repair using the pID in query 9
accident	(6) Insert, (15) delete Range Search	date	(1/wk) (1/d)	Indexed sequential file on search key date	Most frequent operation is a deletion in query 15 which involves a range search on date
repair_accidents	(6) Insert, (15) Delete Random Search	number	(1/wk) (1/d)	Dynamic hashing with hash key as number	Most frequent operation is a deletion in query 15 which involves a random search on number in a specific data range in accident. Note that Azure SQL already creates a clustered index on number since it is a primary key
production_accidents	(6) Insert, (15) Delete Random Search	number	(1/wk) (1/d)	Dynamic hashing with hash key as number	Most frequent operation is a deletion in query 15 which involves a random search on number in a specific data range in accident. Note that Azure SQL already creates a clustered index on number since it is a primary key
account	(4) Insert		(40/d)	Heap	other operations that use account table involve a join, aside from that no search is performed
product1_account	(4) Insert		(40/d)	Heap	files are only inserted, no other operation perfomed
product2_account	(4) Insert		(40/d)	Heap	files are only inserted, no other operation perfomed
product3_account	(4) Insert		(40/d) (40/d)	Heap	other operations that use product3_account table involve a join, aside from that no search is performed

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

To use dynamic hashing index on SQL we need to create memory optimized tables, the traditional way we create tables is called a disk-based tables. To create a memory-optimized products table on Azure SQL we can use the following syntax:

```
CREATE TABLE products (
    pID VARCHAR(64) PRIMARY KEY NONCLUSTERED,
    production_date DATE,
    creation_time VARCHAR(64),
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64)

    CONSTRAINT FK_produced_by FOREIGN KEY (produced_by) REFERENCES worker,
    CONSTRAINT FK_tested_by FOREIGN KEY (tested_by) REFERENCES quality_controller,
    CONSTRAINT FK_repaired_by FOREIGN KEY (repaired_by) REFERENCES technical_staff
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
```

However, once it is run, I see that based on the Azure subscription, I am not allowed to. Since a higher tier is required for memory optimized tables, no dynamic hashing index can be implemented.

```
Started executing query at Line 1
Msg 40536, Level 16, State 2, Line 63
'MEMORY_OPTIMIZED tables' is not supported in this service tier of the database. See Books Online for more details on feature support in different service tiers of Windows Azure SQ
L Database.
Total execution time: 00:00:00.049
```

Just for reference, the syntax to create a hash index on produced_by in products table is shown below:

```
---creating hash index
ALTER TABLE products
ADD INDEX ix_hash_produced_by NONCLUSTERED
HASH (produced_by) WITH (BUCKET_COUNT = 64);
```

Based on the above, we default to using index-sequential files with primary/secondary indexes created on the search key. Note that if the search key is the primary key of the table, Azure SQL by default creates a unique non-clustered index so we do not need to duplicate this. The code to create all the indexes required is shown below:

```
CREATE INDEX salary_index ON employee(salary);
CREATE INDEX degree_index ON technical_staff_degree(degree);
CREATE INDEX name_index ON quality_controller(name);
CREATE INDEX producedBy_index ON products(produced_by);
CREATE INDEX pID1_index ON product1(pID);
CREATE INDEX pID2_index ON product2(pID);
CREATE INDEX repairPID_index ON repair(pID);
CREATE INDEX date_index ON accident(date);
CREATE INDEX raNumber_index ON repair_accidents(number);
CREATE INDEX paNumber_index ON production_accidents(number);
```

Task 4

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

First, we drop all tables if they exist before creating them, this is standard for table creation

```
--DROP TABLE IF EXISTS

DROP TABLE IF EXISTS product1_account;
DROP TABLE IF EXISTS product2_account;
DROP TABLE IF EXISTS product3_account;
DROP TABLE IF EXISTS account;
DROP TABLE IF EXISTS repair_accidents;
DROP TABLE IF EXISTS production_accidents;
DROP TABLE IF EXISTS accident;
DROP TABLE IF EXISTS repair;
DROP TABLE IF EXISTS constraints;
DROP TABLE IF EXISTS technical_staff_degree;
DROP TABLE IF EXISTS product1;
DROP TABLE IF EXISTS product2;
DROP TABLE IF EXISTS product3;
DROP TABLE IF EXISTS complaints;
DROP TABLE IF EXISTS customer;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS technical_staff;
DROP TABLE IF EXISTS quality_controller;
DROP TABLE IF EXISTS worker;
DROP TABLE IF EXISTS employee;


--CREATE TABLES
CREATE TABLE employee (
    name VARCHAR(64) PRIMARY KEY,
    address VARCHAR(64),
    salary REAL
);
```

	name	address	salary

```
CREATE TABLE technical_staff (
    name VARCHAR(64) PRIMARY KEY,
    technical_position VARCHAR(64)

    CONSTRAINT FK_tname FOREIGN KEY (name) REFERENCES employee
);
```

	name	technical_posit…

```sql
CREATE TABLE technical_staff_degree (
    name VARCHAR(64),
    degree VARCHAR(64)

    PRIMARY KEY (name, degree)
    CONSTRAINT FK_tsname FOREIGN KEY (name) REFERENCES technical_staff
    CONSTRAINT degree_check CHECK (degree IN ('BSC', 'MSC', 'PHD'))

);
```

	name	degree

```sql
CREATE TABLE quality_controller (
    name VARCHAR(64) PRIMARY KEY,
    product_type VARCHAR(64)

    CONSTRAINT FK_qcname FOREIGN KEY (name) REFERENCES employee
    CONSTRAINT producttype_check CHECK (product_type IN ('product1', 'product2', 'product3'))
);
```

	name	product_type

```sql
CREATE TABLE worker (
    name VARCHAR(64) PRIMARY KEY,
    max_products_per_day INT

    CONSTRAINT FK_wname FOREIGN KEY (name) REFERENCES employee
);
```

	name	max_products_pe…

```sql
CREATE TABLE products (
    pID VARCHAR(64) PRIMARY KEY,
    production_date DATE,
    creation_time VARCHAR(64),
    produced_by VARCHAR(64) NOT NULL,
    tested_by VARCHAR(64) NOT NULL,
    repaired_by VARCHAR(64)

    CONSTRAINT FK_produced_by FOREIGN KEY (produced_by) REFERENCES worker,
    CONSTRAINT FK_tested_by FOREIGN KEY (tested_by) REFERENCES quality_controller,
    CONSTRAINT FK_repaired_by FOREIGN KEY (repaired_by) REFERENCES technical_staff
);
```

	pID	production_date	creation_time	produced_by	tested_by	repaired_by

```sql
CREATE TABLE product1 (
    pID VARCHAR(64) PRIMARY KEY,
    size VARCHAR(64),
    major_software VARCHAR(64)

    CONSTRAINT FK_pID1 FOREIGN KEY (pID) REFERENCES products,
    CONSTRAINT p1_size_check CHECK (size IN ('small', 'medium', 'large'))
);
```

	pID	size	major_software

```sql
CREATE TABLE product2 (
    pID VARCHAR(64) PRIMARY KEY,
    size VARCHAR(64),
    color VARCHAR(64)

    CONSTRAINT FK_pID2 FOREIGN KEY (pID) REFERENCES products
    CONSTRAINT p2_size_check CHECK (size IN ('small', 'medium', 'large'))
);
```

	pID	size	color

```sql
CREATE TABLE product3 (
    pID VARCHAR(64) PRIMARY KEY,
    size VARCHAR(64),
    weight REAL

    CONSTRAINT FK_pID3 FOREIGN KEY (pID) REFERENCES products
    CONSTRAINT p3_size_check CHECK (size IN ('small', 'medium', 'large'))
);
```

	pID	size	weight

```sql
CREATE TABLE customer (
    cname VARCHAR(64),
    pID VARCHAR(64),
    address VARCHAR(64)

    PRIMARY KEY (cname, pID)
    CONSTRAINT FK_cPID FOREIGN KEY (pID) REFERENCES products

);
```

	cname	pID	address

```sql
CREATE TABLE complaints (
    cID VARCHAR(64) PRIMARY KEY,
    date DATE,
    description VARCHAR(64),
    treatment_expected VARCHAR(64),
    cname VARCHAR(64) NOT NULL,
    pID VARCHAR(64) NOT NULL

    CONSTRAINT FK_customer FOREIGN KEY (cname, PID) REFERENCES customer
);
```

	cID	date	description	treatment_expec…	cname	pID

```sql
CREATE TABLE repair (
    pID VARCHAR(64),
    tech_staff_name VARCHAR(64),
    cID VARCHAR(64),
    qc_name VARCHAR(64),
    date DATE

    PRIMARY KEY (pID, date)
    CONSTRAINT FK_rpID FOREIGN KEY (pID) REFERENCES products,
    CONSTRAINT FK_tsrname FOREIGN KEY (tech_staff_name) REFERENCES technical_staff,
    CONSTRAINT FK_qcrname FOREIGN KEY (qc_name) REFERENCES quality_controller,
    CONSTRAINT FK_cIDr FOREIGN KEY (cID) REFERENCES complaints
);
```

