Assignment: Individual Project

Course Name: Database Management Systems

Course Number: CS/DSA 4513

Section Number: 001

Semester and Year: Fall 2023

Instructor's Name: Dr. Le Gruenwald

Author's Name: Kaustubh Pande

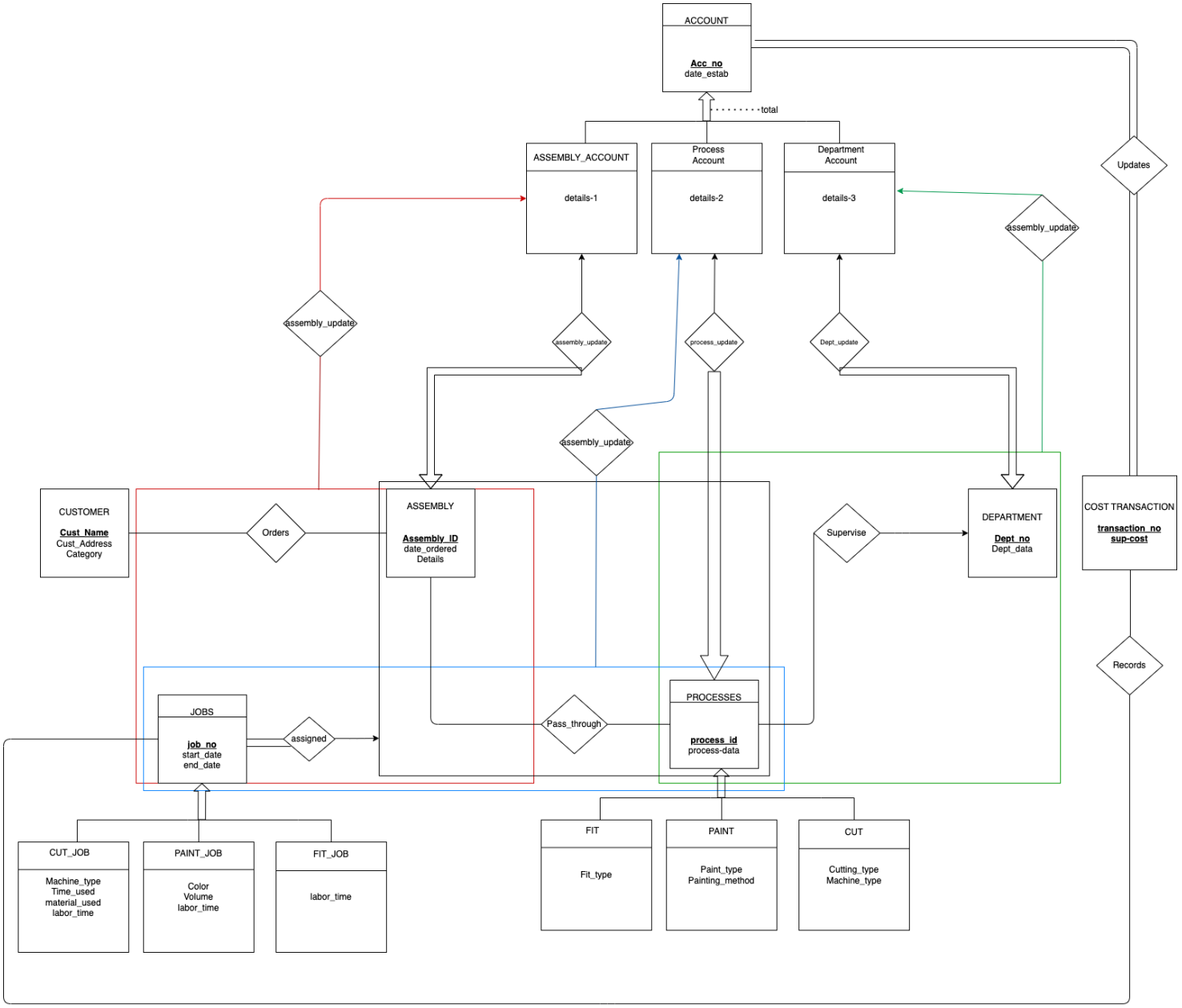
ID: 113527525

Email Address: kaustubhpande@ou.edu

Title: Job Shop Accounting System

Table of Contents:

TASK 1: ER Diagram



TASK 2:

Relational Schema:

Customer(**customer_name**, customer_address, category)

Assembly(**assembly_id**, date_ordered, details)

Process(**process_id**, process_data)

Cut_process(**process_id**, cutting_type, machine_type)

Paint_process(**process_id**, paint_type, painting_method)

Fit_process(**process_id**, fit_type)

Department(**department_number**, department_data)

Job(**job_number**, start_date, end_date)

Cut_job(**job_number**, machine_type, time_used, material_used, labor_time)

Paint_job(**job_number**, color, volume, labor_time)

Fit_job(**job_number**, labor_time)

Account(**account_number**, date_established)

Assembly_account(**account_number**, details-1)

Process_account(**account_number**, details-2)

Department_account(**account_number**, details-3)

Cost_transaction(**transaction_number**, sup-cost)

Process_update(**Acc_no**, **process_id, job_no**)

Assembly_update(**Acc_no**, **assembly_id, job_number**)

Department_update(**Acc_no, department_number,process_id**)

pass_through(**process_id, assembly_id**)

supervise(**process_id, department_number**)

assigned(**process_id, assembly_id, job_number**)

records(**job_number, transaction_number**)

updates(**Acc_no**, **transaction_number)**

TASK 3: Discussion
3.1

Table Name	Query # and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	1 (Insertion), 12 (Range Search)	Customer Name, Category	30/day, 100/day	B+ Tree Index	I chose a B+ Tree Index because we frequently need to search for customers by category, and B+ Trees are great for range searches. Plus, we're inserting new customers daily, so the tree structure helps to keep insertion efficient too.
Department	2 (Insertion)	Department Number	Infrequent	Heap File	Since we don't add new departments often, a simple Heap File is good enough.
Assembly	4 (Insertion), 9 (Random Search)	Assembly ID	40/day, 200/day	Clustered Index on Assembly ID	I went with a Clustered Index on Assembly ID because we do a lot of insertions and searches by Assembly ID. This way, the data is stored in order of the primary key, making searches fast.
Process	3 (Insertion)	Process ID	Infrequent	Heap File	Infrequent insertion, so a heap file is adequate.
Job	6 (Insertion), 7 (Update), 10 (Random Search)	Job Number	50/day, 20/day	Hashing on Job Number	I chose Hashing on Job Number because we insert and update jobs a lot, and hashing is super quick for these operations. It also makes finding a job by its number really efficient.

