NAME: SONAXY MOHANTY
ASSIGNMENT: DBMS PROJECT

Table of Contents

Task 1. *2*

1.1 ER Diagram **2**

1.2 Relational Database Schema **2**

Task 2. Schema Diagram *3*

Task 3. *4*

3.1 Discussion of storage structures for tables **4**

3.2 Discussion of storage structures for tables (Azure SQL Database) **7**

Task 4. SQL Statements and screenshots showing the creation of tables in Azure SQL Database *7*

Task 5. *17*

5.1 SQL statements (and Transact SQL stored procedures, if any) **17**

5.2 The Java source program and screenshots showing its successful compilation **23**

Task 6. Java program Execution *57*

6.1. Screenshots showing the testing of query 1 **57**

6.2 Screenshots showing the testing of query 2 **59**

6.3 Screenshots showing the testing of query 3 **60**

6.4 Screenshots showing the testing of query 4 **61**

6.5 Screenshots showing the testing of query 5 **61**

6.6 Screenshots showing the testing of query 6 **62**

6.7 Screenshots showing the testing of query 7 **63**

6.8 Screenshots showing the testing of query 8 **63**

6.9 Screenshots showing the testing of query 9 **64**

6.10 Screenshots showing the testing of query 10 **64**

6.11 Screenshots showing the testing of query 11 **64**

6.12 Screenshots showing the testing of query 12 **65**

6.13 Screenshots showing the testing of query 13 **65**

6.14 Screenshots showing the testing of query 14 **65**

6.15 Screenshots showing the testing of query 15 **65**

6.16 Screenshots showing the testing of query 16 **65**

6.17 Screenshots showing the testing of query 17 **65**

6.18 Screenshots showing the testing of query 18 **66**

6.19 Error Checks for Query **66**

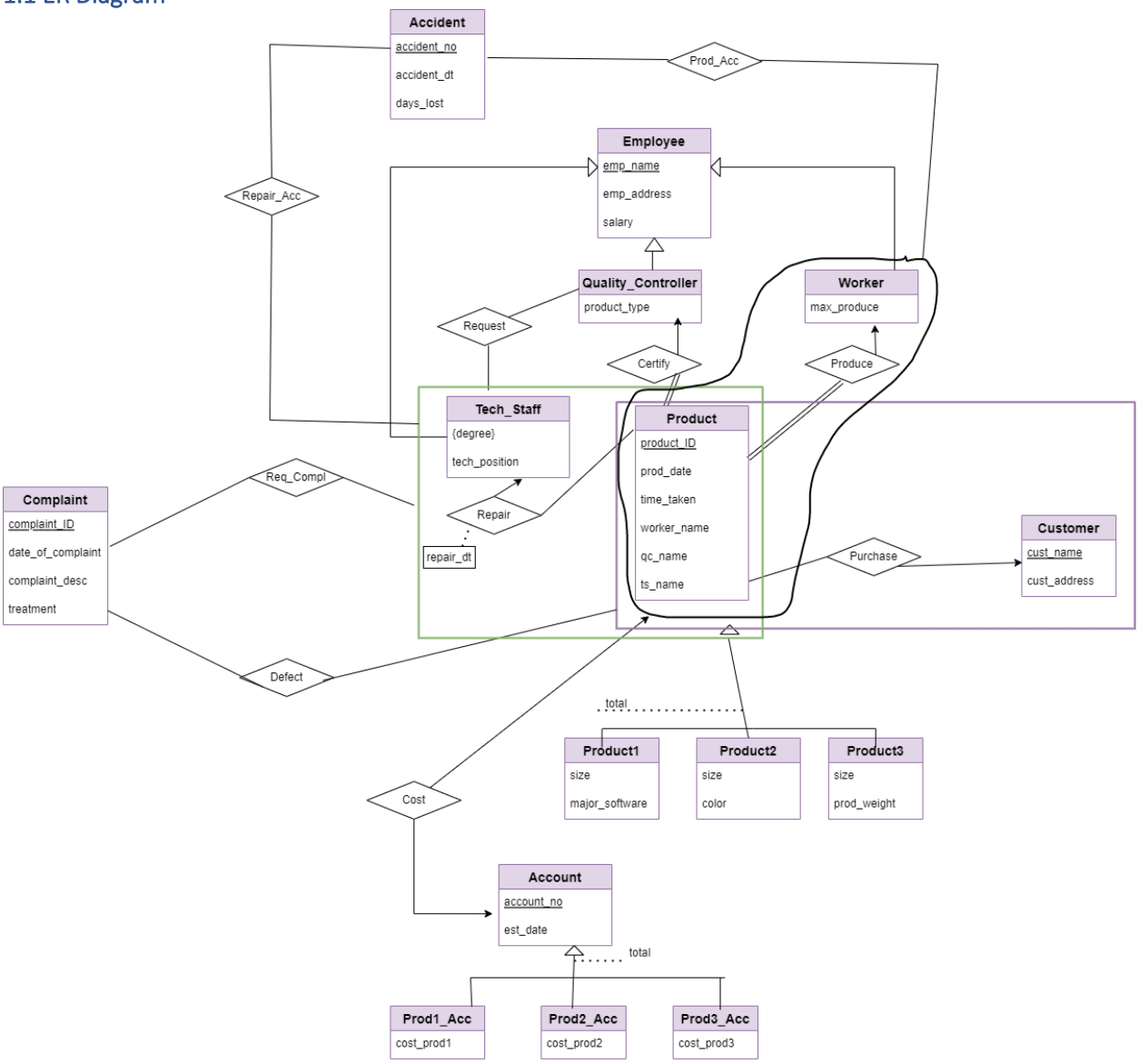
Task 7. Web database application and its execution *67*

7.1 Web database application source program and screenshots showing its successful compilation **67**

7.2 Screenshots showing the testing of the Web database application **73**

Task 1.

1.1 ER Diagram

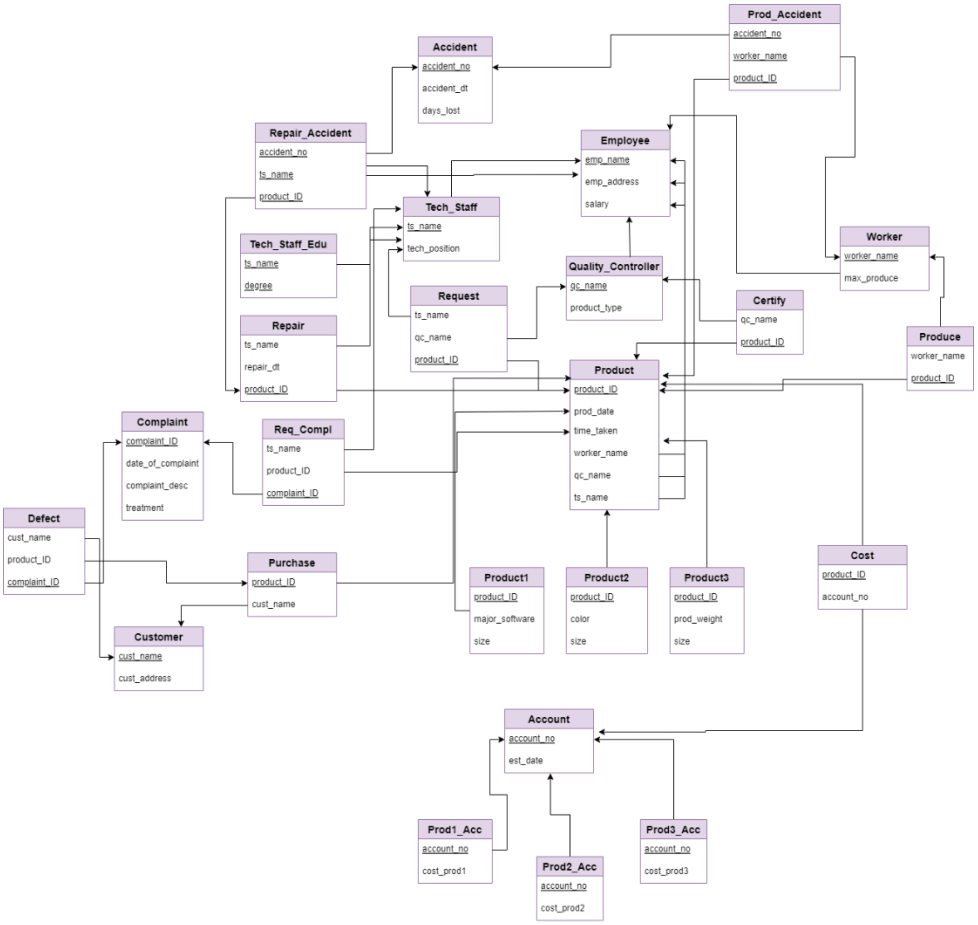


1.2 Relational Database Schema

Employee (<u>emp_name</u>, emp_address, salary)
Tech_Staff (<u>ts_name</u>, tech_position)
Tech_Staff_Edu (<u>ts_name, degree</u>)
Quality_Controller (<u>qc_name</u>, product_type)
Worker (<u>worker_name</u>, max_produce)
Product (<u>product_ID</u>, prod_date, time_taken, worker_name, qc_name, ts_name)
Product1 (<u>product_ID</u>, size, major_software)
Product2 (<u>product_ID</u>, size, color)
Product3 (<u>product_ID</u>, size, prod_weight)

Account (account_no, est_date)
Prod1_Acc (account_no, cost_prod1)
Prod2_Acc (account_no, cost_prod2)
Prod3_Acc (account_no, cost_prod3)
Customer (cust_name, cust_address)
Complaint (complaint_ID, date_of_complaint, complaint_desc, treatment)
Accident (accident_no, accident_dt, days_lost)
Repair (ts_name, product_ID, repair_dt)
Certify (qc_name, product_ID)
Produce (worker_name, product_ID)
Request (qc_name, ts_name, product_ID)
Rep_Compl (ts_name, product_ID, complaint_ID)
Purchase (product_ID, cust_name)
Defect (complaint_ID, product_ID, cust_name)
Cost (product_ID, account_no)
Repair_Accident (accident_no, ts_name, product_ID)
Prod_Accident (accident_no, worker_name, product_ID)

Task 2. Schema Diagram



Task 3.

3.1 Discussion of storage structures for tables

Table Name	Query # and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Employee	1) Insert - INSERT 12) Range Search - SELECT	salary	2/month 1/month	Sequential File Organization using B+ Tree Index with search key as salary	Range search works better with B+ Tree index-sequential file which is why this file organization is selected for this table
Tech_Staff	1) Insert - INSERT		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Tech_Staff_Edu	1) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Quality_Controller	1) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Worker	1) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Product	2) Insert 7) Random Search 8) Random Search 14) Random Search	product_ID product_ID prod_date	400/day 100/day 2000/day 5/day	Dynamic Hashing with hash key as product_ID, prod_date	Random search is efficient with dynamic hashing which is why extendable hash file organization is selected for this table
Product1	2) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Product2	2) Insert 11) Random Search	color	Unknown 5/month	Dynamic hashing with hash key as color	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Product3	2) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Repair	2) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time

Produce	2) Insert 8) Random Search	worker_name	400/day 2000/day	Dynamic hashing with hash key as worker_name	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Certify	2) Insert 9) Random Search	qc_name	400/day 400/day	Dynamic hashing with hash key as qc_name	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Customer	3) Insert 11) Random Search	cust_name	50/day 5/month	B+ tree with search key value as cust_name	Since we need to find the customers in a sorted way along with a random search, it is better to have a B+ tree index
Purchase	3) Insert 11) Random Search	product_ID	50/day 5/month	Dynamic hashing with hash key as product_ID	Random search is efficient with dynamic hashing, hence this table requires extendable hash file organization
Account	4) Insert		40/day	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Prod1_Acc	4) Insert 14) Random Search	account_no	Unknown 5/day	Dynamic hashing with hash key as account_no	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Prod2_Acc	4) Insert 14) Random Search	account_no	Unknown 5/day	Dynamic hashing with hash key as account_no	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Prod3_Acc	4) Insert 10) Random Search 14) Random Search	account_no account_no	Unknown 40/day 5/day	Dynamic hashing with hash key as account_no	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file

					organization is selected for this table
Cost	4) Insert 10) Random Search 14) Random Search	product_ID product_ID	40/day 40/day 5/day	Dynamic hashing with hash key as product_ID	Random search is efficient with dynamic hashing
Complaint	5) Insert		30/day	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Defect	5) Insert 9) Random Search	product_ID	30/day 400/day	Dynamic hashing with hash key as product_ID	Random search is efficient with dynamic hashing
Rep_Compl	5) Insert 13) Random Search	complaint_ID	Unknown 1/month	Dynamic hashing with hash key as complaint_ID	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Accident	6) Insert 13) Random Search 15) Range Search 15) Delete	accident_no accident_dt accident_no	1/week 1/month	B+ tree with search key value as accident_no, accident_dt	>B+ tree index is efficient both for random search and range search >Even insertion and deletion are also faster
Repair_Accident	6) Insert 13) Random Search	product_ID	Unknown 1/month	Dynamic hashing with hash key as product_ID	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file organization is selected for this table
Prod_Accident	6) Insert		Unknown	Heap File Organization	For only insertion, heap file is the best structure as it has fast insertion time
Request	2) Insert 10) Random Search	qc_name	Unknown 40/day	Dynamic hashing with hash key as qc_name	Since Random search is more frequent for this table as compared to Insertion query, so extendable hash file

					organization is selected for this table

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

- All the tables mentioned above which has the Primary key as its search key value (for B+ Tree Sequential file organization) or hash key (for Extendable hash file organization), primary/clustered index are automatically created on these attributes.
- All the tables for which the search key value or hash key are not primary key, non-clustered indexes are created on those attributes using below syntax –