	pID	tech_staff_name	cID	qc_name	date

```sql
CREATE TABLE accident (
    number INT PRIMARY KEY,
    date DATE,
    lost_days REAL

);
```

	number	date	lost_days

```sql
CREATE TABLE repair_accidents (
    number INT,
    name VARCHAR(64),
    pID VARCHAR (64)

    PRIMARY KEY (number, name, pID)
    CONSTRAINT FK_ranumber FOREIGN KEY (number) REFERENCES accident,
    CONSTRAINT FK_raname FOREIGN KEY (name) REFERENCES technical_staff,
    CONSTRAINT FK_rapID FOREIGN KEY (pID) REFERENCES products
);
```

	number	name	pID

```
CREATE TABLE production_accidents (
    number INT,
    name VARCHAR(64),
    pID VARCHAR (64)

    PRIMARY KEY (number, name, pID)
    CONSTRAINT FK_panumber FOREIGN KEY (number) REFERENCES accident,
    CONSTRAINT FK_paname FOREIGN KEY (name) REFERENCES worker,
    CONSTRAINT FK_papID FOREIGN KEY (pID) REFERENCES products
);
```

	number	name	pID

```
CREATE TABLE account (
    account_no VARCHAR(64) PRIMARY KEY,
    establishment_date DATE,
    cost REAL,
    pID VARCHAR(64)

    CONSTRAINT FK_apID FOREIGN KEY (pID) REFERENCES products
);
```

	account_no	establishment_d…	cost	pID

```
CREATE TABLE product1_account (
    account_no VARCHAR(64) PRIMARY KEY,
    pID VARCHAR(64)

    CONSTRAINT FK_p1acc FOREIGN KEY (account_no) REFERENCES account,
    CONSTRAINT FK_p1apID FOREIGN KEY (pID) REFERENCES product1
);
```

	account_no	pID

```
CREATE TABLE product2_account (
    account_no VARCHAR(64) PRIMARY KEY,
    pID VARCHAR(64)

    CONSTRAINT FK_p2acc FOREIGN KEY (account_no) REFERENCES account,
    CONSTRAINT FK_p2apID FOREIGN KEY (pID) REFERENCES product2
);
```

	account_no	pID

```sql
CREATE TABLE product3_account (
    account_no VARCHAR(64) PRIMARY KEY,
    pID VARCHAR(64)

    CONSTRAINT FK_p3acc FOREIGN KEY (account_no) REFERENCES account,
    CONSTRAINT FK_p3apID FOREIGN KEY (pID) REFERENCES product3
);
```