Account	5 (Insertion)	Account Number	10/day	Heap File	Since we're mainly adding new accounts and not doing much else, a Heap File should do the trick. It's straightforward and works well with our moderate insertion rate.
Cost_transaction	8 (Insertion)	Transaction Number	50/day	Heap File or Hashing	We're mostly adding new transactions, and neither method requires complex searches, so it's more about keeping it simple.
Assembly_account, Process_account, Department_account	5 (Insertion)	Account Number	10/day	Heap File	Like the Account table, these tables are mostly for insertions. A Heap File is uncomplicated and serves the purpose without the need for advanced search capabilities.
Cut_process, Paint_process, Fit_process	3 (Insertion)	Process ID	Infrequent	Heap File	A Heap File is sufficient and keeps things simple, which is perfect for our infrequent usage.
Department_update, Assembly_update, Process_update	8 (Insertion)	Transaction Number, Account Number	50/day	Heap File or Hashing	These tables are updated regularly with new transaction records. A Heap File could work, but if we start needing to search by transaction number, Hashing might be a better call for faster access.

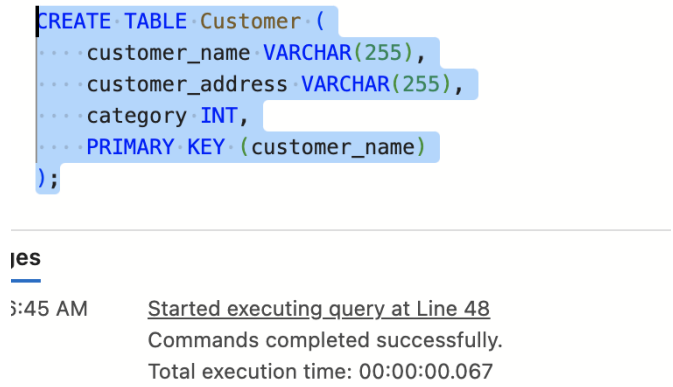
3.2

Azure SQL Databases offer a suite of index types, each leveraging various table attributes to optimize performance. These include Clustered and Non-Clustered indexes, which organize data based on primary key and non-key columns, respectively. Hash indexes are optimized for performance in memory-optimized tables. Unique indexes prevent duplicate entries in columns. Columnstore indexes are designed for high-performance queries on large datasets. Filtered indexes are beneficial for queries on well-defined subsets of data. Spatial indexes are used for queries on spatial data, and XML indexes are intended for XML data operations.

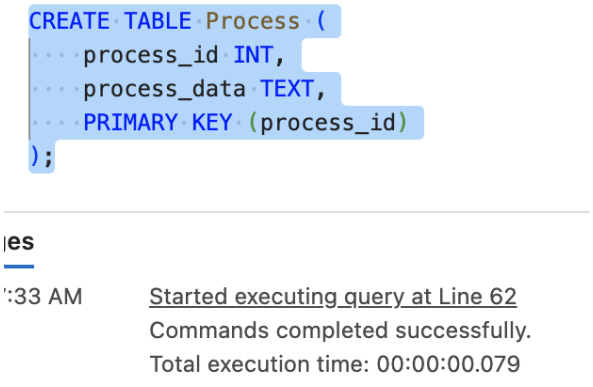
While this project implementation often necessitates the use of an Extendable Hashing Structure for tables to efficiently handle growing data, for the scope of this project, I am considering primary indexes as a standard approach due to their general applicability and ease of use.

Task 4: SQL statements and screenshots showing the creation of tables in Azure SQL Database

```
CREATE TABLE Customer (
    customer_name VARCHAR(255),
    customer_address VARCHAR(255),
    category INT,
    PRIMARY KEY (customer_name)
);
```




```
CREATE TABLE Assembly (
    assembly_id INT,
    date_ordered DATE,
    details TEXT,
    PRIMARY KEY (assembly_id)
);
```




```
CREATE TABLE Department (
    department_number INT,
    department_data TEXT,
    PRIMARY KEY (department_number)
);
```

```
CREATE TABLE Department (
    department_number INT,
    department_data TEXT,
    PRIMARY KEY (department_number)
);
```

```
s

14 AM    Started executing query at Line 68
         Commands completed successfully.
         Total execution time: 00:00:00.074
```

```
CREATE TABLE Job (
    job_number INT,
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (job_number)
);
```

```
74  CREATE TABLE Job (
75      job_number INT,
76      start_date DATE,
77      end_date DATE,
78      PRIMARY KEY (job_number)
79  );
```

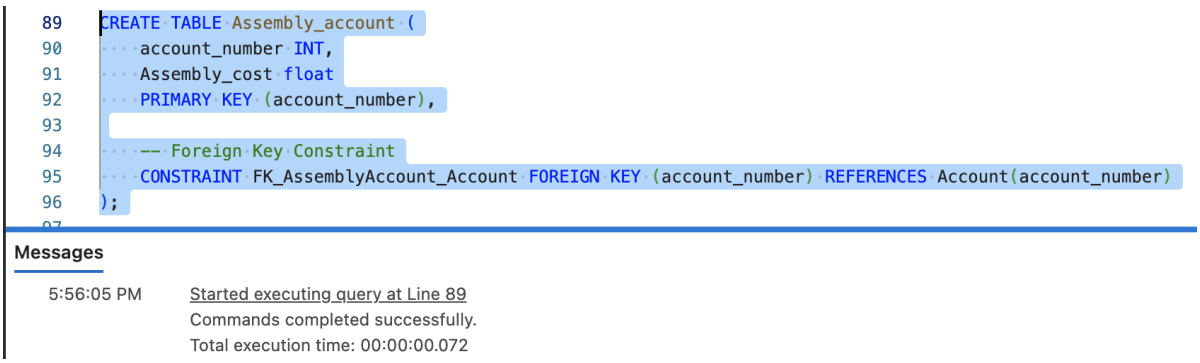
```
Messages

7:09:12 AM    Started executing query at Line 74
              Commands completed successfully.
              Total execution time: 00:00:00.070
```

```
CREATE TABLE Account (
    account_number INT,
    date_established DATE,
    PRIMARY KEY (account_number)
);
```