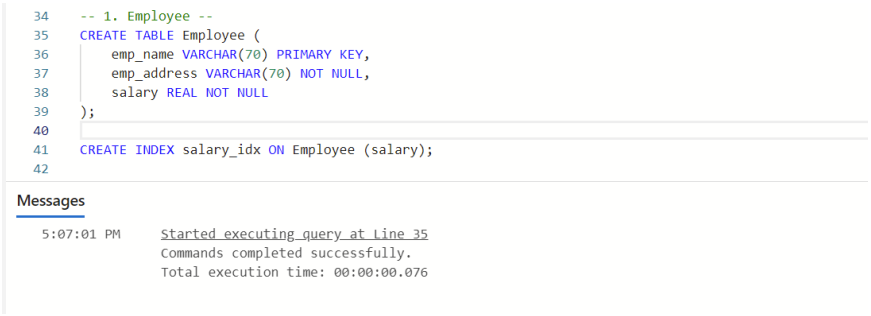
  create index <index_name> on <table> (attribute_name)

- In place of dynamic hashing, B+ Tree Sequential File Organization is being used.

Task 4. SQL Statements and screenshots showing the creation of tables in Azure SQL Database

1. Employee Table

```
CREATE TABLE Employee (
emp_name VARCHAR(70) PRIMARY KEY,
emp_address VARCHAR(70) NOT NULL,
salary REAL NOT NULL
);

CREATE INDEX salary_idx ON Employee (salary);
```



2. Technical Staff Table

```
CREATE TABLE Tech_Staff (
    ts_name VARCHAR(70) PRIMARY KEY,
    tech_position VARCHAR(70) NOT NULL,
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);
```

```
43   -- 2. Technical Staff --
44   CREATE TABLE Tech_Staff (
45       ts_name VARCHAR(70) PRIMARY KEY,
46       tech_position VARCHAR(70) NOT NULL,
47       FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
48   );
49
```

Messages

```
5:11:12 PM     Started executing query at Line 44
               Commands completed successfully.
               Total execution time: 00:00:00.071
```

3. Technical Staff Degree Table

```
CREATE TABLE Tech_Staff_Edu (
    ts_name VARCHAR(70),
    degree VARCHAR(70),
    PRIMARY KEY (ts_name, degree),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);
```

```
50   -- 3. Technical Staff Degree Table --
51   CREATE TABLE Tech_Staff_Edu (
52       ts_name VARCHAR(70),
53       degree VARCHAR(70),
54       PRIMARY KEY (ts_name, degree),
55       FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
56   );
```

Messages

```
5:13:20 PM     Started executing query at Line 50
               Commands completed successfully.
               Total execution time: 00:00:00.073
```

4. Quality Controller Table

```
CREATE TABLE Quality_Controller (
    qc_name VARCHAR(70) PRIMARY KEY,
    product_type VARCHAR(70) NOT NULL,
    FOREIGN KEY (qc_name) REFERENCES Employee (emp_name)
);
```

```
59   CREATE TABLE Quality_Controller (
60       qc_name VARCHAR(70) PRIMARY KEY,
61       product_type VARCHAR(70) NOT NULL,
62       FOREIGN KEY (qc_name) REFERENCES Employee (emp_name)
63   );
64
```

Messages

```
5:16:16 PM     Started executing query at Line 59
               Commands completed successfully.
               Total execution time: 00:00:00.110
```

5. Worker Table

```
CREATE TABLE Worker (
    worker_name VARCHAR(70) PRIMARY KEY,
    max_produce INTEGER NOT NULL,
    FOREIGN KEY (worker_name) REFERENCES Employee (emp_name)
);
```

```
66  CREATE TABLE Worker (
67      worker_name VARCHAR(70) PRIMARY KEY,
68      max_produce INTEGER NOT NULL,
69      FOREIGN KEY (worker_name) REFERENCES Employee (emp_name)
70  );
71
```

Messages

```
5:17:07 PM    Started executing query at Line 66
              Commands completed successfully.
              Total execution time: 00:00:00.074
```

6. Product Table

```
CREATE TABLE Product (
    product_ID INTEGER PRIMARY KEY,
    prod_date DATE NOT NULL,
    time_taken TIME(7) NOT NULL,
    worker_name VARCHAR(70) NOT NULL,
    qc_name VARCHAR(70) NOT NULL,
    ts_name VARCHAR(70),
    FOREIGN KEY (worker_name) REFERENCES Employee (emp_name),
    FOREIGN KEY (qc_name) REFERENCES Employee (emp_name),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);

CREATE INDEX prod_dt_idx ON Product (prod_date);
```

```
73  CREATE TABLE Product (
74      product_ID INTEGER PRIMARY KEY,
75      prod_date DATE NOT NULL,
76      time_taken TIME(7) NOT NULL,
77      worker_name VARCHAR(70) NOT NULL,
78      qc_name VARCHAR(70) NOT NULL,
79      ts_name VARCHAR(70),
80      FOREIGN KEY (worker_name) REFERENCES Employee (emp_name),
81      FOREIGN KEY (qc_name) REFERENCES Employee (emp_name),
82      FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
83  );
84
85  CREATE INDEX prod_dt_idx ON Product (prod_date);
86
```

Messages

```
5:19:06 PM    Started executing query at Line 73
              Commands completed successfully.
              Total execution time: 00:00:00.094
```

7. Product1 category Table

```
CREATE TABLE Product1 (
    product_ID INTEGER PRIMARY KEY,
    size INTEGER NOT NULL,
    major_software VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
);
```

```
87    -- 7. Product1 category Table --
88    CREATE TABLE Product1 (
89        product_ID INTEGER PRIMARY KEY,
90        size INTEGER NOT NULL,
91        major_software VARCHAR(70) NOT NULL,
92        FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
93    );
94
```

Messages

```
5:23:41 PM    Started executing query at Line 88
              Commands completed successfully.
              Total execution time: 00:00:00.083
```

8. Product2 category Table

```
CREATE TABLE Product2 (
    product_ID INTEGER PRIMARY KEY,
    size INTEGER NOT NULL,
    color VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
);

CREATE INDEX prod2_color_idx ON Product2 (color);
```

```
96    CREATE TABLE Product2 (
97        product_ID INTEGER PRIMARY KEY,
98        size INTEGER NOT NULL,
99        color VARCHAR(70) NOT NULL,
100       FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
101   );
102
103   CREATE INDEX prod2_color_idx ON Product2 (color);
104
```

Messages

```
5:25:28 PM    Started executing query at Line 96
              Commands completed successfully.
              Total execution time: 00:00:00.080
```

9. Product3 category Table

```
CREATE TABLE Product3 (
    product_ID INTEGER PRIMARY KEY,
    size INTEGER NOT NULL,
    prod_weight INTEGER NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
);
```

```
105   -- 9. Product3 category Table
106   CREATE TABLE Product3 (
107       product_ID INTEGER PRIMARY KEY,
108       size INTEGER NOT NULL,
109       prod_weight INTEGER NOT NULL,
110       FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
111   );
112
```

Messages

```
5:26:56 PM    Started executing query at Line 106
              Commands completed successfully.
              Total execution time: 00:00:00.114
```

10. Account Table

```
CREATE TABLE Account (
```

```
    account_no INTEGER PRIMARY KEY,
    est_date DATE NOT NULL
);
```

```
114  CREATE TABLE Account (
115      account_no INTEGER PRIMARY KEY,
116      est_date DATE NOT NULL
117  );
118
```

Messages

```
5:28:03 PM    Started executing query at Line 114
              Commands completed successfully.
              Total execution time: 00:00:00.086
```

11. Product1-account category Table

```
CREATE TABLE Prod1_Acc (
    account_no INTEGER PRIMARY KEY,
    cost_prod1 REAL NOT NULL,
    FOREIGN KEY (account_no) REFERENCES Account (account_no)
);
```

```
120  CREATE TABLE Prod1_Acc (
121      account_no INTEGER PRIMARY KEY,
122      cost_prod1 REAL NOT NULL,
123      FOREIGN KEY (account_no) REFERENCES Account (account_no)
124  );
125
```

Messages

```
5:30:14 PM    Started executing query at Line 120
              Commands completed successfully.
              Total execution time: 00:00:00.079
```

12. Product2-account category Table

```
CREATE TABLE Prod2_Acc (
    account_no INTEGER PRIMARY KEY,
    cost_prod2 REAL NOT NULL,
    FOREIGN KEY (account_no) REFERENCES Account (account_no)
);
```

```
126  -- 12. Product2-account category Table
127  CREATE TABLE Prod2_Acc (
128      account_no INTEGER PRIMARY KEY,
129      cost_prod2 REAL NOT NULL,
130      FOREIGN KEY (account_no) REFERENCES Account (account_no)
131  );
132
```

Messages

```
5:31:40 PM    Started executing query at Line 127
              Commands completed successfully.
              Total execution time: 00:00:00.075
```

13. Product3-account category Table

```
CREATE TABLE Prod3_Acc (
    account_no INTEGER PRIMARY KEY,
    cost_prod3 REAL NOT NULL,
    FOREIGN KEY (account_no) REFERENCES Account (account_no)
);
```

```
134   CREATE TABLE Prod3_Acc (
135       account_no INTEGER PRIMARY KEY,
136       cost_prod3 REAL NOT NULL,
137       FOREIGN KEY (account_no) REFERENCES Account (account_no)
138   );
139
```

Messages

```
5:33:14 PM    Started executing query at Line 134
              Commands completed successfully.
              Total execution time: 00:00:00.074
```

14. Customer Table

```
CREATE TABLE Customer (
    cust_name VARCHAR(70) PRIMARY KEY,
    cust_address VARCHAR(70) NOT NULL
);
```

```
141   CREATE TABLE Customer (
142       cust_name VARCHAR(70) PRIMARY KEY,
143       cust_address VARCHAR(70) NOT NULL
144   );
145
```

Messages

```
5:35:25 PM    Started executing query at Line 141
              Commands completed successfully.
              Total execution time: 00:00:00.092
```

15. Complaint Table

```
CREATE TABLE Complaint (
    complaint_ID INTEGER PRIMARY KEY,
    date_of_complaint DATE NOT NULL,
    complaint_desc VARCHAR(70) NOT NULL,
    treatment VARCHAR(70) NOT NULL
);
```

```
146   -- 15. Complaint Table --
147   CREATE TABLE Complaint (
148       complaint_ID INTEGER PRIMARY KEY,
149       date_of_complaint DATE NOT NULL,
150       complaint_desc VARCHAR(70) NOT NULL,
151       treatment VARCHAR(70) NOT NULL
152   );
153
```

Messages

```
5:37:04 PM    Started executing query at Line 147
              Commands completed successfully.
              Total execution time: 00:00:00.097
```

16. Accident Table

```
CREATE TABLE Accident (
    accident_no INTEGER PRIMARY KEY,
    accident_dt DATE NOT NULL,
    days_lost INTEGER NOT NULL
);

CREATE INDEX accident_dt_idx ON Accident (accident_dt);
```

```
155    CREATE TABLE Accident (
156        accident_no INTEGER PRIMARY KEY,
157        accident_dt DATE NOT NULL,
158        days_lost INTEGER NOT NULL
159    );
160
161    CREATE INDEX accident_dt_idx ON Accident (accident_dt);
162
```

Messages

```
5:38:14 PM    Started executing query at Line 155
              Commands completed successfully.
              Total execution time: 00:00:00.075
```