account_no	pID

Task 5

5.1 SQL statements (and T-SQL stored procedures) implementing all queries 1 - 15

```sql
--QUERIES DEFINED IN QUESTION
DROP PROCEDURE IF EXISTS insert_new_employee;
DROP PROCEDURE IF EXISTS insert_new_product;
DROP PROCEDURE IF EXISTS insert_new_customer;
DROP PROCEDURE IF EXISTS create_new_account;
DROP PROCEDURE IF EXISTS create_new_complaint;
DROP PROCEDURE IF EXISTS enter_new_accident;
DROP PROCEDURE IF EXISTS retrieve_q7;
DROP PROCEDURE IF EXISTS retrieve_q8;
DROP PROCEDURE IF EXISTS retrieve_q9;
DROP PROCEDURE IF EXISTS retrieve_q10;
DROP PROCEDURE IF EXISTS retrieve_q11;
DROP PROCEDURE IF EXISTS retrieve_q12;
DROP PROCEDURE IF EXISTS retrieve_q13;
DROP PROCEDURE IF EXISTS retrieve_q14;
DROP PROCEDURE IF EXISTS retrieve_q15;
```

```sql
--1 enter a new employee
GO
CREATE PROCEDURE insert_new_employee --create a procedure for inserting a new employee
    -- specify the parameters needed by the procedure
    @name VARCHAR(64), --the new employee name
    @address VARCHAR(64), --the new employee address
    @salary REAL, --the new employee salary
    @emptype VARCHAR(64), -- the employee type
    @addl_info VARCHAR(64), -- additional information for the employee subgroups
    @addl_info2 VARCHAR(64) -- degree information if the employee is technical staff
AS
BEGIN
    --write insert sql statement for procedure considering the salary calcuations using case
statements
    IF @emptype NOT IN ('technical_staff', 'quality_controller', 'worker') -- check to see
that user entered the right type of employee for insertion into employee subgroup table
    BEGIN
```

```
        PRINT 'ERROR: Incorrect Entry of employee type, please enter "technical_staff",
"quality_controller" or "worker"'
        RETURN;
    END


    -- peform insertions
    INSERT INTO employee
        (name, address, salary)
    VALUES
        (@name, @address, @salary);


    if @emptype = 'technical_staff'
    BEGIN
        INSERT INTO technical_staff
            (name, technical_position)
        VALUES
            (@name, @addl_info);

        INSERT INTO technical_staff_degree
            (name, degree)
        VALUES
            (@name, @addl_info2);
    END
    ELSE IF @emptype = 'quality_controller'
    BEGIN
        INSERT INTO quality_controller
            (name, product_type)
        VALUES
            (@name, @addl_info);
    END
    ELSE
    BEGIN
        INSERT INTO worker
            (name, max_products_per_day)
        VALUES
            (@name, @addl_info);
    END
END

--2 insert new product
GO
CREATE PROCEDURE insert_new_product --create a procedure for inserting a new product
    -- specify the parameters needed by the procedure
    @id VARCHAR(64), --the new product id
    @date DATE, -- date the product was made
    @time VARCHAR(64), -- how long it took to create the product
    @produced_by VARCHAR(64), -- name of worker who made the product
    @tested_by VARCHAR(64), -- name of quality controller who tested the product
    @repaired_by VARCHAR(64), -- name of technical staff who repaired the product, if any
    @producttype VARCHAR(64), -- type of product
    @size VARCHAR(64), -- size of product
```

```
    @addl_info VARCHAR(64) --major software for product 1, color for product 2, weight for
product 3
AS
BEGIN
    --write insert sql statement for procedure considering the salary calcuations using case
statements
    IF @producttype NOT IN ('Product1', 'Product2', 'Product3') -- check that user entered
correct product type for insertion into subgroup tables
    BEGIN
        PRINT 'ERROR: The product type is incorrectly specified, please enter "Product1",
"Product2" or "Product3"'
        RETURN;
    END

    IF @producttype != (SELECT product_type from quality_controller where name = @tested_by) -
- check that the product type can be tested by the entered quality controller name
    BEGIN
        PRINT 'ERROR: This product cannot be tested by the quality controller entered'
        RETURN;
    END

    IF @producttype = 'Product1' AND @repaired_by NOT IN (SELECT name from
technical_staff_degree WHERE degree in ('MSC', 'PHD')) --check that the product can be
repaired by the specified technical staff based on its type
    BEGIN
        PRINT 'ERROR: The person said to have repaired the product does not have the required
qualifications to do so'
        RETURN;
    END

    -- perform the inserts once all the input data has been confirmed
    IF @repaired_by = ''
    BEGIN
        SET @repaired_by = NULL
    END

    INSERT INTO products
        (pID, production_date, creation_time, produced_by, tested_by, repaired_by)
    VALUES
        (@id, @date, @time, @produced_by, @tested_by, @repaired_by);


    IF @producttype = 'Product1'
    BEGIN
        INSERT INTO product1
            (pID, size, major_software)
        VALUES
            (@id, @size, @addl_info);
    END
    ELSE IF @producttype = 'Product2'
    BEGIN
        INSERT INTO product2
            (pID, size, color)
        VALUES
```

```
            (@id, @size, @addl_info);
    END
    ELSE
    BEGIN
        INSERT INTO product3
            (pID, size, weight)
        VALUES
            (@id, @size, @addl_info);
    END
    -- assume all repairs when entering new product was ordered by quality controller and
insert in repair table
    --we assume that the product was repaired 1 day after it was produced and tested by
quality controller,
    --this is reflected in the insert statement into the repair table
    IF @repaired_by IS NOT NULL
    BEGIN
        INSERT INTO repair
            (pID, tech_staff_name, qc_name, date)
        VALUES
            (@id, @repaired_by, @tested_by, (SELECT CONVERT(VARCHAR(64), DATEADD(DAY, 1,
@date), 101)))
    END
END


--3 Enter a customer associated with some product
GO
CREATE PROCEDURE insert_new_customer --create a procedure for inserting a new product
    -- specify the parameters needed by the procedure
    @cname VARCHAR(64), --customer name
    @pID VARCHAR(64), -- product purchased
    @address VARCHAR(64) --customre address
AS
BEGIN
    --write insert sql statement for procedure considering the salary calcuations using case
statements

    INSERT INTO customer
        (cname, pID, address)
    VALUES
        (@cname, @pID, @address);
END

--4 create a new account associated with a product
GO
CREATE PROCEDURE create_new_account --create a procedure for inserting a new product
    -- specify the parameters needed by the procedure
    @account_no VARCHAR(64),
    @date DATE, -- date account was established
    @cost REAL, -- cost of product
    @pID VARCHAR(64) -- product ID
AS
BEGIN
```

```
    --write insert sql statement for procedure considering the salary calcuations using case
statements
    INSERT INTO account
        (account_no, establishment_date, cost, piD)
    VALUES
        (@account_no, @date, @cost, @pID);

    IF (SELECT pID FROM product1 WHERE pID = @pID) = @pID
    BEGIN
        INSERT INTO product1_account
            (account_no, pID)
        VALUES
            (@account_no, @pID)
    END
    ELSE
    BEGIN
        IF (SELECT pID FROM product2 WHERE pID = @pID) = @pID
        BEGIN
            INSERT INTO product2_account
                (account_no, pID)
            VALUES
                (@account_no, @pID)
        END
        ELSE
        BEGIN
            INSERT INTO product3_account
                (account_no, pID)
            VALUES
                (@account_no, @pID)
        END
    END
END

--5 Enter a complaint associated with a customer and product
GO
CREATE PROCEDURE create_new_complaint --create a procedure for inserting a new product
    -- specify the parameters needed by the procedure
    @cID VARCHAR(64), --complaint ID
    @date DATE, -- Date the complaint was made
    @desc VARCHAR(64), -- description of what is wrong with product
    @treatment_expected VARCHAR(64), --refund or replace the product?
    @cname VARCHAR(64), --customer name
    @pID VARCHAR(64) -- product being complained about
AS
BEGIN
    --perfom inserts
    INSERT INTO complaints
        (cID, date, description, treatment_expected, cname, pID)
    VALUES
        (@cID, @date, @desc, @treatment_expected, @cname, @pID);

    -- regardless of if a customer wants a refund or a replacement product, the product has to
been sent back for repairs
    -- hence we need to know who will be repairing this product
```

```
    --This section randomly assigns a techical staff to handle the repair based on the product
type
    DECLARE @tech_staff_name VARCHAR(64)

    IF (SELECT pID from product1 where pID = @pID) = @pID
    BEGIN
        SET @tech_staff_name = (SELECT top 1 name from technical_staff_degree WHERE degree IN
('MSC', 'PHD') ORDER BY NEWID())
    END
    ELSE
    BEGIN
        SET @tech_staff_name = (SELECT top 1 name from technical_staff order by NEWID())
    END
    --insert into repair based on complaint
    --we the repair is done 4 days after complaint is made (3 days to ship the product back
and 1 day to repair).
    --note that the primary assumption here is that every product that is complained about
must be returned, the customer will either
    --get a new one or their money back. and for each product returned it is repaired. We
randomly assign compained product to
    --a technical staff based on the product type, i.e product 1 will only be repaired by MSC
and PHD holders, product 2 & 3 by anyone else
    INSERT INTO repair
            (pID, tech_staff_name, cID, date)
        VALUES
            (@pID, @tech_staff_name, @cID, (SELECT CONVERT(VARCHAR(64), DATEADD(DAY, 4,
@date), 101))); --assume it takes 3 days to return product and 1 day to repair

    --update products table with the tech staff who completed complaints based repair
    UPDATE products
        SET repaired_by = @tech_staff_name
        WHERE pID = @pID;
END

--6 Enter an accident associated with an employee and product
GO
CREATE PROCEDURE enter_new_accident --create a procedure for inserting a new product
    -- specify the parameters needed by the procedure
    @number INT, --unique accident number
    @date DATE, --date accident occured
    @lost_days REAL, -- no of days lost due to accident
    @accidenttype VARCHAR(64), -- what type of accident?
    @name VARCHAR(64), -- name of employee involved in accident
    @pID VARCHAR(64) -- product being worked on when accident occured
AS
BEGIN
    IF @accidenttype NOT IN ('repair', 'production') -- check that the accident type is
correctly entered
    BEGIN
        PRINT 'ERROR: Incorrect entry of accident type, please enter "repair" or "production"'
        RETURN;
    END
```

```
    IF @accidenttype = 'repair' AND (@name NOT IN (SELECT tech_staff_name FROM repair) OR @pID
NOT IN (SELECT pID FROM repair)) -- check that the technical staff and product being repaired
exists in the repair table
    BEGIN
        PRINT 'ERROR: A repair by the technical staff for the specified product does not exist
in database'
        RETURN;
    END

    --perform inserts
    IF @accidenttype = 'repair'
    BEGIN
        INSERT INTO accident
            (number, date, lost_days)
        VALUES
            (@number, @date, @lost_days);

        INSERT INTO repair_accidents
            (number, name, pID)
        VALUES
            (@number, @name, @pID)
    END
    ELSE IF @accidenttype = 'production'
    BEGIN
        INSERT INTO accident
            (number, date, lost_days)
        VALUES
            (@number, @date, @lost_days);

        INSERT INTO production_accidents
            (number, name, pID)
        VALUES
            (@number, @name, @pID)
    END
END


--7 Retrieve the date produced and time spent to produce a particular product
GO
CREATE PROCEDURE retrieve_q7
    -- specify the parameters needed by the procedure
    @pID VARCHAR(64) -- product ID you want to get production data and creation time for
AS
BEGIN
    SELECT production_date, creation_time
    FROM products
    WHERE pID = @pID;
END

--8 Retrieve all products made by a particular worker
GO
CREATE PROCEDURE retrieve_q8
    -- specify the parameters needed by the procedure
    @workername VARCHAR(64) -- worker name that you want to get all products made by them
```

```sql
AS
BEGIN
    SELECT pID
    FROM products
    WHERE produced_by = @workername;
END

--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
GO
CREATE PROCEDURE retrieve_q9
    -- specify the parameters needed by the procedure
    @qcname VARCHAR(64) --quality controller name
AS
BEGIN
    SELECT COUNT(pID) AS total_errors
    FROM products
    WHERE tested_by = @qcname
    AND pID IN ((SELECT pID FROM repair where qc_name IS NULL) EXCEPT (SELECT pID FROM repair
WHERE qc_name IS NOT NULL))
END


--10 Retrieve the total costs of the products in product3 category which were repaired at the
request of a particular quality controller
GO
CREATE PROCEDURE retrieve_q10
    -- specify the parameters needed by the procedure
    @qcname VARCHAR(64) -- quality controller name
AS
BEGIN
    SELECT SUM(A.cost) as total_cost
    FROM account A, product3_account P, repair R
    WHERE A.account_no = P.account_no
    AND P.pID = R.pID
    AND R.qc_name = @qcname;
END

--11 Retrieve all customers (in name order) who purchased all products of a particular color
GO
CREATE PROCEDURE retrieve_q11
    -- specify the parameters needed by the procedure
    @color VARCHAR(64) --product color
AS
BEGIN
    SELECT C.cname
    FROM customer C, product2 P
    WHERE C.pID = P.pID
    AND P.color = @color
    ORDER BY C.cname;
END
```

```sql
--12 Retrieve all employees whose salary is above a particular salary
GO
CREATE PROCEDURE retrieve_q12
    -- specify the parameters needed by the procedure
    @salary REAL -- specified salary
AS
BEGIN
    SELECT name
    FROM employee
    WHERE salary > @salary;
END


--13 Retrieve the total number of work days lost due to accidents in repairing the products
which got complaints
GO
CREATE PROCEDURE retrieve_q13
AS
BEGIN
    SELECT SUM(A.lost_days) AS total_lost_days
    FROM accident A, repair_accidents RA, repair R
    WHERE A.number = RA.number
    AND RA.pID = R.pID
    AND R.cID IS NOT NULL;
END


--14 Retrieve the average cost of all products made in a particular year
GO
CREATE PROCEDURE retrieve_q14
    -- specify the parameters needed by the procedure
    @year VARCHAR(64) -- specified year
AS
BEGIN
    SELECT ROUND(AVG(A.cost),3) AS average_cost
    FROM account A, products P
    WHERE A.pID = P.pID
    AND YEAR(P.production_date) = @year;
END

--15 Delete all accidents whose dates are in some range
GO
CREATE PROCEDURE retrieve_q15
    -- specify the parameters needed by the procedure
    @startdate VARCHAR(64), -- specified start date
    @enddate VARCHAR(64) -- specified end date
AS
BEGIN
    DELETE FROM repair_accidents
    WHERE number IN (
        SELECT number FROM accident
        WHERE date between @startdate and @enddate);

    DELETE FROM production_accidents
```

```
    WHERE number IN (
        SELECT number FROM accident
        WHERE date between @startdate and @enddate);

    DELETE FROM accident
    WHERE date between @startdate and @enddate;
END
```

5.2 The Java Source program and screenshots showing successful compilation

```
import java.sql.Connection;
import java.util.ArrayList;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.SQLWarning;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.FileWriter;

public class project {

    // Database credentials
    final static String HOSTNAME = "nnam0000-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "nnam0000";
    final static String PASSWORD = "Onyinye26$$";

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


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

```java
    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")) { // As user for options until option 4 is selected
            System.out.println(PROMPT); // Print the available options
            option = sc.nextLine(); // Read in the user option selection

            switch (option) { // Switch between different options
                case "1": // Insert a new employee