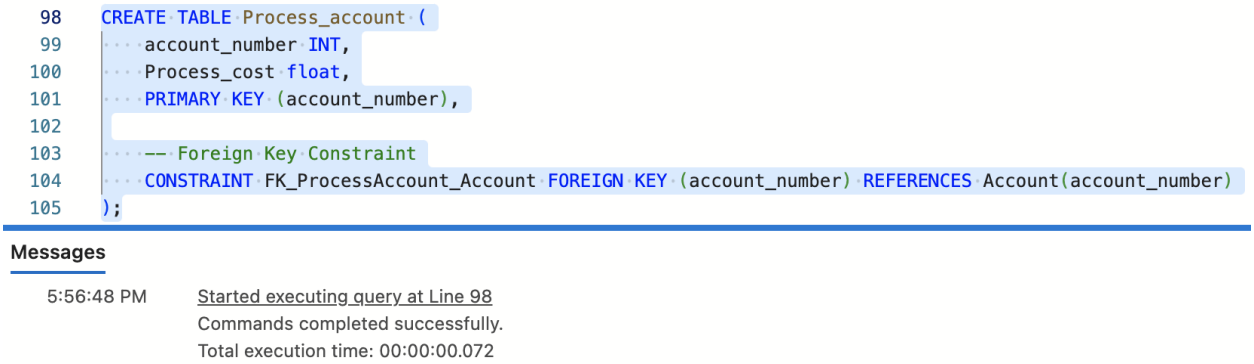
```
81  CREATE TABLE Account (
82      account_number INT,
83      date_established DATE,
84      PRIMARY KEY (account_number)
85  );
```

```
lessages

7:10:05 AM    Started executing query at Line 81
              Commands completed successfully.
              Total execution time: 00:00:00.072
```

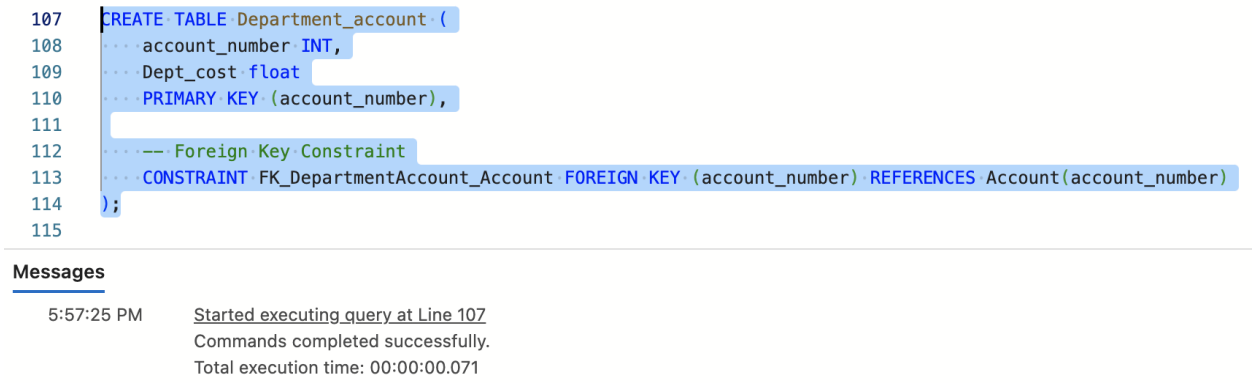
```
CREATE TABLE Assembly_account (
   account_number INT,
   Assembly_cost float
   PRIMARY KEY (account_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_AssemblyAccount_Account FOREIGN KEY (account_number) REFERENCES
Account(account_number)
);
```



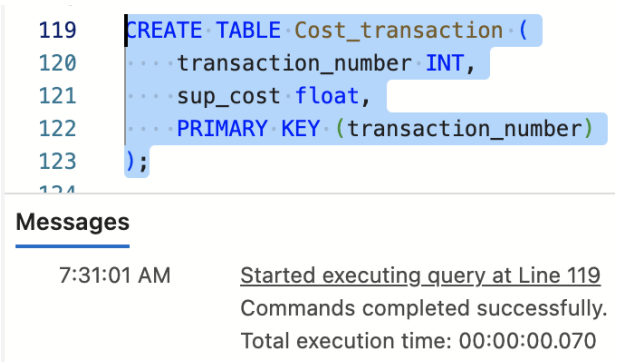

```
CREATE TABLE Process_account (
   account_number INT,
   Process_cost float,
   PRIMARY KEY (account_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_ProcessAccount_Account FOREIGN KEY (account_number) REFERENCES
Account(account_number)
);
```