17. Repair Table

```
CREATE TABLE Repair (
    product_ID INTEGER PRIMARY KEY,
    ts_name VARCHAR(70) NOT NULL,
    repair_dt DATE NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);
```

```
164    CREATE TABLE Repair (
165        product_ID INTEGER PRIMARY KEY,
166        ts_name VARCHAR(70) NOT NULL,
167        repair_dt DATE NOT NULL,
168        FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
169        FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
170    );
171
```

Messages

```
5:39:30 PM    Started executing query at Line 164
              Commands completed successfully.
              Total execution time: 00:00:00.067
```

18. Certify Table

```
CREATE TABLE Certify (
    product_ID INTEGER PRIMARY KEY,
    qc_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (qc_name) REFERENCES Employee (emp_name)
);

CREATE INDEX certify_idx ON Certify (qc_name);
```

```
173    CREATE TABLE Certify (
174        product_ID INTEGER PRIMARY KEY,
175        qc_name VARCHAR(70) NOT NULL,
176        FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
177        FOREIGN KEY (qc_name) REFERENCES Employee (emp_name)
178    );
179
180    CREATE INDEX certify_idx ON Certify (qc_name);
181
```

Messages

```
5:40:38 PM    Started executing query at Line 173
              Commands completed successfully.
              Total execution time: 00:00:00.078
```

19. Produce Table

```
CREATE TABLE Produce (
    product_ID INTEGER PRIMARY KEY,
    worker_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (worker_name) REFERENCES Employee (emp_name)
);

CREATE INDEX produce_idx ON Produce (worker_name);
```

```
183    CREATE TABLE Produce (
184        product_ID INTEGER PRIMARY KEY,
185        worker_name VARCHAR(70) NOT NULL,
186        FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
187        FOREIGN KEY (worker_name) REFERENCES Employee (emp_name)
188    );
189
190    CREATE INDEX produce_idx ON Produce (worker_name);
191
```

Messages

```
5:44:06 PM    Started executing query at Line 183
              Commands completed successfully.
              Total execution time: 00:00:00.060
```

20. Request Table

```
CREATE TABLE Request (
    product_ID INTEGER PRIMARY KEY,
    qc_name VARCHAR(70) NOT NULL,
    ts_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (qc_name) REFERENCES Employee (emp_name),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);

CREATE INDEX request_idx ON Request (qc_name);
```

```
193    CREATE TABLE Request (
194        product_ID INTEGER PRIMARY KEY,
195        qc_name VARCHAR(70) NOT NULL,
196        ts_name VARCHAR(70) NOT NULL,
197        FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
198        FOREIGN KEY (qc_name) REFERENCES Employee (emp_name),
199        FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
200    );
201
202    CREATE INDEX request_idx ON Request (qc_name);
203
```

Messages

```
5:45:55 PM    Started executing query at Line 193
              Commands completed successfully.
              Total execution time: 00:00:00.116
```

21. Repair due to complaint Table

```
CREATE TABLE Rep_Compl (
    complaint_ID INTEGER PRIMARY KEY,
    product_ID INTEGER NOT NULL,
    ts_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (complaint_ID) REFERENCES Complaint (complaint_ID),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);
```

```sql
-- 21. Repair due to complaint Table --
CREATE TABLE Rep_Compl (
    complaint_ID INTEGER PRIMARY KEY,
    product_ID INTEGER NOT NULL,
    ts_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (complaint_ID) REFERENCES Complaint (complaint_ID),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name)
);
```

Messages

```
5:47:10 PM    Started executing query at Line 205
              Commands completed successfully.
              Total execution time: 00:00:00.094
```

22. Purchase Table

```sql
CREATE TABLE Purchase (
    product_ID INTEGER PRIMARY KEY,
    cust_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (cust_name) REFERENCES Customer (cust_name)
);
```

```sql
-- 22. Purchase Table --
CREATE TABLE Purchase (
    product_ID INTEGER PRIMARY KEY,
    cust_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (cust_name) REFERENCES Customer (cust_name)
);
```

Messages

```
5:48:51 PM    Started executing query at Line 215
              Commands completed successfully.
              Total execution time: 00:00:00.055
```

23. Defect Table

```sql
CREATE TABLE Defect (
    complaint_ID INTEGER PRIMARY KEY,
    product_ID INTEGER NOT NULL,
    cust_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (complaint_ID) REFERENCES Complaint (complaint_ID),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (cust_name) REFERENCES Customer (cust_name)
);

CREATE INDEX defect_prod_idx ON Defect (product_ID);
```

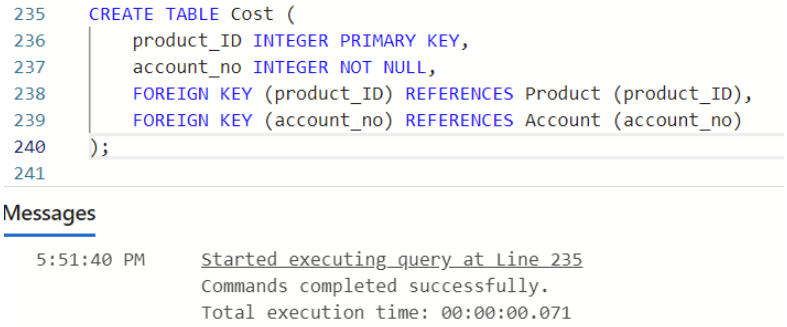
```sql
CREATE TABLE Defect (
    complaint_ID INTEGER PRIMARY KEY,
    product_ID INTEGER NOT NULL,
    cust_name VARCHAR(70) NOT NULL,
    FOREIGN KEY (complaint_ID) REFERENCES Complaint (complaint_ID),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (cust_name) REFERENCES Customer (cust_name)
);

CREATE INDEX defect_prod_idx ON Defect (product_ID);
```

Messages

```
5:49:58 PM    Started executing query at Line 223
              Commands completed successfully.
              Total execution time: 00:00:00.074
```

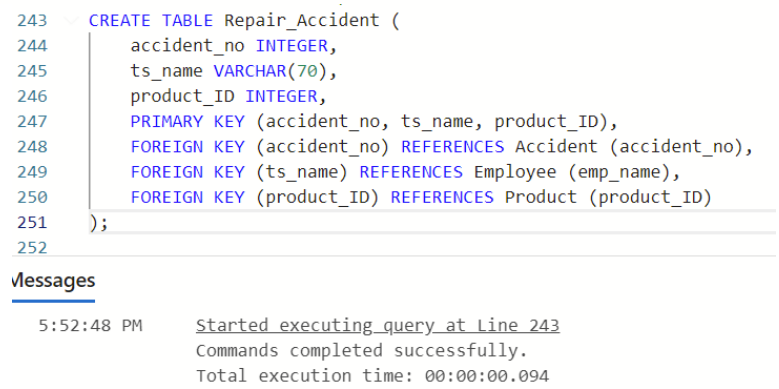
24. Cost Table

```
CREATE TABLE Cost (
    product_ID INTEGER PRIMARY KEY,
    account_no INTEGER NOT NULL,
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID),
    FOREIGN KEY (account_no) REFERENCES Account (account_no)
);
```



25. Accident due to repair Table

```
CREATE TABLE Repair_Accident (
    accident_no INTEGER,
    ts_name VARCHAR(70),
    product_ID INTEGER,
    PRIMARY KEY (accident_no, ts_name, product_ID),
    FOREIGN KEY (accident_no) REFERENCES Accident (accident_no),
    FOREIGN KEY (ts_name) REFERENCES Employee (emp_name),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
);
```



26. Accident due to production Table

```
CREATE TABLE Prod_Accident (
    accident_no INTEGER,
    worker_name VARCHAR(70),
    product_ID INTEGER,
    PRIMARY KEY (accident_no, worker_name, product_ID),
    FOREIGN KEY (accident_no) REFERENCES Accident (accident_no),
    FOREIGN KEY (worker_name) REFERENCES Employee (emp_name),
    FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
);
```

```
254   CREATE TABLE Prod_Accident (
255       accident_no INTEGER,
256       worker_name VARCHAR(70),
257       product_ID INTEGER,
258       PRIMARY KEY (accident_no, worker_name, product_ID),
259       FOREIGN KEY (accident_no) REFERENCES Accident (accident_no),
260       FOREIGN KEY (worker_name) REFERENCES Employee (emp_name),
261       FOREIGN KEY (product_ID) REFERENCES Product (product_ID)
262   );
263
```

Messages

```
5:54:14 PM     Started executing query at Line 254
               Commands completed successfully.
               Total execution time: 00:00:00.071
```

Task 5.

5.1 SQL statements (and Transact SQL stored procedures, if any)

1) Enter a new employee

```
CREATE PROCEDURE New_Employee
@emp_name VARCHAR(70),
@emp_address VARCHAR(70),
@salary REAL
AS
BEGIN
    INSERT INTO Employee VALUES (@emp_name, @emp_address, @salary);
END
GO

-- if the new employee is a technical staff
CREATE PROCEDURE New_Employee_Tech
@emp_name VARCHAR(70),
@tech_position VARCHAR(70),
@BS_ind INTEGER,
@MS_ind INTEGER,
@PhD_ind INTEGER
AS
BEGIN
    INSERT INTO Tech_Staff VALUES (@emp_name, @tech_position);
    IF @BS_ind = 1
        INSERT INTO Tech_Staff_Edu VALUES (@emp_name, 'BS');
    IF @MS_ind = 1
        INSERT INTO Tech_Staff_Edu VALUES (@emp_name, 'MS');
    IF @PhD_ind = 1
        INSERT INTO Tech_Staff_Edu VALUES (@emp_name, 'PhD');

END
GO

-- if the new employee is a quality controller
CREATE PROCEDURE New_Employee_QC
@emp_name VARCHAR(70),
@product_type VARCHAR(70)
AS
BEGIN
    INSERT INTO Quality_Controller VALUES (@emp_name, @product_type);

END
```

```
GO

-- if the new employee is a worker
CREATE PROCEDURE New_Employee_Worker
@emp_name VARCHAR(70),
@max_produce VARCHAR(70)
AS
BEGIN
    INSERT INTO Worker VALUES (@emp_name, @max_produce);

END
GO
```

2) Enter a new product associated with the person who made the product, repaired the product if it is repaired, or checked the product

```
-- procedure for the product with any repair done to the product
CREATE PROCEDURE New_Product_Repair
@product_ID INTEGER,
@prod_date DATE,
@time_taken TIME(7),
@worker_name VARCHAR(64),
@qc_name VARCHAR(64),
@ts_name VARCHAR(64)
AS
BEGIN
   IF @ts_name IS NOT NULL
        INSERT INTO Product (product_ID, prod_date, time_taken, worker_name, qc_name, ts_name)
        VALUES (@product_ID, @prod_date, @time_taken, @worker_name, @qc_name, @ts_name) ;
END
GO

-- insert query for the product with no repair
INSERT INTO Product (product_ID, prod_date, time_taken, worker_name, qc_name) VALUES (?, ?, ?,
?, ?);

-- procedure to associate product with worker
CREATE PROCEDURE New_Produce
@product_ID INTEGER,
@worker_name VARCHAR(64)
AS
BEGIN
    INSERT INTO Produce VALUES (@product_ID, @worker_name);
END
GO

-- procedure to associate product with quality controller
CREATE PROCEDURE New_Certify
@product_ID INTEGER,
@qc_name VARCHAR(64)
AS
BEGIN
    INSERT INTO Certify VALUES (@product_ID, @qc_name);
END
GO

-- insert statement for repair
INSERT INTO Repair VALUES (?, ?, ?);

-- procedure for associating product with quality controller and technical staff
-- in case any request is done by quality controller for the repair
```

```
CREATE PROCEDURE New_Request
@product_ID INTEGER,
@qc_name VARCHAR(64),
@ts_name VARCHAR(64)
AS
BEGIN
    INSERT INTO Request VALUES (@product_ID, @qc_name, @ts_name);
END
GO

-- procedure to insert product type 1 details
CREATE PROCEDURE New_Product1
@product_ID INTEGER,
@size INTEGER,
@major_software VARCHAR(64)
AS
BEGIN
    INSERT INTO Product1 VALUES (@product_ID, @size, @major_software);
END
GO

-- procedure to insert product type 2 details
CREATE PROCEDURE New_Product2
@product_ID INTEGER,
@size INTEGER,
@color VARCHAR(64)
AS
BEGIN
    INSERT INTO Product2 VALUES (@product_ID, @size, @color);
END
GO

-- procedure to insert product type 3 details
CREATE PROCEDURE New_Product3
@product_ID INTEGER,
@size INTEGER,
@prod_weight INTEGER
AS
BEGIN
    INSERT INTO Product3 VALUES (@product_ID, @size, @prod_weight);
END
GO
```