                    // Collect the new employee data from the user
                    System.out.println("Please enter employee type: enter 'technical_staff',
'quality_controller' or 'worker'");
                    final String emptype = sc.nextLine(); // Read in the employee type

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

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

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

                    System.out.println("Please enter additional information for employee:");
                    System.out.println("For technical staff, enter the technical position");
                    System.out.println("For quality controller, enter product type to be checked:
('product1', 'product2', 'product3')");
                    System.out.println("For worker, enter the maximum number of products worker can
produce per day");
                    final String addl_info = sc.nextLine(); // Read in user input of employee additional
info based on type

                    System.out.println("Please enter employee degree if the employee is a technical staff
otherwise press Enter Key:");
                    System.out.println("Enter 'BSC', 'MSC' or 'PHD' for technical staff");
                    final String addl_info2 = sc.nextLine(); // Read in user input of employee degree

                    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("EXEC
insert_new_employee @name = ?, @address = ?, @salary = ?, "
                                                                                                +
"@emptype = ?, @addl_info = ?, @addl_info2 = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setString(1, name);
                            statement.setString(2, address);
                            statement.setFloat(3, salary);
                            statement.setString(4, emptype);
                            statement.setString(5, addl_info);
                            statement.setString(6, addl_info2);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final int rows_inserted = statement.executeUpdate();
                            final SQLWarning warning = statement.getWarnings();
                            if (warning != null)
                                        System.out.println(warning.getMessage());
                            else
```

```
                                System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                            }
                        }
                        catch(SQLException sqe) {
                            System.out.println("Error Message = " + sqe.getMessage());
                        }


                        break;
                    case "2": //Insert a new product

                    // Collect the new faculty data from the user
                    System.out.println("Please enter product type: enter 'Product1', 'Product2' or
'Product3'");
                        final String producttype = sc.nextLine(); // Read in the product type

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

                        System.out.println("Please enter date the product was produced in mm/dd/yyyy:");
                        final String date = sc.nextLine(); // Read in user input of date product was made

                        System.out.println("Please enter time spent to make the product in hours:");
                        final String time = sc.nextLine(); // Read in user input of how long it took to make
product

                        System.out.println("Please enter name of worker who produced the product:");
                        final String produced_by = sc.nextLine(); // Read in user input of worker name who
made product (white-spaces allowed).

                        System.out.println("Please enter name of quality controller who tested the product:");
                        final String tested_by = sc.nextLine(); // Read in user input of quality controller
name who tested product (white-spaces allowed).

                        System.out.println("Please enter name of technical staff who repaired the product:");
                        System.out.println("If product was not repaired, press Enter Key to skip");
                        final String repaired_by = sc.nextLine(); // Read in user input of technical staff
name who repaired product if any (white-spaces allowed).

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

                        System.out.println("Please enter additional information for product:");
                        System.out.println("For Product 1, enter the major software used");
                        System.out.println("For Product 2, enter the color");
                        System.out.println("For Product 3, enter the weight");
                        final String addl_info_q2 = sc.nextLine(); // Read in user input of additional
information for the specific product type


                        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("EXEC
insert_new_product @id = ?, @date = ?, @time = ?, @produced_by = ?, "
                                                                                                    +
"@tested_by = ?, @repaired_by = ?, @producttype = ?, @size = ?, @addl_info = ?;")) {
                                // Populate the query template with the data collected from the user
                                statement.setString(1, id);
                                statement.setString(2, date);
                                statement.setString(3, time);
                                statement.setString(4, produced_by);
                                statement.setString(5, tested_by);
                                statement.setString(6, repaired_by);
                                statement.setString(7, producttype);
                                statement.setString(8, size);
```

```
                            statement.setString(9, addl_info_q2);

                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final int rows_inserted = statement.executeUpdate();
                            final SQLWarning warning = statement.getWarnings();
                            if (warning != null)
                                        System.out.println(warning.getMessage());
                            else
                                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                        }
                    }
                    catch(SQLException sqe) {
                         System.out.println("Error Message = " + sqe.getMessage());
                    }

                    break;
                case "3": //Insert a new customer associated with some products

                // Collect the new faculty data from the user
                System.out.println("Please enter the customer name:'");
                    final String cname = sc.nextLine(); // Read in the customer name (white-spaces
allowed).


                    System.out.println("Please enter the customer address:");
                    final String caddress = sc.nextLine(); // Read in user input of customer address
(white-spaces allowed).

                    System.out.println("Please enter all products purchased by customer, separate each
entry with a space:");
                    System.out.println("When done with entering all products, press Enter and 'end':");
                    ArrayList<String> pidlist = new ArrayList<String>(); // create array list to store all
the user entry of products purchased by customer
                    boolean choice = false;

                    //populate the array list with user input and stop when user enters end keyword
                    while(choice == false){
                        String line = sc.nextLine();
                        if(line.equalsIgnoreCase("end")){
                            break;
                        }
                        else{
                            String[] splitArr = line.split(" "); // split the user entry into individual
products and store in a string array
                            for (String a : splitArr) {
                                 pidlist.add(a);
                            }
                        }
                    }

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

                    int val = 0;

                    //perform insertion into customer table until the products entered are exhausted
                    while (pidlist.size() > val) {
                         String temppid;
                         temppid = pidlist.get(val);



                            // Get a database connection and prepare a query statement
                            try (final Connection connection = DriverManager.getConnection(URL)) {
                                try (
                                    final PreparedStatement statement = connection.prepareStatement("EXEC
insert_new_customer @cname = ?, @pID = ?, "
```

```
+ "@address = ?;")) {
                                        // Populate the query template with the data collected from the user
                                        statement.setString(1, cname);
                                        statement.setString(2, temppid);
                                        statement.setString(3, caddress);


                                        System.out.println("Dispatching the query...");
                                        // Actually execute the populated query
                                        final int rows_inserted = statement.executeUpdate();
                                        final SQLWarning warning = statement.getWarnings();
                                        if (warning != null)
                                                    System.out.println(warning.getMessage());
                                        else
                                             System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                                    }
                                }
                                catch(SQLException sqe) {
                                     System.out.println("Error Message = " + sqe.getMessage());
                                }

                        val++;
                        }

                        break;
                    case "4": //create a new account

                    // Collect the new faculty data from the user
                    System.out.println("Please enter account number:");
                        final String account_no = sc.nextLine(); // Read in user input of the account number

                        System.out.println("Please enter the date the account was established in
mm/dd/yyyy:");
                        final String adate = sc.nextLine(); // Read in user input of the date the account was
established

                        System.out.println("Please enter the id of the product we want to establish an account
for:");
                        final String apid = sc.nextLine(); // Read in user input of product id to establish
account for

                        System.out.println("Please enter cost of product:");
                        final float cost = sc.nextFloat(); // Read in user input of cost of product
                        sc.nextLine();




                        System.out.println("Connecting to the database...");
                        // Get a database connection and prepare a query statement
                        try (final Connection connection = DriverManager.getConnection(URL)) {
                            try (
                                final PreparedStatement statement = connection.prepareStatement("EXEC
create_new_account @account_no = ?, @date = ?, @cost = ?, "
                                                                                                    + "@pID
= ?;")) {
                                // Populate the query template with the data collected from the user
                                statement.setString(1, account_no);
                                statement.setString(2, adate);
                                statement.setFloat(3, cost);
                                statement.setString(4, apid);


                                System.out.println("Dispatching the query...");
                                // Actually execute the populated query
                                final int rows_inserted = statement.executeUpdate();
                                final SQLWarning warning = statement.getWarnings();
```

```
                            if (warning != null)
                                        System.out.println(warning.getMessage());
                            else
                                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                        }
                    }
                    catch(SQLException sqe) {
                         System.out.println("Error Message = " + sqe.getMessage());
                    }

                    break;
                case "5": //Insert a new complaint

                // Collect the new faculty data from the user
                System.out.println("Please enter product complaint id:");
                    final String cID = sc.nextLine(); // Read in user input of complaint id

                    System.out.println("Please enter date of complaint in mm/dd/yyyy:");
                    final String cdate = sc.nextLine(); // Read in user input of date of complaint

                    System.out.println("Please enter name of customer who has filed a complaint:");
                    final String c_cname = sc.nextLine(); // Read in user input of name of customer filing
the complaint

                    System.out.println("Please enter the product id:");
                    final String pid = sc.nextLine(); // Read in user input of the id of the product being
complained about

                    System.out.println("Please enter description of what is wrong with product:");
                    final String desc = sc.nextLine(); // Read in user input of what's wrong with
product(white-spaces allowed).