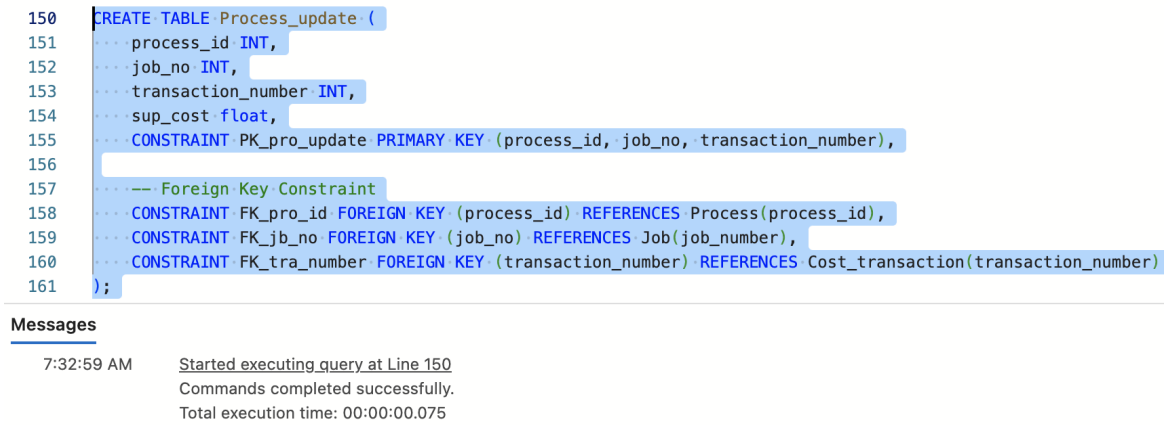
```sql
CREATE TABLE Department_account (
    account_number INT,
    Dept_cost float
    PRIMARY KEY (account_number),

    -- Foreign Key Constraint
    CONSTRAINT FK_DepartmentAccount_Account FOREIGN KEY (account_number) REFERENCES
Account(account_number)
);
```




```sql
CREATE TABLE Cost_transaction (
    transaction_number INT,
    sup_cost float,
    PRIMARY KEY (transaction_number)
);
```



```sql
CREATE TABLE Process_update (
   process_id INT,
   job_no INT,
   transaction_number INT,
   sup_cost float,
   CONSTRAINT PK_pro_update PRIMARY KEY (process_id, job_no, transaction_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_pro_id FOREIGN KEY (process_id) REFERENCES Process(process_id),
   CONSTRAINT FK_jb_no FOREIGN KEY (job_no) REFERENCES Job(job_number),
   CONSTRAINT FK_tra_number FOREIGN KEY (transaction_number) REFERENCES
Cost_transaction(transaction_number)
);
```




```sql
CREATE TABLE Process_update (
   account_number INT,
   process_id INT,
   job_number INT,
   CONSTRAINT FK_Pro_acc FOREIGN KEY (account_number) REFERENCES Account(account_number),
   CONSTRAINT FK_Pro_id FOREIGN KEY (process_id) REFERENCES Process(process_id),
   CONSTRAINT FK_Pro_jn FOREIGN KEY (job_number) REFERENCES Job(job_number)
);

CREATE TABLE Assembly_update (
   account_number INT,
   assembly_id INT,
   job_number INT,
   CONSTRAINT FK_Pro_assemb FOREIGN KEY (account_number) REFERENCES Account(account_number),
```

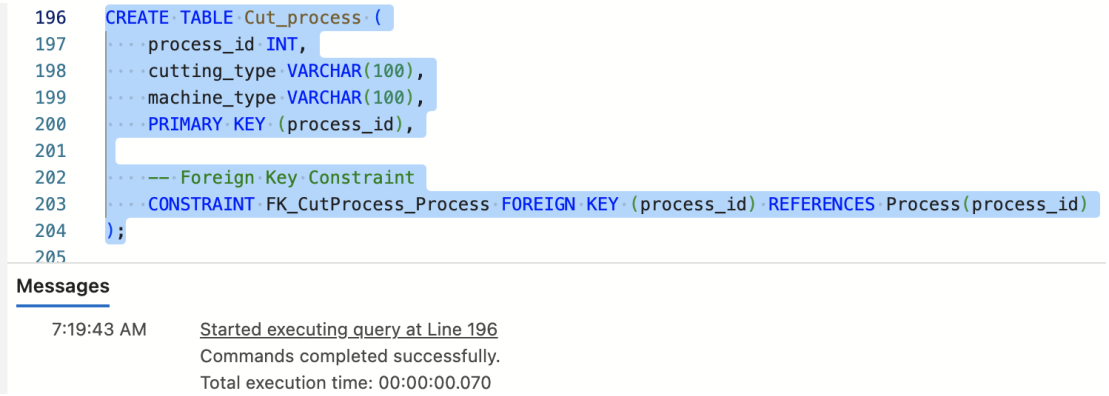
```
    CONSTRAINT FK_Pro_asid FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id),
    CONSTRAINT FK_Pro_acjn FOREIGN KEY (job_number) REFERENCES Job(job_number)
);

CREATE TABLE Department_update (
    account_number INT,
    department_number INT,
    process_id INT,
    FOREIGN KEY (account_number) REFERENCES Account(account_number),
    FOREIGN KEY (department_number) REFERENCES Department(department_number),
    FOREIGN KEY (process_id) REFERENCES Process(process_id)
);
```

```
122  CREATE TABLE Process_update (
123      account_number INT,
124      process_id INT,
125      job_number INT,
126      CONSTRAINT FK_Pro_acc FOREIGN KEY (account_number) REFERENCES Account(account_number),
127      CONSTRAINT FK_Pro_id FOREIGN KEY (process_id) REFERENCES Process(process_id),
128      CONSTRAINT FK_Pro_jn FOREIGN KEY (job_number) REFERENCES Job(job_number)
129  );
130
131  CREATE TABLE Assembly_update (
132      account_number INT,
133      assembly_id INT,
134      job_number INT,
135      CONSTRAINT FK_Pro_assemb FOREIGN KEY (account_number) REFERENCES Account(account_number),
136      CONSTRAINT FK_Pro_asid FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id),
137      CONSTRAINT FK_Pro_acjn FOREIGN KEY (job_number) REFERENCES Job(job_number)
138  );
139
140  CREATE TABLE Department_update (
141      account_number INT,
142      department_number INT,
143      process_id INT,
144      FOREIGN KEY (account_number) REFERENCES Account(account_number),
145      FOREIGN KEY (department_number) REFERENCES Department(department_number),
146      FOREIGN KEY (process_id) REFERENCES Process(process_id)
147  );
148
```