3) Enter a customer associated with some products

```
INSERT INTO Customer VALUES (?, ?);
INSERT INTO Purchase VALUES (?, ?);
```

4) Create a new account associated with a product

```
-- procedure to insert account type 1 details
CREATE PROCEDURE New_Account1
@account_no INTEGER,
@est_date DATE,
@cost REAL,
@product_ID INTEGER
AS
BEGIN
    INSERT INTO Account VALUES (@account_no, @est_date);
    DECLARE @pid INTEGER;
```

```
    SET @pid = (SELECT p.product_ID from Product1 p where p.product_ID = @product_ID);
    IF @pid = @product_ID
        INSERT INTO Prod1_Acc VALUES (@account_no, @cost);
        INSERT INTO Cost VALUES (@product_ID, @account_no);
    IF @pid != @product_ID
        DECLARE @Msg VARCHAR(300);
        SET @Msg = 'Product type mismatch';
        PRINT @Msg;
END
GO

-- procedure to insert account type 2 details
CREATE PROCEDURE New_Account2
@account_no INTEGER,
@est_date DATE,
@cost REAL,
@product_ID INTEGER
AS
BEGIN
    INSERT INTO Account VALUES (@account_no, @est_date);
    DECLARE @pid INTEGER;
    SET @pid = (SELECT p.product_ID from Product2 p where p.product_ID = @product_ID);
    IF @pid = @product_ID
        INSERT INTO Prod2_Acc VALUES (@account_no, @cost);
        INSERT INTO Cost VALUES (@product_ID, @account_no);
    IF @pid != @product_ID
        DECLARE @Msg VARCHAR(300);
        SET @Msg = 'Product type mismatch';
        PRINT @Msg;
END
GO

-- procedure to insert account type 3 details
CREATE PROCEDURE New_Account3
@account_no INTEGER,
@est_date DATE,
@cost REAL,
@product_ID INTEGER
AS
BEGIN
    INSERT INTO Account VALUES (@account_no, @est_date);
    DECLARE @pid INTEGER;
    SET @pid = (SELECT p.product_ID from Product3 p where p.product_ID = @product_ID);
    IF @pid = @product_ID
        INSERT INTO Prod3_Acc VALUES (@account_no, @cost);
        INSERT INTO Cost VALUES (@product_ID, @account_no);
    IF @pid != @product_ID
        DECLARE @Msg VARCHAR(300);
        SET @Msg = 'Product type mismatch';
        PRINT @Msg;
END
GO
```

5) Enter a complaint associated with a customer and product

```
CREATE PROCEDURE New_Complaint
@complaint_ID INTEGER,
@date_of_complaint DATE,
@complaint_desc VARCHAR(64),
@t_ind INTEGER,
@product_ID INTEGER,
@cust_name VARCHAR(64),
@ts_name VARCHAR(64)
AS
BEGIN
    IF @t_ind = 1
        INSERT INTO Complaint VALUES (@complaint_ID, @date_of_complaint, @complaint_desc,
'Refund');
        INSERT INTO Defect VALUES (@complaint_ID, @product_ID, @cust_name);
        INSERT INTO Rep_Compl VALUES (@complaint_ID, @product_ID, @ts_name);
    IF @t_ind = 2
        INSERT INTO Complaint VALUES (@complaint_ID, @date_of_complaint, @complaint_desc,
'Exchange');
        INSERT INTO Defect VALUES (@complaint_ID, @product_ID, @cust_name);
        INSERT INTO Rep_Compl VALUES (@complaint_ID, @product_ID, @ts_name);
END
GO
```

6) Enter an accident associated with an appropriate employee and product

```
CREATE PROCEDURE New_Complaint
@complaint_ID INTEGER,
@date_of_complaint DATE,
@complaint_desc VARCHAR(64),
--@treatment VARCHAR(64),
@t_ind INTEGER,
@product_ID INTEGER,
@cust_name VARCHAR(64),
@ts_name VARCHAR(64)
AS
BEGIN
    IF @t_ind = 1
        INSERT INTO Complaint VALUES (@complaint_ID, @date_of_complaint, @complaint_desc,
'Refund');
        INSERT INTO Defect VALUES (@complaint_ID, @product_ID, @cust_name);
        INSERT INTO Rep_Compl VALUES (@complaint_ID, @product_ID, @ts_name);
    IF @t_ind = 2
        INSERT INTO Complaint VALUES (@complaint_ID, @date_of_complaint, @complaint_desc,
'Exchange');
        INSERT INTO Defect VALUES (@complaint_ID, @product_ID, @cust_name);
        INSERT INTO Rep_Compl VALUES (@complaint_ID, @product_ID, @ts_name);
END
GO

-- 6 --
CREATE PROCEDURE New_Accident
@accident_no INTEGER,
@accident_dt DATE,
@days_lost INTEGER,
@product_ID INTEGER
AS
BEGIN
    INSERT INTO Accident VALUES (@accident_no, @accident_dt, @days_lost);

END
```

```
GO

INSERT INTO Repair_Accident VALUES (?, ?, ?);

INSERT INTO Prod_Accident VALUES (?, ?, ?);
```

7) Retrieve the date produced and time spent to produce a particular product

```
SELECT P.prod_date, P.time_taken
FROM Product P
WHERE P.product_ID = ?;
```

8) Retrieve all products made by a particular worker

```
SELECT  *
FROM Product P
WHERE P.worker_name = ?;
```

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

```
SELECT COUNT(C.complaint_ID)
FROM Complaint C
INNER JOIN Defect D ON C.complaint_ID = D.complaint_ID
INNER JOIN Product P ON D.product_ID = P.product_ID
WHERE P.qc_name = ?;
```

10) Retrieve the total costs of the products in the product3 category which were repaired at the request of a particular quality controller

```
SELECT SUM(P.cost_prod3)
FROM Request R
INNER JOIN Cost C ON R.product_ID = C.product_ID
INNER JOIN Prod3_Acc P ON C.account_no = P.account_no
WHERE R.qc_name = ?;
```

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

```
SELECT C.*
FROM Customer C
INNER JOIN Purchase P ON C.cust_name = P.cust_name
INNER JOIN Product2 Pr ON Pr.product_ID = P.product_ID
WHERE Pr.color = ?
ORDER BY C.cust_name;
```

12) Retrieve all employees whose salary is above a particular salary

```
SELECT *
FROM Employee E
WHERE E.emp_sal > ?;
```

13) Retrieve the total number of work days lost due to accidents in repairing the products which got complaints

```
SELECT SUM(A.days_lost)
FROM Accident A
INNER JOIN Repair_Accident Ra ON A.accident_no = Ra.accident_no
INNER JOIN Rep_Compl R ON R.product_ID = Ra.product_ID;
```

14) Retrieve the average cost of all products made in a particular year

```
SELECT AVG(T.cost) as avg_cost FROM
(SELECT P.product_ID as ID, P1.cost_prod1 as cost, YEAR(P.prod_date)  as year
FROM Product P
```

```
INNER JOIN Cost C ON P.product_ID = C.product_ID
INNER JOIN Prod1_Acc P1 ON C.account_no = P1.account_no
UNION
SELECT P.product_ID as ID, P2.cost_prod2 as cost, YEAR(P.prod_date)  as year
FROM Product P
INNER JOIN Cost C ON P.product_ID = C.product_ID
INNER JOIN Prod2_Acc P2 ON C.account_no = P2.account_no
UNION
SELECT P.product_ID as ID, P3.cost_prod3 as cost, YEAR(P.prod_date)  as year
FROM Product P
INNER JOIN Cost C ON P.product_ID = C.product_ID
INNER JOIN Prod3_Acc P3 ON C.account_no = P3.account_no
) T
WHERE T.year = ?
```

15) Delete all accidents whose dates are in some range

```
DELETE FROM Repair_Accident WHERE accident_no = (SELECT accident_no FROM Accident
WHERE accident_dt BETWEEN ? AND ?);

DELETE FROM Prod_Accident WHERE accident_no = (SELECT accident_no FROM Accident
WHERE accident_dt BETWEEN ? AND ?);

DELETE FROM Accident
WHERE accident_dt BETWEEN ? AND ?;
```