                    System.out.println("Please enter treament expected for complaint:");
                    System.out.println("Enter 'refund' if customer wants to return product and be
refunded");
                    System.out.println("Enter 'replace' if customer wants the product to be replaced with
a working one");
                    final String treatment_expected = sc.nextLine(); // Read in user input of treatment
expected


                    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("EXEC
create_new_complaint @cID = ?, @date = ?, @desc = ?, @treatment_expected = ?, "
                                                                                                    +
"@cname = ?, @pID = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setString(1, cID);
                            statement.setString(2, cdate);
                            statement.setString(3, desc);
                            statement.setString(4, treatment_expected);
                            statement.setString(5, c_cname);
                            statement.setString(6, pid);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final int rows_inserted = statement.executeUpdate();
                            final SQLWarning warning = statement.getWarnings();
                            if (warning != null)
                                        System.out.println(warning.getMessage());
                            else
                                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
```

```
                        }
                    }
                    catch(SQLException sqe) {
                        System.out.println("Error Message = " + sqe.getMessage());
                    }

                    break;
                case "6": //Insert a new accident

                // Collect the new faculty data from the user
                System.out.println("Please enter the accident type:");
                    System.out.println("Enter 'repair' if accident occured during product repair");
                    System.out.println("Enter 'production' if accident occured during production of
product");
                    final String accidenttype = sc.nextLine(); // Read in user input of accident type

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

                    System.out.println("Please enter date of accident in mm/dd/yyyy:");
                    final String acdate = sc.nextLine(); // Read in user input of the date the accident
occurred

                    System.out.println("Please enter number of days lost due to accident:");
                    final Float lost_days = sc.nextFloat(); // Read in user input of the number of days
lost due to the accident
                    sc.nextLine();

                    System.out.println("Please enter name of employee involved in accident:");
                    final String acname = sc.nextLine(); // Read in user input of employee name involved
in accident (white-spaces allowed).


                    System.out.println("Please enter id of product being worked on during the accident:");
                    final String acpid = sc.nextLine(); // Read in user input of product id being worked
on during the accident


                    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("EXEC
enter_new_accident @number = ?, @date = ?, @lost_days = ?, @accidenttype = ?, "
                                                                                                    +
"@name = ?, @pID = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setInt(1, number);
                            statement.setString(2, acdate);
                            statement.setFloat(3, lost_days);
                            statement.setString(4, accidenttype);
                            statement.setString(5, acname);
                            statement.setString(6, acpid);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final int rows_inserted = statement.executeUpdate();
                            final SQLWarning warning = statement.getWarnings();
                            if (warning != null)
                                        System.out.println(warning.getMessage());
                            else
                                System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                        }
                    }
                    catch(SQLException sqe) {
                        System.out.println("Error Message = " + sqe.getMessage());
```

```
                    }

                    break;
                case "7": //Retrieve the date produced and the time spent to make a product

                // Collect the new faculty data from the user
                System.out.println("Please enter product ID:");
                    final String pID_q7 = sc.nextLine(); // Read in the user input of the product ID



                    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("EXEC
retrieve_q7 @pID = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setString(1, pID_q7);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final ResultSet resultSet = statement.executeQuery();

                            System.out.println("Option 7 query results:");
                            System.out.println("production date | creation time ");

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

                        }
                    }
                    catch(SQLException sqe) {
                         System.out.println("Error Message = " + sqe.getMessage());
                    }

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

                // Collect the new faculty data from the user
                System.out.println("Please enter name of worker:");
                    final String workername = sc.nextLine(); // Read in the user input of the worker name



                    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("EXEC
retrieve_q8 @workername = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setString(1, workername);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final ResultSet resultSet = statement.executeQuery();

                            System.out.println("Option 8 query results:");
                            System.out.println("product id ");

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

```
                    }

                }
            }
            catch(SQLException sqe) {
                System.out.println("Error Message = " + sqe.getMessage());
            }

            break;
        case "9": //Retrieve erroneously certified products by a quality controller

        // Collect the new faculty data from the user
        System.out.println("Please enter name of quality controller:");
            final String qcname = sc.nextLine(); // Read in the user input of the quality
controller name



            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("EXEC
retrieve_q9 @qcname = ?;")) {
                    // Populate the query template with the data collected from the user
                    statement.setString(1, qcname);


                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final ResultSet resultSet = statement.executeQuery();

                    System.out.println("Option 9 query results:");
                    System.out.println("total errors ");

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

                }
            }
            catch(SQLException sqe) {
                System.out.println("Error Message = " + sqe.getMessage());
            }

            break;
        case "10": //Retrieve total costs of product3 type products repaired by a quality
controller

        // Collect the new faculty data from the user
        System.out.println("Please enter name of quality controller:");
            final String qcname2 = sc.nextLine(); // Read in the product type



            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("EXEC
retrieve_q10 @qcname = ?;")) {
                    // Populate the query template with the data collected from the user
                    statement.setString(1, qcname2);


                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final ResultSet resultSet = statement.executeQuery();
```

```
                        System.out.println("Option 10 query results:");
                        System.out.println("total cost ");

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

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

                break;
            case "11": //Retrieve all customers (sorted by name) who purchased product of particular
color

            // Collect the new faculty data from the user
            System.out.println("Please enter color of products:");
                final String color = sc.nextLine(); // Read in the product 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("EXEC
retrieve_q11 @color = ?;")) {
                        // Populate the query template with the data collected from the user
                        statement.setString(1, color);


                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final ResultSet resultSet = statement.executeQuery();

                        System.out.println("Option 11 query results:");
                        System.out.println("customer name ");

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

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

                break;
            case "12": //Retrieve all employees with salary above a particular salary

            // Collect the new faculty data from the user
            System.out.println("Please enter target salary:");
                final Float salary_q12 = sc.nextFloat(); // Read in the particular salary
                sc.nextLine();



                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement = connection.prepareStatement("EXEC
retrieve_q12 @salary = ?;")) {
```

```
                        // Populate the query template with the data collected from the user
                        statement.setFloat(1, salary_q12);


                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final ResultSet resultSet = statement.executeQuery();

                        System.out.println("Option 12 query results:");
                        System.out.println("employee name ");

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

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

                break;
            case "13": //Retrieve total number of lost work days due to accidents in repairing the
products which got complaints
                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("EXEC
retrieve_q13;")) {



                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final ResultSet resultSet = statement.executeQuery();

                        System.out.println("Option 13 query results:");
                        System.out.println("total lost days ");

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

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

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

            // Collect the new faculty data from the user
            System.out.println("Please enter year of interest:");
                final String year = sc.nextLine(); // Read in the year of interest



                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("EXEC
retrieve_q14 @year = ?;")) {
                        // Populate the query template with the data collected from the user
                        statement.setString(1, year);
```

```
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final ResultSet resultSet = statement.executeQuery();

                        System.out.println("Option 14 query results:");
                        System.out.println("average cost ");

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

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

                break;
            case "15": //Delete all accidents whose dates are in some range

            // Collect the new faculty data from the user
            System.out.println("Please enter start date in mm/dd/yyyy:");
                final String startdate = sc.nextLine(); // Read in the start date of period being
considered


                // Collect the new faculty data from the user
            System.out.println("Please enter end date in mm/dd/yyyy:");
                final String enddate = sc.nextLine(); // Read in the end date of period being
considered


                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("EXEC
retrieve_q15 @startdate = ?, @enddate = ?;")) {
                        // Populate the query template with the data collected from the user
                        statement.setString(1, startdate);
                        statement.setString(2, enddate);


                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_affected = statement.executeUpdate();
                        final SQLWarning warning = statement.getWarnings();
                        if (warning != null)
                                    System.out.println(warning.getMessage());
                        else
                             System.out.println(String.format("Done. %d rows affected.",
rows_affected));

                    }
                }
                catch(SQLException sqe) {
                     System.out.println("Error Message = " + sqe.getMessage());
                }

                break;
            case "16": //Import: enter new employees from file
            System.out.println("Please enter file name:");
                final String filename = sc.nextLine(); //Read in file name from user

                // try to read in file and get the specific user input for insertion using query 1
            try {
```

```
                File myFile = new File(filename+".txt");
                Scanner myReader = new Scanner(myFile);
                System.out.println("Reading File...");
                while (myReader.hasNextLine()) {
                        String data = myReader.nextLine();
                        String[] splitdata = data.split(", ");
                        final String f_emptype = splitdata[0];
                        final String f_name = splitdata[1];
                        final String f_address = splitdata[2];
                        final String f_salary =  splitdata[3];
                        final String f_addl_info = splitdata[4];
                        int spd_size = splitdata.length;
                        final String f_addl_info2;
                        if (spd_size == 6) {
                                f_addl_info2 = splitdata[5];
                        }
                        else {
                                f_addl_info2 = " ";
                        }
                        // after reading in all the required info, run query 1
                        try (final Connection connection = DriverManager.getConnection(URL)) {
                            try (
                                final PreparedStatement statement = connection.prepareStatement("EXEC
insert_new_employee @name = ?, @address = ?, @salary = ?, "
                                                                                                    +
"@emptype = ?, @addl_info = ?, @addl_info2 = ?;")) {
                                // Populate the query template with the data collected from the user
                                statement.setString(1, f_name);
                                statement.setString(2, f_address);
                                statement.setString(3, f_salary);
                                statement.setString(4, f_emptype);
                                statement.setString(5, f_addl_info);
                                statement.setString(6, f_addl_info2);


                                System.out.println("Dispatching the query...");
                                // Actually execute the populated query
                                final int rows_inserted = statement.executeUpdate();
                                final SQLWarning warning = statement.getWarnings();
                                if (warning != null)
                                        System.out.println(warning.getMessage());
                                else
                                 System.out.println(String.format("Done. %d rows inserted.",
rows_inserted));
                                 }
                                 }
                        catch(SQLException sqe) {
                        System.out.println("Error Message = " + sqe.getMessage());
                       }

                        }
                myReader.close();
                } catch (FileNotFoundException e) {
                        System.out.println("ERROR: File Not Found, Please Enter right file name.");
                }


                    break;
                case "17": //Export: retrieve all customers that purchased particular color products into
a file

                // Collect the data from the user
                System.out.println("Please enter color of products:");
                    final String f_color = sc.nextLine(); // Read in the color of products

                    System.out.println("Please enter output file name:");
                    final String outfile = sc.nextLine(); // Read in file name to store output file
```

```
                    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("EXEC
retrieve_q11 @color = ?;")) {
                            // Populate the query template with the data collected from the user
                            statement.setString(1, f_color);


                            System.out.println("Dispatching the query...");
                            // Actually execute the populated query
                            final ResultSet resultSet = statement.executeQuery();
                            // write the ouput of teh query into a new text file and save it
                            try {
                                FileWriter myWriter = new FileWriter(outfile+".txt");
                                while (resultSet.next()) {
                                        myWriter.write(resultSet.getString(1));
                                        myWriter.write("\n");

                                }
                                myWriter.close();
                                } catch (IOException e) {
                                        System.out.println("An Error Occured, Could not write to File");
                                }
                            }

                        }
                    catch(SQLException sqe) {
                         System.out.println("Error Message = " + sqe.getMessage());
                    }


                    break;
                case "18": // Do nothing, the while loop will terminate upon the next iteration
                    System.out.println("Exiting! Goodbye!");
                    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
    }
}
```