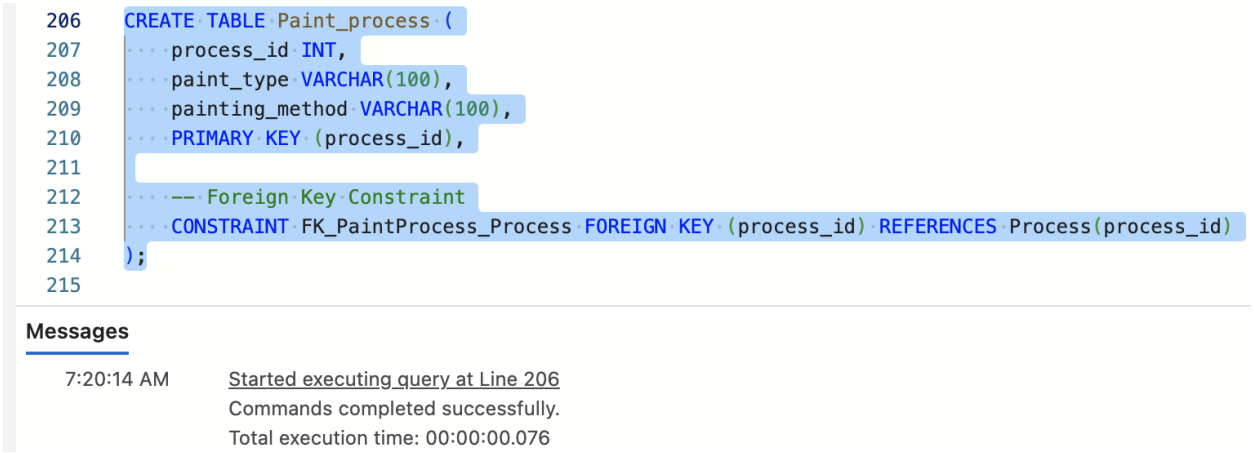
Messages

6:01:40 PM Started executing query at Line 122
Commands completed successfully.
Total execution time: 00:00:00.102

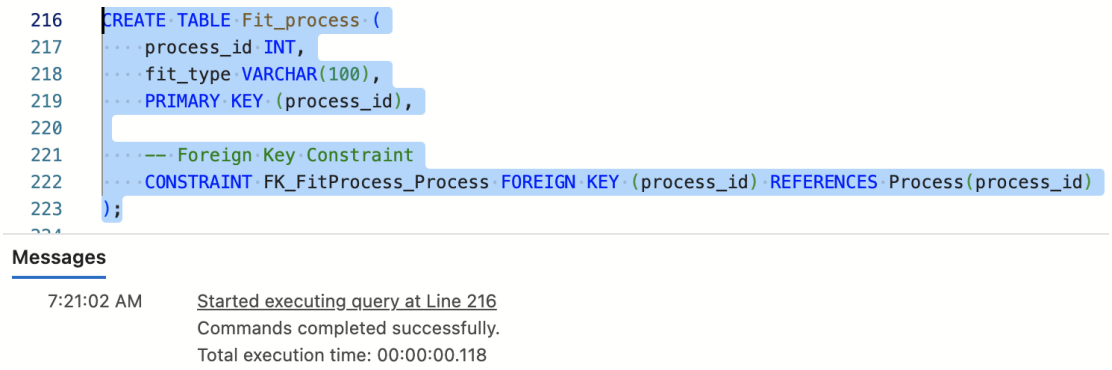
```sql
CREATE TABLE Cut_process (
    process_id INT,
    cutting_type VARCHAR(100),
    machine_type VARCHAR(100),
    PRIMARY KEY (process_id),

    -- Foreign Key Constraint
    CONSTRAINT FK_CutProcess_Process FOREIGN KEY (process_id) REFERENCES Process(process_id)
);
```



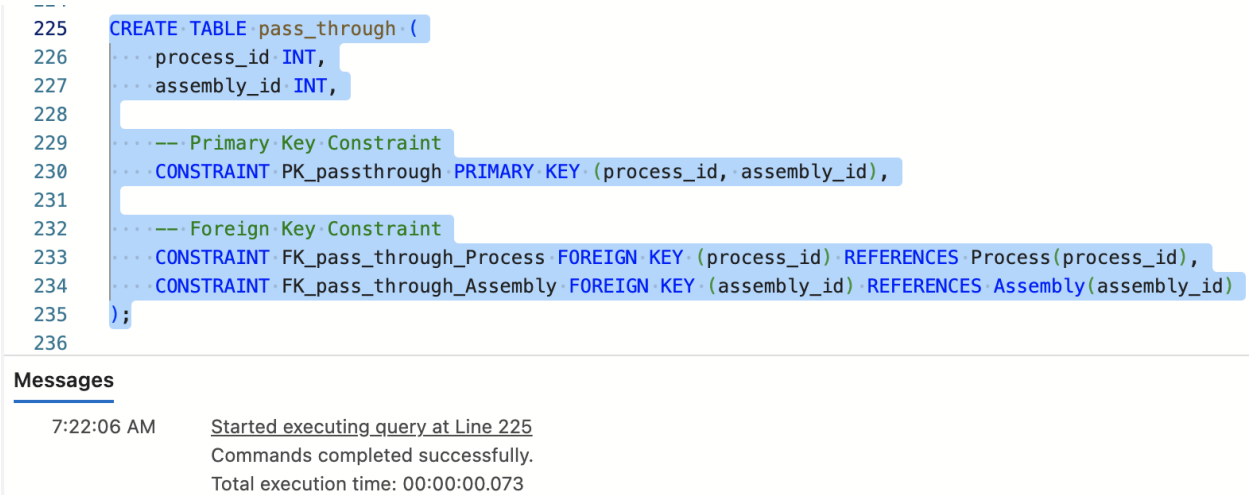

```sql
CREATE TABLE Paint_process (
    process_id INT,
    paint_type VARCHAR(100),
    painting_method VARCHAR(100),
    PRIMARY KEY (process_id),

    -- Foreign Key Constraint
    CONSTRAINT FK_PaintProcess_Process FOREIGN KEY (process_id) REFERENCES Process(process_id)
);
```



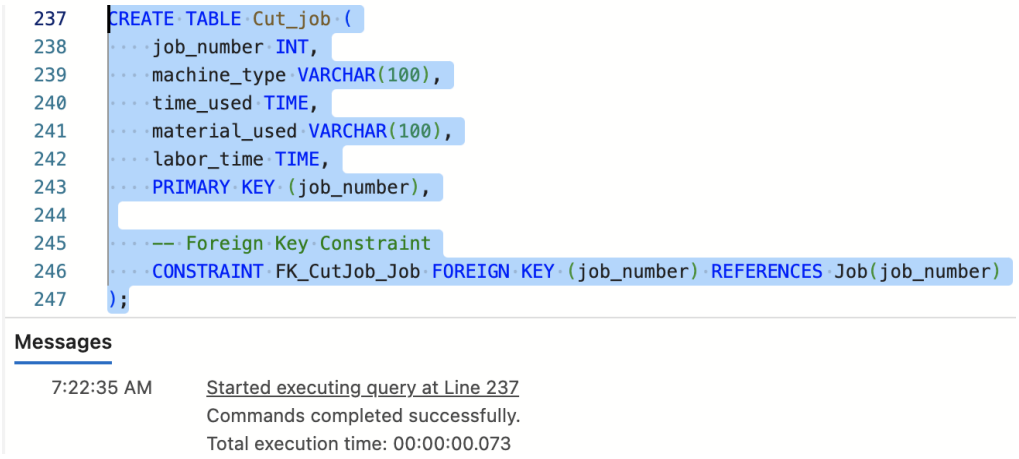
```sql
CREATE TABLE Fit_process (
   process_id INT,
   fit_type VARCHAR(100),
   PRIMARY KEY (process_id),

   -- Foreign Key Constraint
   CONSTRAINT FK_FitProcess_Process FOREIGN KEY (process_id) REFERENCES Process(process_id)
);
```