5.2 The Java source program and screenshots showing its successful compilation

```
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;


public class Mohanty_Sonaxy_IP_Task5b {

   // Database credentials
   final static String HOSTNAME = " xxxx ";
   final static String DBNAME = " xxxx ";
   final static String USERNAME = " xxxx ";
   final static String PASSWORD = " xxxx ";

   // Database connection string
   final static String URL =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustS
erverCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
```

```
        HOSTNAME, DBNAME, USERNAME, PASSWORD);

 // Query templates
        final static String QUERY_2a = "INSERT INTO Product (product_ID, prod_date,
time_taken, worker_name, qc_name) VALUES"
                        + "(?, ?, ?, ?, ?)";
        final static String QUERY_2c = "INSERT INTO Repair VALUES (?, ?, ?)";
        final static String QUERY_3a = "INSERT INTO Customer VALUES (?, ?)";
        final static String QUERY_3b = "INSERT INTO Purchase VALUES (?, ?)";
        final static String QUERY_6a = "INSERT INTO Repair_Accident VALUES (?, ?, ?)";
        final static String QUERY_6b = "INSERT INTO Prod_Accident VALUES (?, ?, ?)";
        final static String QUERY_7 = "SELECT P.prod_date, P.time_taken \r\n"
                        + "FROM Product P\r\n"
                        + "WHERE P.product_ID = ?";
        final static String QUERY_8 = "SELECT  *\r\n"
                        + "FROM Product P\r\n"
                        + "WHERE P.worker_name = ?";
        final static String QUERY_9 = "SELECT COUNT(C.complaint_ID)\r\n"
                        + "FROM Complaint C\r\n"
                        + "INNER JOIN Defect D ON C.complaint_ID = D.complaint_ID\r\n"
                        + "INNER JOIN Product P ON D.product_ID = P.product_ID\r\n"
                        + "WHERE P.qc_name = ?";
        final static String QUERY_10 = "SELECT SUM(P.cost_prod3)\r\n"
                        + "FROM Request R\r\n"
                        + "INNER JOIN Cost C ON R.product_ID = C.product_ID\r\n"
                        + "INNER JOIN Prod3_Acc P ON C.account_no = P.account_no\r\n"
                        + "WHERE R.qc_name = ?";
        final static String QUERY_11 = "SELECT C.*  \r\n"
                        + "FROM Customer C\r\n"
                        + "INNER JOIN Purchase P ON C.cust_name = P.cust_name\r\n"
                        + "INNER JOIN Product2 Pr ON Pr.product_ID = P.product_ID\r\n"
                        + "WHERE Pr.color = ?\r\n"
                        + "ORDER BY C.cust_name";
        final static String QUERY_12 = "SELECT *\r\n"
                        + "FROM Employee E\r\n"
                        + "WHERE E.emp_sal > ?";
        final static String QUERY_13 = "SELECT SUM(A.days_lost) \r\n"
                        + "FROM Accident A\r\n"
                        + "INNER JOIN Repair_Accident Ra ON A.accident_no =
Ra.accident_no\r\n"
                        + "INNER JOIN Rep_Compl R ON R.product_ID = Ra.product_ID";
        final static String QUERY_14 = "SELECT AVG(T.cost) as avg_cost FROM\r\n"
                        + "(SELECT P.product_ID as ID, P1.cost_prod1 as cost,
YEAR(P.prod_date)  as year\r\n"
                        + "FROM Product P\r\n"
                        + "INNER JOIN Cost C ON P.product_ID = C.product_ID\r\n"
                        + "INNER JOIN Prod1_Acc P1 ON C.account_no = P1.account_no\r\n"
                        + "UNION\r\n"
```

```
                        + "SELECT P.product_ID as ID, P2.cost_prod2 as cost, YEAR(P.prod_date)
as year\r\n"
                        + "FROM Product P\r\n"
                        + "INNER JOIN Cost C ON P.product_ID = C.product_ID\r\n"
                        + "INNER JOIN Prod2_Acc P2 ON C.account_no = P2.account_no\r\n"
                        + "UNION\r\n"
                        + "SELECT P.product_ID as ID, P3.cost_prod3 as cost, YEAR(P.prod_date)
as year \r\n"
                        + "FROM Product P\r\n"
                        + "INNER JOIN Cost C ON P.product_ID = C.product_ID\r\n"
                        + "INNER JOIN Prod3_Acc P3 ON C.account_no = P3.account_no\r\n"
                        + ") T\r\n"
                        + "WHERE T.year = ?";
        final static String QUERY_15a = "DELETE FROM Repair_Accident WHERE accident_no =
(SELECT accident_no FROM Accident\r\n"
                        + "WHERE accident_dt BETWEEN ? AND ?);";
        final static String QUERY_15b = "DELETE FROM Prod_Accident WHERE accident_no =
(SELECT accident_no FROM Accident\r\n"
                        + "WHERE accident_dt BETWEEN ? AND ?);";
        final static String QUERY_15c = "DELETE FROM Accident \r\n"
                        + "WHERE accident_dt BETWEEN ? AND ?;";
        final static String QUERY_16 = "INSERT INTO Employee VALUES (?, ?, ?)";
        final static String QUERY_17 = "SELECT C.*  \r\n"
                        + "FROM Customer C\r\n"
                        + "INNER JOIN Purchase P ON C.cust_name = P.cust_name\r\n"
                        + "INNER JOIN Product2 Pr ON Pr.product_ID = P.product_ID\r\n"
                        + "WHERE Pr.color = 'Red'\r\n"
                        + "ORDER BY C.cust_name";

 // 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 associate 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; \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(exit the program)!";

    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 18 is selected
            System.out.println(PROMPT); // Print the available options
            option = sc.next(); // Read in the user option selection

            switch (option) { // Switch between different options
                case "1": // Insert a new employee option
                    // Collect the new employee data from the user
                    System.out.println("Please enter employee name:");
                    // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                    // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                    sc.nextLine();
                    final String emp_name = sc.nextLine(); // Read in user input of employee name
(white-spaces allowed).

                    System.out.println("Please enter employee address:");
                    // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                    // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                    //sc.nextLine();
                    final String emp_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


                    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
New_Employee @emp_name = ?, @emp_address = ?, @salary = ?;")) {
              // Populate the stored procedure with the data collected from the user
              statement.setString(1, emp_name);
              statement.setString(2, emp_address);
              statement.setFloat(3, salary);

             // No need to Call the stored procedure here
              //ResultSet resultSet = statement.executeQuery();

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

          System.out.println("Is employee technical staff?");
          System.out.println("Enter 1 for Yes 2 for No:");
          // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
          // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
          sc.nextLine();
          final String ts_ind = sc.nextLine(); // Read in user input .

          String QUERY_1a = "";
                              if(ts_ind.equals("1"))
                              {
                                    //Set the query to call a procedure to create a Technical
Staff employee
                                    QUERY_1a = "EXEC New_Employee_Tech ?,?,?,?,?";


                                    //Collect the required inputs for the technical staff
                                    System.out.println("Please enter technical position :");
                                    //sc.nextLine();
            final String tech_position = sc.nextLine(); // Read in the user input of technical
position

                                    System.out.println("Does he/she has a BS degree?");
                                    System.out.println("Enter 1 for Yes 2 for No:");
                                    //sc.nextLine();
                                    final String BS_ind = sc.nextLine(); // Read in user input

                                    System.out.println("Does he/she has a MS degree?");
                                    System.out.println("Enter 1 for Yes 2 for No:");
```

```
                                        //sc.nextLine();
                                        final String MS_ind = sc.nextLine(); // Read in user input

                                        System.out.println("Does he/she has a PhD degree?");
                                        System.out.println("Enter 1 for Yes 2 for No:");
                                        //sc.nextLine();
                                        final String PhD_ind = sc.nextLine(); // Read in the user
input

                                        System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_1a))
                                                {
                                                        // Populate the query template with the
data collected from the user
                                                        statement.setString(1, emp_name);
                                                        statement.setString(2, tech_position);
                                                        statement.setString(3, BS_ind);
                                                        statement.setString(4, MS_ind);
                                                        statement.setString(5, PhD_ind);

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

                                }
                                        else
                                        {
                                                // Unrecognized option, re-prompt the user for
the correct one
                                                System.out.println("No insert into the table.");

                                        }

                                        System.out.println("Is employee quality controller?");
                                        System.out.println("Enter 1 for Yes 2 for No:");
```

```
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
            //sc.nextLine();
            final String qc_ind = sc.nextLine(); // Read in user input

            String QUERY_1b = "";
                        if(qc_ind.equals("1"))
                        {
                                    //Set the query to call a procedure to create a Quality
Controller employee
                                    QUERY_1b = "EXEC New_Employee_QC ?,?";

                                    //Collect the required inputs
                                    System.out.println("Please enter product type as
Product 1/Product 2/Product 3 :");
                                    //sc.nextLine();
                final String prod_type = sc.nextLine(); // Read in the user input of 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(QUERY_1b))
                                                {
                                                            // Populate the query template with the
data collected from the user
                                                            statement.setString(1, emp_name);
                                                            statement.setString(2, prod_type);

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

                        }
                                    else
                                    {
```

```
                                        // Unrecognized option, re-prompt the user for
the correct one
                                        System.out.println("No insert into the table.");

                                }

                                System.out.println("Is employee worker?");
                                System.out.println("Enter 1 for Yes 2 for No:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent
nextLine doesn't return nothing.
            //sc.nextLine();
            final String worker_ind = sc.nextLine(); // Read in user input

            String QUERY_1c = "";
                        if(worker_ind.equals("1"))
                        {
                                //Set the query to call a procedure to create a Worker
employee
                                QUERY_1c = "EXEC New_Employee_Worker ?,?";

                                //Collect the required inputs
                                System.out.println("Please enter  maximum number of
products a worker can produce per day :");
                   final int max_produce = sc.nextInt(); // Read in the user input for max hrs/day
a worker works


                                System.out.println("Connecting to the database...");
                                // Get a database connection and prepare a query
statement
                                try (final Connection connection =
DriverManager.getConnection(URL))
                                {
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_1c))
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setString(1, emp_name);
                                                statement.setInt(2, max_produce);

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

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

                }
                        else
                        {
                                // Unrecognized option, re-prompt the user for
the correct one
                                System.out.println("No insert into the table.");

                        }


                break;
            case "2": // Insert a new product option
                    System.out.println("Please enter product ID:");
                // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                final int product_ID = sc.nextInt(); // Read in user input

                System.out.println("Please enter production date(YYYY-MM-DD):");
                // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String prod_date = sc.nextLine(); // Read in user input

                System.out.println("Please enter time spent(hh:mm:ss) to make the product:");
                final String time_taken = sc.nextLine(); // Read in user input

                System.out.println("Please enter employee who produced the product:");
                final String worker_name = sc.nextLine(); // Read in user input

                System.out.println("Please enter employee who tested the product:");
                final String qc_name = sc.nextLine(); // Read in user input

                System.out.println("Is the product repaired?");
                System.out.println("Enter 1 for Yes 2 for No");
                final String rep_ind = sc.nextLine(); //if the product is repaired then enter repairing
details for the associated tables

                if(rep_ind.equals("1")) {
```

```
            System.out.println("Please enter employee who repaired the product if product is
repaired:");
            final String ts_name = sc.nextLine();

            System.out.println("Please enter repair date(YYYY-MM-DD):");
            final String repair_dt = 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
New_Product_Repair @product_ID = ?, @prod_date = ?, @time_taken = ?, @worker_name = ?,
@qc_name = ?, @ts_name = ?;")) {
                  // Populate the stored procedure with the data collected from the user
                  statement.setInt(1, product_ID);
                  statement.setString(2, prod_date);
                  statement.setString(3, time_taken);
                  statement.setString(4, worker_name);
                  statement.setString(5, qc_name);
                  statement.setString(6, ts_name);

                // No need to Call the stored procedure here
                  //ResultSet resultSet = statement.executeQuery();

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

               }
            }

            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
               try (
                  final PreparedStatement statement = connection.prepareStatement(QUERY_2c))
{
                  // Populate the query with the data collected from the user
                  statement.setInt(1, product_ID);
                  statement.setString(2, ts_name);
                  statement.setString(3, repair_dt);

                // No need to Call the stored procedure here
                  //ResultSet resultSet = statement.executeQuery();

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

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

            }
        }
        System.out.println("Is repair requested by a quality controller?");
        System.out.println("Enter 1 for Yes and 2 for No");
        final String req_ind = sc.nextLine();

        if(req_ind.equals("1")) {
             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
New_Request @product_ID = ?, @qc_name = ?, @ts_name = ?;")) {
                 // Populate the stored procedure with the data collected from the user
                 statement.setInt(1, product_ID);
                 statement.setString(2, qc_name);
                 statement.setString(3, ts_name);

                // No need to Call the stored procedure here
                 //ResultSet resultSet = statement.executeQuery();

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

              }
           }
        }
        else
                                        {
                                                // Unrecognized option, re-prompt the user for
the correct one
                                                System.out.println("No insert into the table.");

                                        }

        }
        // if the product is not repaired then product table is updated without any technical
staff info
        else {
             System.out.println("Connecting to the database...");
                                // Get a database connection and prepare a query statement
```

```
                                    try (final Connection connection =
DriverManager.getConnection(URL))
                                    {
                                            try (final PreparedStatement statement =
connection.prepareStatement(QUERY_2a))
                                            {
                                                    // Populate the query template with the data
collected from the user
                                                    statement.setInt(1, product_ID);
                                                    statement.setString(2, prod_date);
                                                    statement.setString(3, time_taken);
                                                    statement.setString(4,  worker_name);
                                                    statement.setString(5, qc_name);

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

            }
            // the below associated tables are updated with the user info
            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
New_Produce @product_ID = ?, @worker_name = ?;")) {
                  // Populate the stored procedure with the data collected from the user
                  statement.setInt(1, product_ID);
                  statement.setString(2, worker_name);

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

            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
New_Certify @product_ID = ?, @qc_name = ?;")) {
```

```
                    // Populate the stored procedure with the data collected from the user
                    statement.setInt(1, product_ID);
                    statement.setString(2, qc_name);

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

            System.out.println("Is product of type 1/ type 2/ type 3?");
            System.out.println("Enter 1 for type 1, 2 for type 2, 3 for type 3:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
            //sc.nextLine();
            final String p_ind = sc.nextLine(); // Read in user input for product type and update
the appropriate product-type table

            String QUERY_2b = "";
                                    if(p_ind.equals("1"))
                                    {
                                            //Set the query to call a procedure to create Product of
type 1
                                            QUERY_2b = "EXEC New_Product1 ?,?,?";


                                            //Collect the required inputs for the type 1 product
                                            System.out.println("Please enter size of the product:");
                                            //sc.nextLine();
                final int size = sc.nextInt(); // Read in the user input

                                            System.out.println("Please enter major software
used:");
                                            sc.nextLine();
                                            final String major_software = sc.nextLine(); // Read in
user input

                                            System.out.println("Connecting to the database...");
                                            // Get a database connection and prepare a query
statement
                                            try (final Connection connection =
DriverManager.getConnection(URL))
                                            {
                                                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_2b))
```

```
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setInt(1, product_ID);
                                                statement.setInt(2, size);
                                                statement.setString(3, major_software);
                                                //statement.setString(4, MS_ind);
                                                //statement.setString(5, PhD_ind);

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

                        }

                        else if(p_ind.equals("2"))
                        {
                                //Set the query to call a procedure to create product of
type 2
                                QUERY_2b = "EXEC New_Product2 ?,?,?";

                                System.out.println("Please enter size of the product:");
                                final int size = sc.nextInt();

                                System.out.println("Please enter color of the product:");
                                sc.nextLine();
                                final String color = 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(QUERY_2b))
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setInt(1, product_ID);
                                                statement.setInt(2, size);
```

```
                                        statement.setString(3, color);

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

                            }
                            else if(p_ind.equals("3"))
                            {
                                //Set the query to call a procedure to create a product of
type 3
                                QUERY_2b = "EXEC New_Product3 ?,?,?";

                                System.out.println("Please enter size of the product:");
                                final int size = sc.nextInt();

                                System.out.println("Please enter weight of the
product:");
                                final int prod_weight = sc.nextInt();

                                System.out.println("Connecting to the database...");
                                // Get a database connection and prepare a query
statement
                                try (final Connection connection =
DriverManager.getConnection(URL))
                                {
                                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_2b))
                                    {
                                        // Populate the query template with the
data collected from the user
                                        statement.setInt(1, product_ID);
                                        statement.setInt(2, size);
                                        statement.setInt(3, prod_weight);

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

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

                            }
                            else
                            {
                                // Unrecognized option, re-prompt the user for the
correct one
                                System.out.println("Unrecognized option! Please select
the option and try again.");
                                break;
                            }

        case "3": // Insert a new customer option
                System.out.println("Please enter customer name:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
            sc.nextLine();
                final String cust_name = sc.nextLine(); // Read in user input

            System.out.println("Please enter customer address:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
            final String cust_address = 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(QUERY_3a))
                                    {
                                        // Populate the query template with the
data collected from the user
                                        statement.setString(1, cust_name);
                                        statement.setString(2, cust_address);

                                        System.out.println("Dispatching the
query...");
                                        // Actually execute the populated query
```

```
                                        final int rows_inserted =
statement.executeUpdate();
                                        System.out.println(String.format("Done.
%d rows inserted.", rows_inserted));
                                }
                        }
                        // populating associated purchase table for the new
customer
                        System.out.println("Enter number of products customer
brought:");
                        final int n_prod = sc.nextInt();

                        int i = 0;
                        while(i < n_prod) {
                                System.out.println("Please enter the product
ID");
                                final int p_id = sc.nextInt();
                                System.out.println("Connecting to the
database...");
                                // Get a database connection and prepare a
query statement
                                try (final Connection connection =
DriverManager.getConnection(URL))
                                {
                                        try (final PreparedStatement statement
= connection.prepareStatement(QUERY_3b))
                                        {
                                                // Populate the query template
with the data collected from the user
                                                statement.setInt(1, p_id);
                                                statement.setString(2,
cust_name);

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

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

                        }
                        break;
         case "4": // Insert a new account option
```

```
                System.out.println("Please enter account no:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                final int account_no = sc.nextInt(); // Read in user input

            System.out.println("Please enter the date the account established:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
            sc.nextLine();
            final String est_date = sc.nextLine();

            System.out.println("Please enter the cost of the product:");
            final float cost = sc.nextFloat();

            System.out.println("Please enter the product ID associated with this account:");
            final int prod_ID = sc.nextInt();
            // user-input to recognize the type of account created
            System.out.println("Please enter the account type:");
            System.out.println("Enter 1 for product1-account, 2 for product2-account, 3 for
product3-account");
            sc.nextLine();
            final String acc_ind = sc.nextLine();

            String QUERY_4 = "";
            if (acc_ind.equals("1")) {
                QUERY_4 = "EXEC New_Account1 ?,?,?,?"; //account is created for product type
1
                System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_4))
                                                {
                                                        // Populate the query template with the
data collected from the user
                                                        statement.setInt(1, account_no);
                                                        statement.setString(2, est_date);
                                                        statement.setFloat(3, cost);
                                                        statement.setInt(4,  prod_ID);
```

```
                                                System.out.println("Dispatching the
query...");
                                                // Actually execute the populated query
                                                final int rows_inserted =
statement.executeUpdate();
                                                System.out.println(String.format("Done.
%d rows inserted.", rows_inserted));
                                        }
                                    }
                                    break;
            }
            else if (acc_ind.equals("2")) {
                QUERY_4 = "EXEC New_Account2 ?,?,?,?"; //account is created for product type
2
                System.out.println("Connecting to the database...");
                                    // Get a database connection and prepare a query
statement
                                    try (final Connection connection =
DriverManager.getConnection(URL))
                                    {
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_4))
                                        {
                                            // Populate the query template with the
data collected from the user
                                            statement.setInt(1, account_no);
                                            statement.setString(2, est_date);
                                            statement.setFloat(3, cost);
                                            statement.setInt(4,  prod_ID);

                                            System.out.println("Dispatching the
query...");
                                            // Actually execute the populated query
                                            final int rows_inserted =
statement.executeUpdate();
                                            System.out.println(String.format("Done.
%d rows inserted.", rows_inserted));
                                        }
                                    }
                                    break;
            }
            else if (acc_ind.equals("3")) { //account is created for product type 3
                QUERY_4 = "EXEC New_Account3 ?,?,?,?";
                System.out.println("Connecting to the database...");
                                    // Get a database connection and prepare a query
statement
                                    try (final Connection connection =
DriverManager.getConnection(URL))
```

```
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_4))
                                                {
                                                        // Populate the query template with the
data collected from the user
                                                        statement.setInt(1, account_no);
                                                        statement.setString(2, est_date);
                                                        statement.setFloat(3, cost);
                                                        statement.setInt(4,  prod_ID);


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


                else
                                {
                                        // Unrecognized option, re-prompt the user for the
correct one
                                        System.out.println("Unrecognized option! Please select
the option and try again.");
                                        break;
                                }
        case "5": // Insert a new complaint option
                System.out.println("Please enter complaint ID:");

                final int complaint_ID = sc.nextInt(); // Read in user input

                System.out.println("Please enter the date of the complaint:");
                // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String date_of_complaint = sc.nextLine();

                System.out.println("Please enter complaint description:");
                final String complaint_desc = sc.nextLine();
```

```
            System.out.println("Please enter treatment expected:");
            System.out.println("Enter 1 for Refund or 2 for Exchange");
            final int t_ind = sc.nextInt();

            System.out.println("Please enter the product ID for which the complaint is raise:");
            final int pID = sc.nextInt();

            System.out.println("Please enter the customer name who raised the complaint:");
            sc.nextLine();
            final String c_name = sc.nextLine();

            System.out.println("Please enter the technician name who will work on this
complaint");
            final String t_name = 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 New_Complaint @complaint_ID=?,
@date_of_complaint=?, @complaint_desc=?, @t_ind=?, @product_ID=?, @cust_name=?,
@ts_name=?;"))
                                            {
                                                    // Populate the query template with the
data collected from the user
                                                    statement.setInt(1, complaint_ID);
                                                    statement.setString(2,
date_of_complaint);
                                                    statement.setString(3, complaint_desc);
                                                    statement.setInt(4,  t_ind);
                                                    statement.setInt(5, pID);
                                                    statement.setString(6, c_name);
                                                    statement.setString(7, t_name);

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