Screen shot showing successful compilation below:

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

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

Task 6

Java Program Execution

6.1 Screenshots showing testing of query 1

```
18) Quit
1
Please enter employee type: enter 'technical_staff', 'quality_controller' or 'worker'
technical_staff
Please enter employee name:
Macy
Please enter employee address:
TX
Please enter employee salary:
45000
Please enter additional information for employee:
For technical staff, enter the technical position
For quality controller, enter product type to be checked: ('product1', 'product2', 'product3')
For worker, enter the maximum number of products worker can produce per day
Senior
Please enter employee degree if the employee is a technical staff otherwise press Enter Key:
Enter 'BSC', 'MSC' or 'PHD' for technical staff
MSC
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Insertion into employee main table screen shot

	name	address	salary
1	Ada	OK	25000
2	Adam	OK	30000
3	Carey	KS	28000
4	James	NY	33000
5	Jerry	KS	45000
6	Jessie	NY	35000
7	John	NY	33000
8	Macy	TX	45000
9	Mazekin	OK	45000
10	Nancy	OK	45000

Insertion into sub tables screenshots

technical_staff and technical_staff_degree

	name	technical_position
1	Jerry	Manager
2	Macy	Senior
3	Mazekin	Senior
4	Nancy	Junior

	name	degree
1	Jerry	PHD
2	Macy	MSC
3	Mazekin	MSC
4	Nancy	BSC

quality_controller

	name	product_type
1	Carey	product3
2	James	product2
3	John	product1

worker

	name	max_products_per_day
1	Ada	10
2	Adam	15
3	Jessie	18

6.2 Screenshots showing testing of query 2

```
18) Quit
2
Please enter product type: enter 'Product1', 'Product2' or 'Product3'
Product1
Please enter the product id:
001
Please enter date the product was produced in mm/dd/yyyy:
10/26/2022
Please enter time spent to make the product in hours:
7
Please enter name of worker who produced the product:
Ada
Please enter name of quality controller who tested the product:
John
Please enter name of technical staff who repaired the product:
If product was not repaired, press Enter Key to skip
Macy
Please enter the size of the product, ('small', 'medium' or 'large'):
small
Please enter additional information for product:
For Product 1, enter the major software used
For Product 2, enter the color
For Product 3, enter the weight
Apple
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Screenshot of products main table

	pID	production_date	creation_time	produced_by	tested_by	repaired_by
1	001	2022-10-26	7	Ada	John	Macy
2	002	2022-10-26	10	Adam	John	*NULL*
3	003	2022-10-26	15	Adam	John	Jerry
4	004	2023-10-26	7	Ada	John	*NULL*
5	005	2022-10-26	5	Adam	James	Nancy
6	006	2022-10-26	7	Jessie	James	Mazekin
7	007	2023-10-26	10	Adam	James	*NULL*
8	008	2022-10-26	3	Jessie	Carey	*NULL*
9	009	2022-10-26	5	Jessie	Carey	Macy
10	010	2023-10-26	7	Adam	Carey	*NULL*

Screenshots of sub tables

product1

	pID	size	major_software
1	001	small	Apple
2	002	medium	Google
3	003	large	Google
4	004	small	Google

product2

	pID	size	color
1	005	small	green
2	006	medium	yellow
3	007	large	green

product3

	pID	size	weight
1	008	small	20.5
2	009	medium	30.5
3	010	large	40.5

Screenshot of repair table that has been updated due to repairs ordered by quality controller

	pID	tech_staff_name	cID	qc_name	date
1	001	Macy	NULL	John	2022-10-27
2	003	Jerry	NULL	John	2022-10-27
3	005	Nancy	NULL	James	2022-10-27
4	006	Mazekin	NULL	James	2022-10-27
5	009	Macy	NULL	Carey	2022-10-27

6.3 Screenshots showing testing of query 3

```
18) Quit
3
Please enter the customer name:'
David Nnamdi
Please enter the customer address:
TX
Please enter all products purchased by customer, separate each entry with a space:
When done with entering all products, press Enter and 'end':
001 004 009
end
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

Customer table after all 10 insertions

	cname	pID	address
1	Chinelo Nnamdi	006	TX
2	David Nnamdi	001	TX
3	David Nnamdi	004	TX
4	David Nnamdi	009	TX
5	Kanayo Nnamdi	003	TX
6	Kanayo Nnamdi	010	TX
7	Melissa Nnamdi	002	TX
8	Melissa Nnamdi	005	TX
9	Nnamdi Nnamdi	007	TX
10	Samuel Nnamdi	008	TX

6.4 Screenshots showing testing of query 4

```
18) Quit
4
Please enter account number:
001
Please enter the date the account was established in mm/dd/yyyy:
10/26/2022
Please enter the id of the product we want to establish an account for:
001
Please enter cost of product:
50.5
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Screenshot of the main account table

	account_no	establishment_date	cost	pID
1	001	2022-10-26	50.5	001
2	002	2022-10-26	66.5	002
3	003	2022-10-26	70.5	003
4	004	2023-10-26	50.5	004
5	005	2022-10-26	66.5	005
6	006	2022-10-26	70.5	006
7	007	2023-10-26	50.5	007
8	008	2022-10-26	66.5	008
9	009	2022-10-26	70.5	009
10	010	2023-10-26	70.5	010

Screenshot of the sub tables

product1_account

	account_no	pID
1	001	001
2	002	002
3	003	003
4	004	004

product2_account

	account_no	pID
1	005	005
2	006	006
3	007	007

product3_account

	account_no	pID
1	008	008
2	009	009
3	010	010

6.5 Screenshots showing testing of query 5

```
18) Quit
5
Please enter product complaint id:
111
Please enter date of complaint in mm/dd/yyyy:
11/01/2022
Please enter name of customer who has filed a complaint:
David Nnamdi
Please enter the product id:
004
Please enter description of what is wrong with product:
not working
Please enter treament expected for complaint:
Enter 'refund' if customer wants to return product and be refunded
Enter 'replace' if customer wants the product to be replaced with a working one
replace
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Screenshot of complaints table after 3 entries

	cID	date	description	treatment_expected	cname	pID
1	111	2022-11-01	not working	replace	David Nnamdi	004
2	112	2022-11-02	broken	replace	Melissa Nnamdi	002
3	113	2022-11-03	not working	refund	Chinelo Nnamdi	006

Screenshot of updated repair table after entering complaint

	pID	tech_staff_name	cID	qc_name	date
1	001	Macy	NULL	John	2022-10-27
2	002	Macy	112	NULL	2022-11-06
3	003	Jerry	NULL	John	2022-10-27
4	004	Macy	111	NULL	2022-11-05
5	005	Nancy	NULL	James	2022-10-27
6	006	Mazekin	NULL	James	2022-10-27
7	006	Jerry	113	NULL	2022-11-07
8	009	Macy	NULL	Carey	2022-10-27

6.6 Screenshots showing testing of query 6