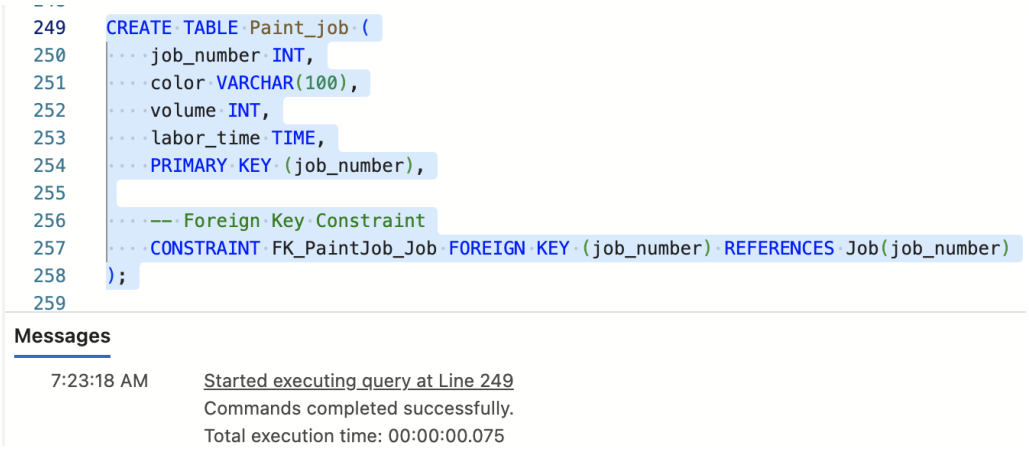

```sql
CREATE TABLE pass_through (
   process_id INT,
   assembly_id INT,

   -- Primary Key Constraint
   CONSTRAINT PK_passthrough PRIMARY KEY (process_id, assembly_id),

   -- Foreign Key Constraint
   CONSTRAINT FK_pass_through_Process FOREIGN KEY (process_id) REFERENCES
Process(process_id),
   CONSTRAINT FK_pass_through_Assembly FOREIGN KEY (assembly_id) REFERENCES
Assembly(assembly_id)
);
```



```sql
CREATE TABLE Cut_job (
    job_number INT,
    machine_type VARCHAR(100),
    time_used TIME,
    material_used VARCHAR(100),
    labor_time TIME,
    PRIMARY KEY (job_number),

    -- Foreign Key Constraint
    CONSTRAINT FK_CutJob_Job FOREIGN KEY (job_number) REFERENCES Job(job_number)
);
```




```sql
CREATE TABLE Paint_job (
    job_number INT,
    color VARCHAR(100),
    volume INT,
    labor_time TIME,
    PRIMARY KEY (job_number),

    -- Foreign Key Constraint
    CONSTRAINT FK_PaintJob_Job FOREIGN KEY (job_number) REFERENCES Job(job_number)
);
```



```
CREATE TABLE Fit_job (
   job_number INT,
   labor_time TIME,
   PRIMARY KEY (job_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_FitJob_Job FOREIGN KEY (job_number) REFERENCES Job(job_number)
);
```

```
260  CREATE TABLE Fit_job (
261      job_number INT,
262      labor_time TIME,
263      PRIMARY KEY (job_number),
264
265      -- Foreign Key Constraint
266      CONSTRAINT FK_FitJob_Job FOREIGN KEY (job_number) REFERENCES Job(job_number)
267  );
```

Messages

7:23:52 AM Started executing query at Line 260
Commands completed successfully.
Total execution time: 00:00:00.074

```
CREATE TABLE supervised (
   process_id INT,
   department_number INT,

   -- Primary Key Constraint
   CONSTRAINT PK_supervised PRIMARY KEY (process_id, department_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_supervise_Process FOREIGN KEY (process_id) REFERENCES Process(process_id),
   CONSTRAINT FK_supervise_Department FOREIGN KEY (department_number) REFERENCES
Department(department_number)
);
```

```
269  CREATE TABLE supervised (
270      process_id INT,
271      department_number INT,
272
273      -- Primary Key Constraint
274      CONSTRAINT PK_supervised PRIMARY KEY (process_id, department_number),
275
276      -- Foreign Key Constraint
277      CONSTRAINT FK_supervise_Process FOREIGN KEY (process_id) REFERENCES Process(process_id),
278      CONSTRAINT FK_supervise_Department FOREIGN KEY (department_number) REFERENCES Department(department_number)
279  );
280
```

Messages

7:24:31 AM Started executing query at Line 269
Commands completed successfully.
Total execution time: 00:00:00.072

```sql
CREATE TABLE assigned (
   process_id INT,
   assembly_id INT,
   job_number INT,

   -- Primary Key Constraint
   CONSTRAINT PK_assigned PRIMARY KEY (process_id, assembly_id, job_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_assigned_Process FOREIGN KEY (process_id) REFERENCES Process(process_id),
   CONSTRAINT FK_assigned_Assembly FOREIGN KEY (assembly_id) REFERENCES
Assembly(assembly_id),
   CONSTRAINT FK_assigned_Job FOREIGN KEY (job_number) REFERENCES Job(job_number)
);
```


```sql
CREATE TABLE records (
   job_number INT,
   transaction_number INT,

   -- Primary Key Constraint
   CONSTRAINT PK_recods PRIMARY KEY (job_number, transaction_number),

   -- Foreign Key Constraint
   CONSTRAINT FK_records_Job FOREIGN KEY (job_number) REFERENCES Job(job_number),
   CONSTRAINT FK_records_CostTransaction FOREIGN KEY (transaction_number) REFERENCES
Cost_transaction(transaction_number)
);
```

```sql
CREATE TABLE records (
    job_number INT,
    transaction_number INT,

    -- Primary Key Constraint
    CONSTRAINT PK_recods PRIMARY KEY (job_number, transaction_number),

    -- Foreign Key Constraint
    CONSTRAINT FK_records_Job FOREIGN KEY (job_number) REFERENCES Job(job_number),
    CONSTRAINT FK_records_CostTransaction FOREIGN KEY (transaction_number) REFERENCES Cost_transaction(transaction_number)
);
```