```
            case "6": // Insert a new accident option
                    System.out.println("Please enter accident number:");
                final int accident_no = sc.nextInt(); // Read in user input

                System.out.println("Please enter the date of accident:");
                // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String accident_dt = sc.nextLine();

                System.out.println("Please enter number of work days lost due to the accident:");
                final int days_lost = sc.nextInt();

                System.out.println("Please enter product ID with which the accident is associated:");
                final int prid = sc.nextInt();

                System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement("EXEC New_Accident @accident_no=?, @accident_dt=?,
@days_lost=?, @product_ID=?;"))
                                                {
                                                        // Populate the query template with the
data collected from the user
                                                        statement.setInt(1, accident_no);
                                                        statement.setString(2, accident_dt);
                                                        statement.setInt(3, days_lost);
                                                        statement.setInt(4, prid);

                                                        System.out.println("Dispatching the
query...");
                                                        // Actually execute the populated query
                                                        final int rows_inserted =
statement.executeUpdate();
                                                        System.out.println(String.format("Done.
%d rows inserted.", rows_inserted));
                                                }
                                        }
                                        // user-input will determine which type of accident has
taken place and related space-holders will be updated
                                        System.out.println("Please enter reason of accident:");
                                        System.out.println("1 for Repair or 2 Produce:");
```

```
                                        sc.nextLine();
                                        final String a_ind = sc.nextLine();

                                        if(a_ind.equals("1")) {
                                                System.out.println("Please enter the technical
staff involved in the accident:");
                                                final String rep_acc = 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(QUERY_6a))
                                                        {
                                                                // Populate the query template
with the data collected from the user
                                                                statement.setInt(1,
accident_no);
                                                                statement.setString(2, rep_acc);
                                                                statement.setInt(3,  prid);

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

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

                                        else if(a_ind.equals("2")) {
                                                System.out.println("Please enter the worker
involved in the accident:");
                                                final String prod_acc = 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(QUERY_6b))
                                                {
                                                        // Populate the query template
with the data collected from the user
                                                        statement.setInt(1,
accident_no);
                                                        statement.setString(2,
prod_acc);
                                                        statement.setInt(3,  prid);

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

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

                                else
                                {
                                        // Unrecognized option, re-prompt the user for the
correct one
                                        System.out.println("Unrecognized option! Please select
the option and try again.");
                                        break;
                                }

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

             System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_7))
```

```
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setInt(1, pr);

                                                ResultSet resultSet =
statement.executeQuery();
                System.out.println("Contents of the Product table:");
                System.out.println("date produced | time spent to produce ");

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

            case "8": // products made by a particular worker option
                    System.out.println("Please enter worker name:");
              // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
              // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                    sc.nextLine();
                    final String w = sc.nextLine(); // Read in user input

              System.out.println("Connecting to the database...");
                                    // Get a database connection and prepare a query
statement
                                    try (final Connection connection =
DriverManager.getConnection(URL))
                                    {
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_8))
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setString(1, w);

                                                ResultSet resultSet =
statement.executeQuery();
                System.out.println("Contents of the Product table:");
                System.out.println("product ID | date produced | time spent | worker | quality
controller | technical staff (if any) ");

                while (resultSet.next()) {
```

```
                System.out.println(String.format("%s | %s | %s | %s | %s | %s  ",
                    resultSet.getString(1),
                    resultSet.getString(2),
                    resultSet.getString(3),
                    resultSet.getString(4),
                    resultSet.getString(5),
                    resultSet.getString(6)));
                }
                                                    }
                                                }
                                                break;

        case "9": // total number of errors a particular quality controller made option
                System.out.println("Please enter quality controller name:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String q = sc.nextLine(); // Read in user input

            System.out.println("Connecting to the database...");
                                                // Get a database connection and prepare a query
statement
                                                try (final Connection connection =
DriverManager.getConnection(URL))
                                                {
                                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_9))
                                                        {
                                                                // Populate the query template with the
data collected from the user
                                                                statement.setString(1, q);

                                                                ResultSet resultSet =
statement.executeQuery();
                System.out.println("total no.of errors");

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

        case "10": // total costs of the products in the product3 category which were repaired
                //at the request of a particular quality controller option
```

```
                System.out.println("Please enter quality controller name:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String qc = sc.nextLine(); // Read in user input

            System.out.println("Connecting to the database...");
                                    // Get a database connection and prepare a query
statement
                                    try (final Connection connection =
DriverManager.getConnection(URL))
                                    {
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_10))
                                        {
                                            // Populate the query template with the
data collected from the user
                                            statement.setString(1, qc);

                                            ResultSet resultSet =
statement.executeQuery();
                    System.out.println("total cost");

                    while (resultSet.next()) {
                       System.out.println(String.format("%s",
                          resultSet.getString(1)));
                    }
                                        }
                                    }
                                    break;
          case "11": // all customers (in name order) who purchased all products of a particular
color option
                System.out.println("Please enter product color:");
            // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
            // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                sc.nextLine();
                final String c = sc.nextLine(); // Read in user input

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

```
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_11))
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setString(1, c);

                                                ResultSet resultSet =
statement.executeQuery();
                System.out.println("customer name | address");

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

        case "12": // all employees whose salary is above a particular salary option
                 System.out.println("Please enter salary:");
           final float s = sc.nextFloat(); // Read in user input

           System.out.println("Connecting to the database...");
                                    // Get a database connection and prepare a query
statement
                                    try (final Connection connection =
DriverManager.getConnection(URL))
                                    {
                                        try (final PreparedStatement statement =
connection.prepareStatement(QUERY_12))
                                        {
                                                // Populate the query template with the
data collected from the user
                                                statement.setFloat(1, s);

                                                ResultSet resultSet =
statement.executeQuery();
                System.out.println("employee name | address | salary");

                while (resultSet.next()) {
                   System.out.println(String.format("%s | %s | %s",
                      resultSet.getString(1),
                      resultSet.getString(2),
                      resultSet.getString(3)));
                }
                                        }
```

```
                                        }
                                        break;

            case "13": // total number of work days lost due to accidents in repairing the products
which got complaints option
                        System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_13))
                                                {
                                                        ResultSet resultSet =
statement.executeQuery();
                        System.out.println("total number of work days lost");

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

            case "14": // average cost of all products made in a particular year option
                        System.out.println("Please enter production year:");
                      // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
                     // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                        sc.nextLine();
                        final String yr = sc.nextLine(); // Read in user input

                    System.out.println("Connecting to the database...");
                                        // Get a database connection and prepare a query
statement
                                        try (final Connection connection =
DriverManager.getConnection(URL))
                                        {
                                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_14))
                                                {
                                                        // Populate the query template with the
data collected from the user
                                                        statement.setString(1, yr);
```

```
                                                    ResultSet resultSet =
statement.executeQuery();
                System.out.println("average cost");

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

          case "15": // all accidents whose dates are in some range
                    System.out.println("Please enter start date:");
              // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
              // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                    sc.nextLine();
                    final String s_dt = sc.nextLine(); // Read in user input

                    System.out.println("Please enter end date:");
              // Preceding nextInt, nextFloar, etc. do not consume new line characters from the
user input.
              // We call nextLine to consume that newline character, so that subsequent nextLine
doesn't return nothing.
                    final String e_dt = 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(QUERY_15a))
                                                        {
                                                                // Populate the query template with the
data collected from the user
                                                                statement.setString(1, s_dt);
                                                                statement.setString(2,  e_dt);


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


                                                                // Actually execute the delete query for
repair related accidents
```

```
                                        final int rows_deleted =
statement.executeUpdate();
                                        System.out.println(String.format("Done.
QUERY RESULT :%d", rows_deleted));

                }
                                }
                        System.out.println("Connecting to the database...");
                        // Get a database connection and prepare a query
statement
                        try (final Connection connection =
DriverManager.getConnection(URL))
                        {
                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_15b))
                                {
                                        // Populate the query template with the
data collected from the user
                                        statement.setString(1, s_dt);
                                        statement.setString(2,  e_dt);


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

                                        // Actually execute the delete query for
production related accidents
                                        final int rows_deleted =
statement.executeUpdate();
                                        System.out.println(String.format("Done.
QUERY RESULT :%d", rows_deleted));

                }
                                }
                        System.out.println("Connecting to the database...");
                        // Get a database connection and prepare a query
statement
                        try (final Connection connection =
DriverManager.getConnection(URL))
                        {
                                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_15c))
                                {
                                        // Populate the query template with the
data collected from the user
                                        statement.setString(1, s_dt);
                                        statement.setString(2,  e_dt);
```