```
18) Quit
6
Please enter the accident type:
Enter 'repair' if accident occured during product repair
Enter 'production' if accident occured during production of product
repair
Please enter accident number:
1
Please enter date of accident in mm/dd/yyyy:
10/26/2022
Please enter number of days lost due to accident:
1
Please enter name of employee involved in accident:
Macy
Please enter id of product being worked on during the accident:
001
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Screenshot of accident table after 3 entries

	number	date	lost_days
1	1	2022-10-26	1
2	2	2022-10-26	2
3	3	2022-10-26	1

Screenshot of accident sub tables

repair_accident

	number	name	pID
1	1	Macy	001
2	3	Mazekin	004

production_accident

	number	name	pID
1	2	Adam	007

6.7 Screenshots showing testing of query 7

```
18) Quit
7
Please enter product ID:
001
Connecting to the database...
Dispatching the query...
Option 7 query results:
production date | creation time
2022-10-26 | 7
```

```
18) Quit
7
Please enter product ID:
002
Connecting to the database...
Dispatching the query...
Option 7 query results:
production date | creation time
2022-10-26 | 10
```

```
18) Quit
7
Please enter product ID:
003
Connecting to the database...
Dispatching the query...
Option 7 query results:
production date | creation time
2022-10-26 | 15
```

6.8 Screenshots showing testing of query 8

```
18) Quit
8
Please enter name of worker:
Ada
Connecting to the database...
Dispatching the query...
Option 8 query results:
product id
001
004
```

```
18) Quit
8
Please enter name of worker:
Adam
Connecting to the database...
Dispatching the query...
Option 8 query results:
product id
002
003
005
007
010
```

```
18) Quit
8
Please enter name of worker:
Jessie
Connecting to the database...
Dispatching the query...
Option 8 query results:
product id
006
008
009
```

6.9 Screenshots showing testing of query 9

```
18) Quit
9
Please enter name of quality controller:
John
Connecting to the database...
Dispatching the query...
Option 9 query results:
total errors
2
```

```
18) Quit
9
Please enter name of quality controller:
James
Connecting to the database...
Dispatching the query...
Option 9 query results:
total errors
0
```

```
18) Quit
9
Please enter name of quality controller:
Carey
Connecting to the database...
Dispatching the query...
Option 9 query results:
total errors
0
```

6.10 Screenshots showing testing of query 10

```
18) Quit
10
Please enter name of quality controller:
Carey
Connecting to the database...
Dispatching the query...
Option 10 query results:
total cost
70.5
```

```
18) Quit
10
Please enter name of quality controller:
James
Connecting to the database...
Dispatching the query...
Option 10 query results:
total cost
null
```

```
18) Quit
10
Please enter name of quality controller:
John
Connecting to the database...
Dispatching the query...
Option 10 query results:
total cost
null
```

6.11 Screenshots showing testing of query 11

```
11
Please enter color of products:
green
Connecting to the database...
Dispatching the query...
Option 11 query results:
customer name
Melissa Nnamdi
Nnamdi Nnamdi
```

```
18) Quit
11
Please enter color of products:
yellow
Connecting to the database...
Dispatching the query...
Option 11 query results:
customer name
Chinelo Nnamdi
```

```
18) Quit
11
Please enter color of products:
blue
Connecting to the database...
Dispatching the query...
Option 11 query results:
customer name
```

6.12 Screenshots showing testing of query 12

```
18) Quit
12
Please enter target salary:
30000
Connecting to the database...
Dispatching the query...
Option 12 query results:
employee name
James
Jerry
Jessie
John
Macy
Mazekin
Nancy
```

6.13 Screenshots showing testing of query 13

```
18) Quit
13
Connecting to the database...
Dispatching the query...
Option 13 query results:
total lost days
1.0
```

6.14 Screenshots showing testing of query 14

```
18) Quit
14
Please enter year of interest:
2023
Connecting to the database...
Dispatching the query...
Option 14 query results:
average cost
57.167
```

6.15 Screenshots showing testing of query 15

```
18) Quit
15
Please enter start date in mm/dd/yyyy:
9/1/2022
Please enter end date in mm/dd/yyyy:
11/1/2022
Connecting to the database...
Dispatching the query...
Done. 2 rows affected.
```

Screenshot of affected accident table

	number	date	lost_days

Note: repair_accident and production_accident table also affected and are empty

6.16 Screenshots showing testing of the import and export options

Import:

A text file named input will be used to insert new employees. The contents of the file are shown below

input - Notepad

File Edit Format View Help

```
technical_staff, Jeremiah, OK, 53000, Senior, PHD
technical_staff, Harrieth, NG, 75000, Manager, PHD
technical_staff, Breanna, OK, 50000, Junior, BSC
quality_controller, Ayomide, TX, 25000, product1
quality_controller, Jenson, KS, 27000, product2
quality_controller, Yemi, NG, 15000, product3
worker, Jimmy, TX, 35000, 15
worker, Paper, OK, 25000, 10
worker, Raider, CA, 40000, 20
```

Running it on java:

```
18) Quit
16
Please enter file name:
input
Reading File...
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
Dispatching the query...
Done. 1 rows inserted.
```

The updated employees main table after insertion

	name	address	salary
1	Ada	OK	25000
2	Adam	OK	30000
3	Ayomide	TX	25000
4	Breanna	OK	50000
5	Carey	KS	28000
6	Harrieth	NG	75000
7	James	NY	33000
8	Jenson	KS	27000
9	Jeremiah	OK	53000
10	Jerry	KS	45000
11	Jessie	NY	35000
12	Jimmy	TX	35000
13	John	NY	33000
14	Macy	TX	45000
15	Mazekin	OK	45000
16	Nancy	OK	45000
17	Paper	OK	25000
18	Raider	CA	40000
19	Yemi	NG	15000

Export:

The results of the query will be exported to a file called results. In java it is run as follows:

```
18) Quit
17
Please enter color of products:
green
Please enter output file name:
results
Connecting to the database...
Dispatching the query...
```

The content of the file is shown below:

```
results - Notepad
File  Edit  Format  View  Help
Melissa Nnamdi
Nnamdi Nnamdi
```

6.17 Screenshots showing testing of the 3 types of errors

3 erroneous queries will be run to demonstrate error handling capacity.

1. Inserting an employee that already exists into the table – Azure SQL program should return a duplicate key error

```
18) Quit
1
Please enter employee type: enter 'technical_staff', 'quality_controller' or 'worker'
technical_staff
Please enter employee name:
Macy
Please enter employee address:
TX
Please enter employee salary:
45000
Please enter additional information for employee:
For technical staff, enter the technical position
For quality controller, enter product type to be checked: ('product1', 'product2', 'product3')
For worker, enter the maximum number of products worker can produce per day
Senior
Please enter employee degree if the employee is a technical staff otherwise press Enter Key:
Enter 'BSC', 'MSC' or 'PHD' for technical staff
MSC
Connecting to the database...
Dispatching the query...
Error Message = Violation of PRIMARY KEY constraint 'PK__employee__72E12F1A1D0875A3'. Cannot insert duplicate key in object 'dbo.employee'. The duplicate key value is (Macy).
```

2. Inserting a customer with purchase of a product that does not exist in the products table or inventory – Azure SQL program should return a foreign key constraint since in this case we try to add customer for product ID '015' but we all we have in inventory is '001-010'

```
18) Quit
3
Please enter the customer name:'
Jamil Sunil
Please enter the customer address:
OK
Please enter all products purchased by customer, separate each entry with a space:
When done with entering all products, press Enter and 'end':
015
end
Connecting to the database...
Dispatching the query...
Error Message = The INSERT statement conflicted with the FOREIGN KEY constraint "FK_cPID". The conflict occurred in database "cs-dsa-4513-sql-db", table "dbo.products", column 'pID'.
```

3. Inserting a product with a worker name (Amina) that does not exist in the worker table, here the foreign key constraint error pops up

```
18) Quit
2
Please enter product type: enter 'Product1', 'Product2' or 'Product3'
Product1
Please enter the product id:
014
Please enter date the product was produced in mm/dd/yyyy:
10/26/2023
Please enter time spent to make the product in hours:
7
Please enter name of worker who produced the product:
Amina
Please enter name of quality controller who tested the product:
John
Please enter name of technical staff who repaired the product:
If product was not repaired, press Enter Key to skip
Macy
Please enter the size of the product, ('small', 'medium' or 'large'):
small
Please enter additional information for product:
For Product 1, enter the major software used
For Product 2, enter the color
For Product 3, enter the weight
Apple
Connecting to the database...
Dispatching the query...
Error Message = The INSERT statement conflicted with the FOREIGN KEY constraint "FK_produced_by". The conflict occurred in database "cs-dsa-4513-sql-db", table "dbo.worker", column 'name'.
```

6.18 Screenshots showing testing of the Quit option

```
18) Quit
18
Exiting! Goodbye!
```

Task 7

Web database application and its execution

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

DataHandler.java

```
package DSA4513_project;

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

public class DataHandler {

    private Connection conn;

    // Azure SQL connection credentials
    private String server = "nnam0000-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "nnam0000";
    private String password = "Onyinye26$$";

    // Resulting connection string
    final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificat
e=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
                    server, database, username, password);

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

```
        }

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



    // Inserts an employee using stored procedure for q1 with the given attribute values
    public boolean addEmployee(
            String emptype, String name, String address, float salary, String addl_info, String
addl_info2) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
                "EXEC insert_new_employee @name = ?, @address = ?, @salary = ?, " +
                                "@emptype = ?, @addl_info = ?, @addl_info2 = ?;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, name);
        stmt.setString(2, address);
        stmt.setFloat(3, salary);
        stmt.setString(4, emptype);
        stmt.setString(5, addl_info);
        stmt.setString(6, addl_info2);