Messages

7:25:41 AM Started executing query at Line 295
Commands completed successfully.
Total execution time: 00:00:00.072

Task – 5

5.1 SQL statements (and Transact SQL stored procedures) - Implementing all queries

QUERY 1:

```
42  -- Query 1: Enter a new customer ------------------------------------
43  GO
44
45  CREATE PROCEDURE Q1
46      @customer_name VARCHAR(100),
47      @customer_address VARCHAR(100),
48      @category INT
49  AS
50  BEGIN
51      INSERT INTO Customer (customer_name, customer_address, category)
52      VALUES (@customer_name, @customer_address, @category);
53  END;
54  GO
55
```

Messages

```
7:50:13 AM   Started executing query at Line 42
             Commands completed successfully.
7:50:13 AM   Started executing query at Line 44
             Commands completed successfully.
             Total execution time: 00:00:00.147
```

QUERY 2:

```
56  -- Query 2: Enter a new department ------------------------------
57
58  CREATE PROCEDURE Q2
59      @department_number INT,
60      @department_data VARCHAR(100)
61  AS
62  BEGIN
63      INSERT INTO Department (department_number, department_data)
64      VALUES (@department_number, @department_data);
65  END;
66
```

Messages

```
7:51:28 AM   Started executing query at Line 56
             Commands completed successfully.
             Total execution time: 00:00:00.080
```

QUERY 3:

```
70  -- Query 3: Inserting Processes----------------------------------------
71
72  GO
73  CREATE PROCEDURE Q3
74      @process_id INT,
75      @process_data VARCHAR(100)
76  AS
77  BEGIN
78      INSERT INTO Process(process_id, process_data)
79      VALUES (@process_id, @process_data)
80  END;
81
82  -- Paint Process Process insertion
83  GO
84
85  CREATE PROCEDURE Q3_a
86      @process_id INT,
87      @paint_type VARCHAR(100),
88      @painting_method VARCHAR(100)
89  AS
90  BEGIN
91      INSERT INTO Paint_process(process_id, paint_type, painting_method)
92      VALUES (@process_id, @paint_type, @painting_method)
93  END;
94
```

Messages

7:52:54 AM	Started executing query at Line 70
	Commands completed successfully.
7:52:54 AM	Started executing query at Line 73
	Commands completed successfully.
7:52:54 AM	Started executing query at Line 84
	Commands completed successfully.

```sql
-- Cut Process insertion
GO
CREATE PROCEDURE Q3_b
    @process_id INT,
    @cutting_type VARCHAR(100),
    @machine_type VARCHAR(100)
AS
BEGIN
    INSERT INTO Cut_process(process_id, cutting_type, machine_type)
    VALUES (@process_id, @cutting_type, @machine_type)
END;

-- Fit Process insertion
GO
CREATE PROCEDURE Q3_c
    @process_id INT,
    @fit_type VARCHAR(100)
AS
BEGIN
    INSERT INTO Fit_process(process_id, fit_type)
    VALUES (@process_id, @fit_type)
END;
```

Messages

```
7:52:54 AM  Started executing query at Line 70
            Commands completed successfully.
7:52:54 AM  Started executing query at Line 73
            Commands completed successfully.
7:52:54 AM  Started executing query at Line 84
            Commands completed successfully.
7:52:54 AM  Started executing query at Line 97
            Commands completed successfully.
7:52:54 AM  Started executing query at Line 109
            Commands completed successfully.
```

```sql
-- insert supervised
GO
CREATE PROCEDURE Q3_d
    @process_id INT,
    @department_number INT
AS
BEGIN
    INSERT INTO supervised(process_id, department_number)
    VALUES (@process_id, @department_number)
END;
```

Messages

```
7:58:22 AM  Started executing query at Line 118
            Commands completed successfully.
7:58:22 AM  Started executing query at Line 120
            Commands completed successfully.
            Total execution time: 00:00:00.128
```

QUERY 4:

```
131
132  -- Query 4 : Enter Assembly with customer name --------------------------------
133  GO
134  CREATE PROCEDURE Q4
135      @assembly_id INT,
136      @date_ordered DATE,
137      @details TEXT,
138      @customer_name VARCHAR(100)
139  AS
140  BEGIN
141      INSERT INTO Assembly (assembly_id, date_ordered, details)
142      VALUES (@assembly_id, @date_ordered, @details);
143  END;
144
145  GO
146  CREATE PROCEDURE Q4_a
147      @assembly_id INT,
148      @process_id INT
149  AS
150  BEGIN
151      INSERT INTO pass_through (process_id, assembly_id)
152      VALUES (@process_id, @assembly_id);
153  END;
154
```

Messages

```
6:08:29 PM    Started executing query at Line 132
              Commands completed successfully.
6:08:29 PM    Started executing query at Line 134
              Commands completed successfully.
6:08:29 PM    Started executing query at Line 146
              Commands completed successfully.
              Total execution time: 00:00:00.189
```