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

                                                // Actually execute the delete query of
accident table (primary table)
                                                final int rows_deleted =
statement.executeUpdate();
                                                System.out.println(String.format("Done.
QUERY RESULT :%d", rows_deleted));

                }
                                        }

                                    break;

        case "16":
                                //Import data from file to Employee Table

                                //Capture file name to store the output
                                System.out.println("Please enter the file name:");
                                sc.nextLine();
                                final String importFile = sc.nextLine(); // Read in user input - file
name

                                // Get a database connection and prepare a query statement
                                try (final Connection connection =
DriverManager.getConnection(URL))
                                {
                                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_16))
                                    {
                                        //CSV Reader to read the input file
                                        BufferedReader csvFileReader = new
BufferedReader(new FileReader(importFile));
                                        String currentLine = null;
                                        int recordCount = 0;
                                        //IGNORE HEADER
                                        csvFileReader.readLine();

                                        //Loop to read the file till the end
                                        while((currentLine = csvFileReader.readLine()) !=
null)
                                        {
                                            //currentLine =
csvFileReader.readLine();
                                            System.out.println(currentLine);
                                            String record[] = currentLine.split(",");
```

```
                                                // Populate data collected from the file
                                                statement.setString(1, record[0]);
                                                statement.setString(2, record[1]);

        statement.setFloat(3,Integer.parseInt(record[2]));

                                                //System.out.println("Dispatching the
query...");
                                                // Actually execute the populated query
                                                statement.executeUpdate();

        //System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                                                recordCount++;
                                        }
                                        System.out.println(recordCount+" records have
been inserted successfully!");
                                        csvFileReader.close(); //Close reader
                                        break;
                                }
                                catch (SQLException e)
                                {
                                        System.out.println("Datababse error:");
                                        e.printStackTrace();
                                }
                                catch (IOException e)
                                {
                                        System.out.println("File IO error:");
                                        e.printStackTrace();
                                }
                        }
                        break;

        case "17":
                        //Export data from CUSTOMER table
                        //Capture file name to store the output
                        System.out.println("Please enter the file name:");
                        sc.nextLine();
                        final String exportFile = sc.nextLine(); // Read in user input - file
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(QUERY_17))
                                {
```

```
                                        //Writer object to write the records to database
                                        BufferedWriter writer = new
BufferedWriter(new FileWriter(exportFile));

                                        // Writing header for the file
                                        writer.write("cust_name,cust_address");
                                        System.out.println("Dispatching the query...");

                                        //Execute query and get data from database
                                        ResultSet result = statement.executeQuery();

                                        //Loop to write every record to file
                                        while(result.next())
                                        {
                                                //Extract fields from record to put in
place
                                                String export_name =
result.getString("cust_name");
                                                String export_address =
result.getString("cust_address");
                                                String record =
String.format("\"%s\",%s",export_name, export_address);

                                                writer.newLine();
                                                //Write record to file
                                                writer.write(record);
                                        }
                                        //Closing the writer object
                                        writer.close();
                                        System.out.println("File Export Successful!");
                                }
                                catch (SQLException e)
                                {
                                        System.out.println("Datababse error:");
                                        e.printStackTrace();
                                }
                                catch (IOException e)
                                {
                                        System.out.println("File IO error:");
                                        e.printStackTrace();
                                }
                        }
                        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
   }
}
```

```
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 associate with the person who made the product, repaired the product if it is repaired, or che
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;
10) Retrieve the total costs of the products in the product3 category which were repaired at the request of a particu
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 t
18) Quit(exit the program)!
18
Exiting! Goodbye!
```

Task 6. Java program Execution

6.1. Screenshots showing the testing of query 1

```
1
Please enter employee name:
Zayn
Please enter employee address:
Mesa
Please enter employee salary:
70000
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
Is employee technical staff?
Enter 1 for Yes 2 for No:
1
Please enter technical position :
IT
Does he/she has a BS degree?
Enter 1 for Yes 2 for No:
1
Does he/she has a MS degree?
Enter 1 for Yes 2 for No:
1
Does he/she has a PhD degree?
Enter 1 for Yes 2 for No:
1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Is employee quality controller?
Enter 1 for Yes 2 for No:
1
Please enter product type as Product 1/Product 2/Product 3 :
Product 1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Is employee worker?
Enter 1 for Yes 2 for No:
1
Please enter  maximum number of products a worker can produce per day :
8
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Affected Tables:

	emp_name	emp_address	emp_sal
1	Adams	OKC	70000
2	Bethany	OKC	70000
3	Codd	Stillwater	60000
4	Daniels	Norman	65000
5	Gordon	Austin	80000
6	Smith	Austin	80000
7	Sovan	OKC	80000
8	Tyler	Dallas	75000
9	Will	Norman	65000
10	Zayn	Mesa	70000

	ts_name	degree
1	Adams	BS
2	Adams	MS
3	Codd	BS
4	Codd	PhD
5	Daniels	BS
6	Zayn	BS
7	Zayn	MS
8	Zayn	PhD

	ts_name	tech_position
1	Adams	Assembler
2	Codd	IT
3	Daniels	Welder
4	Zayn	IT

	qc_name	product_type
1	Bethany	Product 3
2	Daniels	Product 1
3	Gordon	Product 3
4	Smith	Product 2
5	Sovan	Product 2
6	Will	Product 3
7	Zayn	Product 1

	worker_name	max_produce
1	Adams	8
2	Smith	8
3	Tyler	9
4	Will	8
5	Zayn	8

6.2 Screenshots showing the testing of query 2

```
Please enter product ID:
133
Please enter production date(YYYY-MM-DD):
2008-11-11
Please enter time spent(hh:mm:ss) to make the product:
12:00:00
Please enter employee who produced the product:
Tyler
Please enter employee who tested the product:
Zayn
Is the product repaired?
Enter 1 for Yes 2 for No
2
Connecting to the database...
Dispatching the query...
Done. QUERY RESULT : 1
Is product of type 1/ type 2/ type 3?
Enter 1 for type 1, 2 for type 2, 3 for type 3:
1
Please enter size of the product:
155
Please enter major software used:
R
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Affected Tables:

	product_ID	prod_date	time_taken	worker_name	qc_name	ts_name
1	101	2008-11-11	19:30:10	Smith	Daniels	*NULL*
2	102	2008-11-11	10:30:10	Tyler	Bethany	*NULL*
3	103	2008-11-11	10:30:10	Will	Smith	Daniels
4	104	2008-12-11	08:30:00	Adams	Gordon	*NULL*
5	105	2009-02-11	07:30:10	Adams	Gordon	Codd
6	106	2009-10-01	02:30:00	Adams	Bethany	*NULL*
7	107	2009-08-01	02:30:00	Smith	Gordon	*NULL*
8	108	2009-08-01	05:30:00	Tyler	Gordon	*NULL*
9	109	2010-02-11	05:30:10	Will	Gordon	Adams
10	110	2010-08-11	23:30:10	Adams	Smith	Daniels
11	123	2022-02-08	00:30:00	Will	Bethany	Codd
12	125	2022-08-09	01:00:00	Will	Will	*NULL*
13	127	2008-10-10	01:00:00	Adams	Daniels	*NULL*
14	128	2022-11-11	23:00:00	Adams	Daniels	Codd
15	129	2022-11-11	01:00:00	Will	Will	*NULL*
16	131	2009-11-11	01:00:00	Adams	Daniels	Codd
17	132	2008-11-11	01:00:00	Adams	Daniels	*NULL*

	product_ID	size	major_software
1	101	100	Python
2	102	100	Python
3	123	100	W
4	129	100	Q
5	133	155	R

	product_ID	size	color
1	103	110	Black
2	104	120	Black
3	105	100	Red
4	110	120	Red
5	132	100	Black
6	150	10	Red

	product_ID	size	prod_weight
1	106	110	100
2	107	120	125
3	108	100	100
4	109	100	125
5	131	100	23

	product_ID	ts_name	repair_dt
1	103	Daniels	2009-05-31
2	105	Codd	2009-03-11
3	109	Adams	2010-03-11
4	110	Daniels	2011-09-13
5	131	Codd	2009-01-01
6	150	Codd	2022-01-12

	product_ID	worker_name
1	104	Adams
2	105	Adams
3	106	Adams
4	110	Adams
5	127	Adams
6	131	Adams
7	132	Adams
8	150	Adams
9	101	Smith
10	107	Smith
11	102	Tyler
12	108	Tyler
13	133	Tyler
14	103	Will
15	109	Will
16	123	Will
17	129	Will

	product_ID	qc_name
1	102	Bethany
2	106	Bethany
3	101	Daniels
4	127	Daniels
5	131	Daniels
6	132	Daniels
7	150	Daniels
8	104	Gordon
9	105	Gordon
10	107	Gordon
11	108	Gordon
12	109	Gordon
13	103	Smith
14	110	Smith
15	129	Will
16	133	Zayn

	product_ID	qc_name	ts_name
1	105	Gordon	Codd
2	109	Gordon	Adams
3	150	Daniels	Codd

6.3 Screenshots showing the testing of query 3

```
3
Please enter customer name:
Sandhya
Please enter customer address:
Denver
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Enter number of products customer brought:
2
Please enter the product ID
123
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Please enter the product ID
125
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Affected Tables:

	cust_name	cust_address
1	Black	Austin
2	Jack	OKC
3	Jay	Michigan
4	Johnson	Dallas
5	Lynn	Dallas
6	Robert	OKC
7	Sam	Mesa
8	Sandhya	Denver
9	Viv	OKC
10	Zach	OKC

	product_ID	cust_name
1	101	Jack
2	102	Viv
3	103	Black
4	104	Viv
5	106	Lynn
6	109	Jack
7	110	Robert
8	123	Sandhya
9	125	Sandhya
10	132	Jay

6.4 Screenshots showing the testing of query 4

```
4
Please enter account no:
12
Please enter the date the account established:
2020-11-11
Please enter the cost of the product:
650
Please enter the product ID associated with this account:
132
Please enter the account type:
Enter 1 for product1-account, 2 for product2-account, 3 for product3-account
2
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

Affected Tables:

	account_no	est_date
1	1	2009-01-01
2	2	2009-01-01
3	3	2009-01-01
4	4	2009-01-01
5	5	2010-01-01
6	6	2010-01-01
7	7	2010-01-01
8	8	2010-01-01
9	9	2010-12-01
10	10	2010-12-01
11	12	2020-11-11
12	111	2021-01-01

	account_no	cost_prod1
1	1	500
2	2	500
3	111	300

	account_no	cost_prod2
1	3	600
2	4	600
3	5	600
4	10	600
5	12	650

	account_no	cost_prod3
1	6	650
2	7	650
3	8	650
4	9	650

	product_ID	account_no
1	101	1
2	102	2
3	103	3
4	104	4
5	105	5
6	106	6
7	107	7
8	108	8
9	109	9
10	110	10
11	123	111
12	132	12

6.5 Screenshots showing the testing of query 5