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

    // Return all employees with salary great than the given attribute value
    public ResultSet getEmpGreaterSalary(float salary) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery = "EXEC retrieve_q12 @salary = ?;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

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


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

}
```

get_emp_salary_greater.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Get Employees With Salary Above A Particular Salary</title>
    </head>
    <body>
        <h2>Get Employees With Salary Above A Particular Salary</h2>
        <!--
            Form for collecting user input for the new movie_night record.
            Upon form submission, add_movie.jsp file will be invoked.
        -->
```

```
        <form action="get_emp_salary_greater.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Enter the Salary:</th>
                </tr>
                <tr>
                    <td>Salary:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=salary>
                    </div></td>
                </tr>
                <tr>
                    <td><div style="text-align: center;">
                    <input type=reset value=Clear>
                    </div></td>
                    <td><div style="text-align: center;">
                    <input type=submit value=Insert>
                    </div></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

get_emp_salary_greater.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="DSA4513_project.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

    // Get the attribute values passed from the input form.
    String salary = request.getParameter("salary");


    /*
     * If the user hasn't filled out the salary correctly. This is very simple checking.
     */
    if (salary.equals("")) {
        response.sendRedirect("get_emp_salary_greater_form.jsp");
    } else {
        float salary_ = Float.parseFloat(salary);

                // Now perform the query with the data from the form.
            final ResultSet employees = handler.getEmpGreaterSalary(salary_);
            %>
            <!-- The table for displaying all the employee records -->
                <table cellspacing="2" cellpadding="2" border="1">
                    <tr> <!-- The table headers row -->
                      <td align="center">
                        <h4>Employee name</h4>
```

```
                    </td>
                  </tr>
                  <%
                     while(employees.next()) { // For each movie_night record returned...
                         // Extract the attribute values for every row returned
                         final String name = employees.getString("name");


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

                     }
    %>
          </table>
          <a href="add_employee_form.jsp">Add new employee.</a>
    </body>
</html>
```

add_employee_form.jsp

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

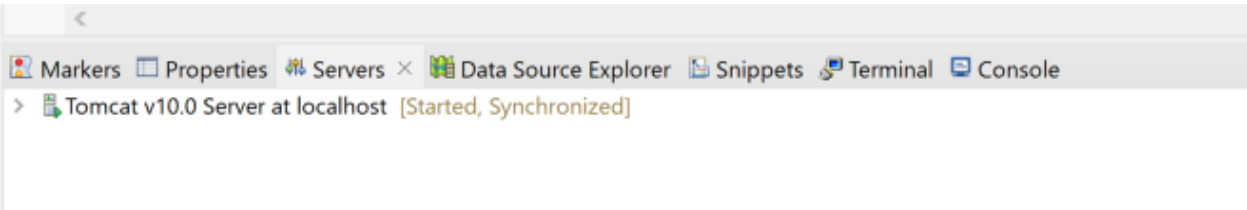
```
                <td><div style="text-align: center;">
                <input type=text name=addl_info>
                </div></td>
            </tr>
            <tr>
                <td>Degree: </td>
                <td><div style="text-align: center;">
                <input type=text name=addl_info2>
                </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>
    <h3>Note: </h3>
    <a>For Employee Type: Please enter "technical_staff", "quality_controller" or "worker"</a>
    <br/>
    <a>For Technical Staff: The additional information is the technical position</a>
    <br/>
    <a>For Quality Controller: The additional information is product type, please enter ("product1",
"product2" or "product3")</a>
    <br/>
    <a>For Worker: The additional information is the max number of products they can make each day</a>
    <br/>
    <a>For Degree: Please enter "BSC", "MSC" or "PHD". Note that this entry is only required for
technical staff</a>
    </body>
</html>
```

add_employee.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="DSA4513_project.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

    // Get the attribute values passed from the input form.
    String name = request.getParameter("name");
    String address = request.getParameter("address");
    String salaryString = request.getParameter("salary");
    String emptype = request.getParameter("emptype");
    String addl_info = request.getParameter("addl_info");
    String addl_info2 = request.getParameter("addl_info2");

    /*
     * If the user hasn't filled out all the name, address, salary, employee type and addition info
correctly.
     This is very simple checking.
     */
```

```
    if (name.equals("") || address.equals("") || salaryString.equals("") || emptype.equals("") ||
addl_info.equals(""))
    {
        response.sendRedirect("add_employee_form.jsp");
    }

    else if (emptype.equals("technical_staff") && addl_info2.equals("")) {
        response.sendRedirect("add_employee_form.jsp");
    }
    else {
        float salary = Float.parseFloat(salaryString);

        // Now perform the query with the data from the form.
        boolean success = handler.addEmployee(emptype, name, address, salary, addl_info, addl_info2);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the employee</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The Employee:</h2>

            <ul>
                <li>Name: <%=name%></li>
                <li>Address: <%=address%></li>
                <li>Salary: <%=salaryString%></li>
                <li>Employee Type: <%=emptype%></li>
                <li>Additional Information: <%=addl_info%></li>
                <li>Degree: <%=addl_info2%></li>
            </ul>

            <h2>Was successfully inserted.</h2>

            <a href="get_emp_salary_greater_form.jsp">See all employees with a salary greater than
particular salary.</a>
            <%
        }
    }
    %>
    </body>
</html>
```

Screenshot showing that file was good and running on tomcat server



7.2 Screenshots Showing the testing of the web database application

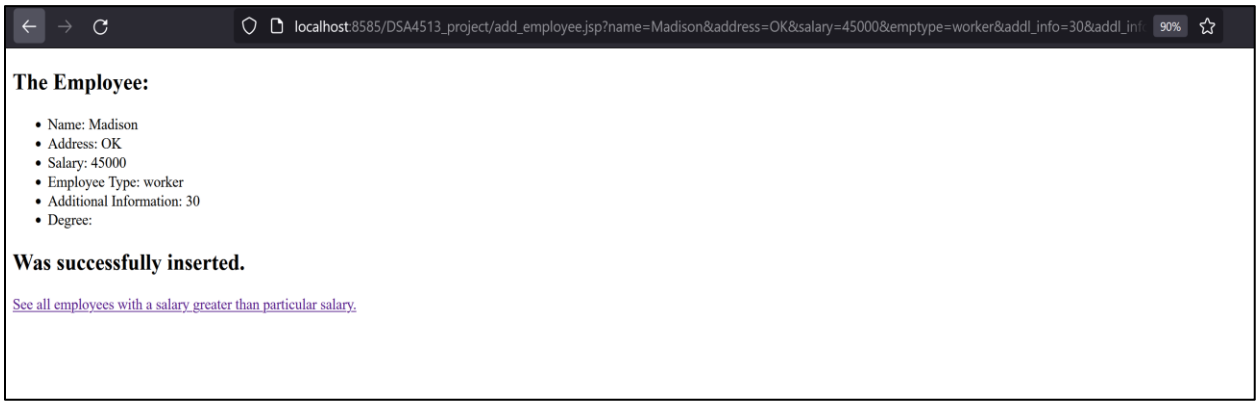
Query 12: salary input of 30000

Query 12 results

Clicking on "*Add new employee*" link on webpage will take us to the employee data input form

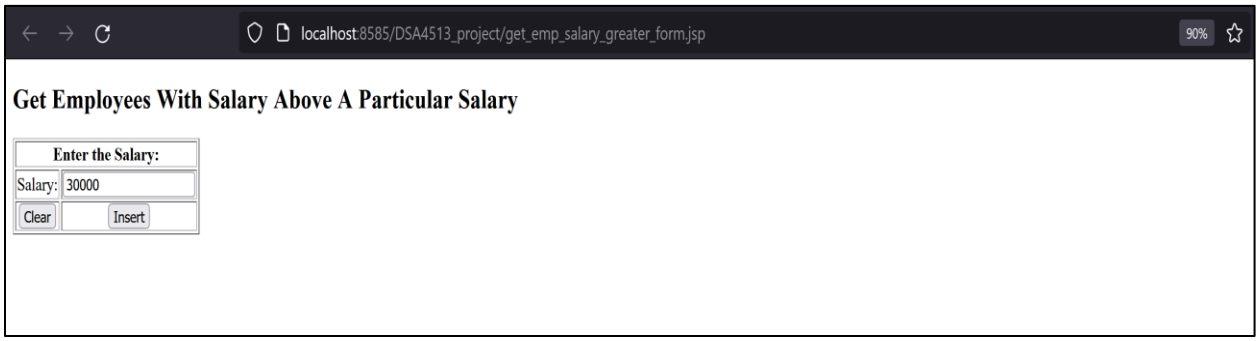
I added a new worker called Madison who earns 45000

Clicking on Insert will tell if the new worker was successfully inserted using query 1 stored procedure



In our case it was, so we can rerun query 12 by clicking on "*See all employees with a salary greater than a particular salary*" link

It is run for 30000 again



The results this time has been updated with Madison now included