```
2020-09-09
Please enter complaint description:
size defect
Please enter treatment expected:
Enter 1 for Refund or 2 for Exchange
2
Please enter the product ID for which the complaint is raise:
125
Please enter the customer name who raised the complaint:
Sandhya
Please enter the technician name who will work on this complaint
Codd
Connecting to the database...
Dispatching the query...
```

Affected Tables:

	complaint_ID	date_of_complaint	complaint_desc	treatment
1	1	2009-05-01	Size Defect	Refund
2	2	2011-02-01	Color fade	Exchange
3	3	2011-01-03	Size Defect	Refund
4	4	2021-09-09	Size defect	Refund
5	5	2020-11-12	Size defect	Exchange

	complaint_ID	product_ID	cust_name
1	1	103	Black
2	2	110	Robert
3	3	109	Jack
4	4	104	Viv
5	5	105	Sandhya

	complaint_ID	product_ID	ts_name
1	1	103	Daniels
2	2	110	Daniels
3	3	109	Adams
4	4	104	Adams
5	5	105	Codd

6.6 Screenshots showing the testing of query 6

```
6
Please enter accident number:
4
Please enter the date of accident:
2009-11-11
Please enter number of work days lost due to the accident:
67
Please enter product ID with which the accident is associa
101
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
Please enter reason of accident:
1 for Repair or 2 Produce:
2
Please enter the worker involved in the accident:
Smith
Connecting to the database...
Dispatching the query...
```

Affected Tables:

	accident_no	accident_dt	days_lost
1	1	2009-06-04	6
2	2	2011-09-15	10
3	3	2022-11-11	4
4	4	2009-11-11	67

	accident_no	ts_name	product_ID
1	1	Daniels	103
2	2	Daniels	110

	accident_no	worker_name	product_ID
1	3	Will	103
2	4	Smith	101

6.7 Screenshots showing the testing of query 7

```
7
Please enter product ID:
101
Connecting to the database...
Contents of the Product table:
date produced | time spent to produce
2008-11-11 | 19:30:10.0000000
```

```
7
Please enter product ID:
132
Connecting to the database...
Contents of the Product table:
date produced | time spent to produce
2008-11-11 | 01:00:00.0000000
```

```
7
Please enter product ID:
110
Connecting to the database...
Contents of the Product table:
date produced | time spent to produce
2010-08-11 | 23:30:10.0000000
```

6.8 Screenshots showing the testing of query 8

```
8
Please enter worker name:
Adams
Connecting to the database...
Contents of the Product table:
product ID | date produced | time spent | worker | quality controller | technical staff (if any)
104 | 2008-12-11 | 08:30:00.0000000 | Adams | Gordon | null
105 | 2009-02-11 | 07:30:10.0000000 | Adams | Gordon | Codd
106 | 2009-10-01 | 02:30:00.0000000 | Adams | Bethany | null
110 | 2010-08-11 | 23:30:10.0000000 | Adams | Smith | Daniels
127 | 2008-10-10 | 01:00:00.0000000 | Adams | Daniels | null
128 | 2022-11-11 | 23:00:00.0000000 | Adams | Daniels | Codd
131 | 2009-11-11 | 01:00:00.0000000 | Adams | Daniels | Codd
132 | 2008-11-11 | 01:00:00.0000000 | Adams | Daniels | null
150 | 2022-01-03 | 03:00:00.0000000 | Adams | Daniels | Codd
```

```
8
Please enter worker name:
Tyler
Connecting to the database...
Contents of the Product table:
product ID | date produced | time spent | worker | quality controller | technical staff (if any)
102 | 2008-11-11 | 10:30:10.0000000 | Tyler | Bethany | null
108 | 2009-08-01 | 05:30:00.0000000 | Tyler | Gordon | null
133 | 2008-11-11 | 12:00:00.0000000 | Tyler | Zayn | null
```

```
8
Please enter worker name:
Will
Connecting to the database...
Contents of the Product table:
product ID | date produced | time spent | worker | quality controller | technical staff (if any)
103 | 2008-11-11 | 10:30:10.0000000 | Will | Smith | Daniels
109 | 2010-02-11 | 05:30:10.0000000 | Will | Gordon | Adams
123 | 2022-02-08 | 00:30:00.0000000 | Will | Bethany | Codd
125 | 2022-08-09 | 01:00:00.0000000 | Will | Will | null
129 | 2022-11-11 | 01:00:00.0000000 | Will | Will | null
```

6.9 Screenshots showing the testing of query 9

```
9
Please enter quality controller name:
Daniels
Connecting to the database...
total no.of errors
0
```

```
9
Please enter quality controller name:
Gordon
Connecting to the database...
total no.of errors
3
```

```
9
Please enter quality controller name:
Smith
Connecting to the database...
total no.of errors
2
```

6.10 Screenshots showing the testing of query 10

```
10
Please enter quality controller name:
Gordon
Connecting to the database...
total cost
650.0
```

```
10
Please enter quality controller name:
Smith
Connecting to the database...
total cost
null
```

```
10
Please enter quality controller name:
Daniels
Connecting to the database...
total cost
null
```

6.11 Screenshots showing the testing of query 11

```
11
Please enter product color:
Red
Connecting to the database...
customer name | address
Robert | OKC
```

```
11
Please enter product color:
Black
Connecting to the database...
customer name | address
Black | Austin
Jay | Michigan
Viv | OKC
```

6.12 Screenshots showing the testing of query 12

```
12
Please enter salary:
65000
Connecting to the database...
employee name | address | salary
Adams | OKC | 70000.0
Bethany | OKC | 70000.0
Gordon | Austin | 80000.0
Smith | Austin | 80000.0
Sovan | OKC | 80000.0
Tyler | Dallas | 75000.0
Zayn | Mesa | 70000.0
```

6.13 Screenshots showing the testing of query 13

```
13
Connecting to the database...
total number of work days lost
16
```

6.14 Screenshots showing the testing of query 14

```
14
Please enter production year:
2008
Connecting to the database...
average cost
570.0
```

6.15 Screenshots showing the testing of query 15

```
15
Please enter start date:
2022-01-01
Please enter end date:
2022-11-12
Connecting to the database...
Dispatching the query...
Done. QUERY RESULT :0
Connecting to the database...
Dispatching the query...
Done. QUERY RESULT :1
Connecting to the database...
Dispatching the query...
Done. QUERY RESULT :1
```

6.16 Screenshots showing the testing of query 16

```
16
Please enter the file name:
import.csv
Joshua,Waco,10000
Kathy,Arizona,90000
Sierra,Chicago,90000
3 records have been inserted successfully!
```

6.17 Screenshots showing the testing of query 17

```
17
Please enter the file name:
exportfile.csv
Connecting to the database...
Dispatching the query...
File Export Successful!
```

6.18 Screenshots showing the testing of query 18

```
18) Quit(exit the program)!
17
Please enter the file name:
exportfile.csv
Connecting to the database...
Dispatching the query...
File Export Successful!

Please select one of the options below:
1) Enter a new employee;
2) Enter a new product associate with the person who made the product, repaired the product if it is repaired, or
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;
10) Retrieve the total costs of the products in the product3 category which were repaired at the request of a parti
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 ther
18) Quit(exit the program)!
18
Exiting! Goodbye!
```

6.19 Error Checks for Query

- Tried to insert a duplicate *employee name* into the Employee table, it threw a PK violation error

```
1
Please enter employee name:
Vis
Please enter employee address:
OKC
Please enter employee salary:
65000
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK_
```

- Since *account_no* attribute of Account table is an integer, when tried to insert character into it, threw a domain violation error

```
4
Please enter account no:
ABC
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at Mohanty_Sonaxy_IP_Task5b.main(Mohanty_Sonaxy_IP_Task5b.java:660)
```

- Accident is the main table for two other tables – Repair_Accident and Prod_Accident. Tried to delete an accident from the Accident table without deleting from its children tables, and the database threw an error

	accident_no	accident_dt	days_lost
1	1	2009-06-04	6
2	2	2011-09-15	10

	accident_no	ts_name	product_ID
1	1	Daniels	103
2	2	Daniels	110

Task 7. Web database application and its execution

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

DataHandler.java

```
package IP_jsp;

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

public class DataHandler {
        private Connection conn;

   // Azure SQL connection credentials
   private String server = "xxxx";
   private String database = " xxxx ";
   private String username = " xxxx ";
   private String password = " xxxx";

   // Resulting connection string
   final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustS
erverCertificate=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);
   }

   // Return the result of selecting everything from the employee table where salary
```

```
    //condition is satisfied
    public ResultSet getAllEmployees(int emp_sal) throws SQLException
    {
        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery = "SELECT * FROM Employee WHERE emp_sal > ?";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace ?s in the query with user inputs
        stmt.setInt(1, emp_sal);


        return stmt.executeQuery();
    }

    // Inserts a record into the employee table with the given attribute values
    public boolean addEmployee(String emp_name, String emp_address, int emp_sal) throws
SQLException
    {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery = "INSERT INTO Employee (emp_name, emp_address, emp_sal) VALUES
(?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, emp_name);
        stmt.setString(2, emp_address);
        stmt.setInt(3, emp_sal);

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

}
```

getAllEmployeesForm.jsp

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Employee Details</title>
    </head>
    <body>
        <h2>Employee Details</h2>
```

```
        <!--
           Form for collecting user input for the employee table.
           Upon form submission, getAllEmployees.jsp file will be invoked.
        -->
        <form action="getAllEmployees.jsp">
           <!-- The form organized in an HTML table for better clarity. -->
           <table border=1>
              <tr>
                 <th colspan="2">Enter the Employee Salary:</th>
              </tr>
              <tr>
                 <td>Salary:</td>
                 <td><div style="text-align: center;">
                 <input type=text name=emp_sal>
                 </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=Search>
                 </div></td>
              </tr>
           </table>
        </form>
     </body>
  </html>
```

getAllEmployees.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="IP_jsp.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 from the user
```

```
String salaryString = request.getParameter("emp_sal");

int emp_sal = Integer.parseInt(salaryString);
        final ResultSet Employee = handler.getAllEmployees(emp_sal);
        %>
        <!-- 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>
                        <td align="center">
                                <h4>Address</h4>
                        </td>
                        <td align="center">
                                <h4>Salary</h4>
                        </td>
                </tr>
                <%
         while(Employee.next()) { // For each employee record returned...
              // Extract the attribute values for every row returned
              final String name = Employee.getString("emp_name");
              final String address = Employee.getString("emp_address");
              final String salary = Employee.getString("emp_sal");

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

addEmployeeForm.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Add Employee</title>
  </head>
  <body>
    <h2>Add Employee</h2>
```

```
<!--
   Form for collecting user input for the new employee record.
   Upon form submission, addEmployee.jsp file will be invoked.
-->
<form action="addEmployee.jsp">
   <!-- The form organized in an HTML table for better clarity. -->
   <table border=1>
      <tr>
         <th colspan="2">Enter the Employee Data:</th>
      </tr>
      <tr>
         <td>Employee Name:</td>
         <td><div style="text-align: center;">
         <input type=text name=emp_name>
         </div></td>
      </tr>
      <tr>
         <td>Address:</td>
         <td><div style="text-align: center;">
         <input type=text name=emp_address>
         </div></td>
      </tr>
      <tr>
         <td>Salary:</td>
         <td><div style="text-align: center;">
         <input type=text name=emp_sal>
         </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>
```

addEmployee.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="IP_jsp.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 emp_name = request.getParameter("emp_name");
  String emp_address = request.getParameter("emp_address");
  String salaryString = request.getParameter("emp_sal");


  /*
   * If the user hasn't filled out all the attributes, form will again be invoked
   */
  if (emp_name.equals("") || emp_address.equals("") || salaryString.equals("")) {
    response.sendRedirect("addEmployeeForm.jsp");
  } else {
    int emp_sal = Integer.parseInt(salaryString);

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

      <ul>
        <li>Name: <%=emp_name%></li>
        <li>Address: <%=emp_address%></li>
        <li>Salary: <%=salaryString%></li>

      </ul>

      <h2>Was successfully inserted.</h2>

      <a href="getAllEmployees.jsp">See all employees.</a>
      <%
    }
  }
```

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

7.2 Screenshots showing the testing of the Web database application

- first query 12 is executed:

Employee Details

Enter the Employee Salary:	
Salary:	65000
Clear	Search

Employee Name	Address	Salary
Adams	OKC	70000
Bethany	OKC	70000
Gordon	Austin	80000
Smith	Austin	80000
Tyler	Dallas	75000

- then query 1 is executed:

Add Employee

Enter the Employee Data:	
Employee Name:	Zayn
Address:	Mesa
Salary:	70000
Clear	Insert

Employee:

- Name: Zayn
- Address: Mesa
- Salary: 70000

Was successfully inserted.

- then again query 12 is executed

Employee Details

Enter the Employee Salary:	
Salary:	65000
Clear	Search

Employee Name	Address	Salary
Adams	OKC	70000
Bethany	OKC	70000
Gordon	Austin	80000
Smith	Austin	80000
Tyler	Dallas	75000
Zayn	Mesa	70000