QUERY 5:

```
156  -- Query 5 : Create account and association-------------------------------------
157  GO
158  CREATE PROCEDURE Q5
159      @account_number INT,
160      @date_established DATE
161  AS
162  BEGIN
163      INSERT INTO Account (account_number, date_established)
164      VALUES (@account_number, @date_established);
165  END;
166  -- Assembly account
167  GO
168  CREATE PROCEDURE Q5_a
169      @account_number INT,
170      @Assembly_cost float
171  AS
172  BEGIN
173      INSERT INTO Assembly_account (account_number, Assembly_cost)
174      VALUES (@account_number, @Assembly_cost);
175  END;
176
177  -- Process account
178  GO
179  CREATE PROCEDURE Q5_b
180      @account_number INT,
181      @Process_cost float
182  AS
183  BEGIN
184      INSERT INTO Process_account (account_number, Process_cost)
```

```
183  BEGIN
184      INSERT INTO Process_account (account_number, Process_cost)
185      VALUES (@account_number, @Process_cost);
186  END;
187
188
189  -- Department account
190  GO
191  CREATE PROCEDURE Q5_d
192      @account_number INT,
193      @Dept_cost float
194  AS
195  BEGIN
196      INSERT INTO Department_account (account_number, dept_cost)
197      VALUES (@account_number, @dept_cost);
198  END;
199
```

Messages

```
6:09:19 PM    Started executing query at Line 156
              Commands completed successfully.
6:09:19 PM    Started executing query at Line 158
              Commands completed successfully.
6:09:19 PM    Started executing query at Line 168
              Commands completed successfully.
6:09:19 PM    Started executing query at Line 179
              Commands completed successfully.
6:09:19 PM    Started executing query at Line 191
              Commands completed successfully.
              Total execution time: 00:00:00.302
```

QUERY 6:

```
216  -- Query 6 : Enter new Job ------------------------------------------------
217  GO
218  CREATE PROCEDURE Q6
219      @job_number INT,
220      @start_date DATE,
221      @end_date DATE
222  AS
223  BEGIN
224      INSERT INTO Job(job_number, start_date, end_date)
225      VALUES (@job_number, @start_date, @end_date)
226  END;
227
228  -- Inserting assigned
229  GO
230  CREATE PROCEDURE Q6_a
231      @process_id INT,
232      @assembly_id INT,
233      @job_number INT
234  AS
235  BEGIN
236      INSERT INTO assigned(process_id, assembly_id, job_number)
237      VALUES (@process_id, @assembly_id, @job_number)
238  END;
239
240
```

Messages

```
8:06:29 AM    Started executing query at Line 216
              Commands completed successfully.
8:06:29 AM    Started executing query at Line 218
              Commands completed successfully.
8:06:29 AM    Started executing query at Line 230
              Commands completed successfully.
              Total execution time: 00:00:00.208
```

QUERY 7:

```
241  -- Query 7 : At the completion of a job, enter the date it completed and the information relevant to the type of job
242  GO
243  CREATE PROCEDURE Q7
244      @job_number INT,
245      @end_date DATE
246  AS
247  BEGIN
248      INSERT INTO Job(job_number, end_date)
249      VALUES(@job_number, @end_date)
250  END;
251
252  -- Insert cut job
253  GO
254  CREATE PROCEDURE Q7_a
255      @job_number INT,
256      @machine_type VARCHAR(100),
257      @cut_time_used TIME,
258      @cut_material_used VARCHAR(100),
259      @cut_labor_time TIME
260  AS
261  BEGIN
262      INSERT INTO Cut_job (job_number, machine_type, time_used, material_used, labor_time)
263      VALUES (@job_number, @machine_type, @cut_time_used, @cut_material_used, @cut_labor_time)
264  END;
```

Messages

```
8:08:37 AM  Started executing query at Line 241
            Commands completed successfully.
8:08:37 AM  Started executing query at Line 243
            Commands completed successfully.
8:08:37 AM  Started executing query at Line 254
            Commands completed successfully.
            Total execution time: 00:00:00.207
```

```
266  -- Insert paint job
267  GO
268  CREATE PROCEDURE Q7_b
269      @job_number INT,
270      @paint_color VARCHAR(100),
271      @paint_volume INT,
272      @paint_labor_time TIME
273  AS
274  BEGIN
275      INSERT INTO Paint_job (job_number, color, volume, labor_time)
276      VALUES (@job_number, @paint_color, @paint_volume, @paint_labor_time);
277  END;
278
279  -- Insert fit job
280  GO
281  CREATE PROCEDURE Q7_c
282      @job_number INT,
283      @labor_time TIME
284  AS
285  BEGIN
286      INSERT INTO Fit_job (job_number, labor_time)
287      VALUES (@job_number, @labor_time);
288  END;
289
```

Messages

```
8:09:26 AM  Started executing query at Line 266
            Commands completed successfully.
8:09:26 AM  Started executing query at Line 268
            Commands completed successfully.
8:09:26 AM  Started executing query at Line 281
            Commands completed successfully.
            Total execution time: 00:00:00.211
```

QUERY 8:

```
-- Query 8 : Transaction No and Sup-cost ------------------------------
DROP PROCEDURE IF EXISTS Q8;
GO
CREATE PROCEDURE Q8
    @transaction_number INT,
    @sup_cost FLOAT
AS
BEGIN
    SET NOCOUNT ON;

    -- Inserting the transaction details into Cost_transaction table
    INSERT INTO Cost_transaction (transaction_number, sup_cost)
    VALUES (@transaction_number, @sup_cost);

    -- Update Assembly_account table
    UPDATE Assembly_account
    SET Assembly_cost = Assembly_cost + @sup_cost
    FROM Assembly_account AS aa
    INNER JOIN updates AS u ON aa.account_number = u.account_number
    WHERE u.transaction_number = @transaction_number;

    -- Update Process_account table
    UPDATE Process_account
    SET Process_cost = Process_cost + @sup_cost
    FROM Process_account AS pa
    INNER JOIN updates AS u ON pa.account_number = u.account_number
    WHERE u.transaction_number = @transaction_number;

    -- Update Department_account table
    UPDATE Department_account
    SET Dept_cost = Dept_cost + @sup_cost
    FROM Department_account AS da
    INNER JOIN updates AS u ON da.account_number = u.account_number
    WHERE u.transaction_number = @transaction_number;
END;
GO
```

Messages

```
6:11:42 PM    Started executing query at Line 275
              Commands completed successfully.
6:11:42 PM    Started executing query at Line 278
              Commands completed successfully.
              Total execution time: 00:00:00.134
```

QUERY 9:

```
316  -- Query 9 : Retrieve total cost----------------------------------------
317  DROP PROCEDURE IF EXISTS Q9;
318
319  GO
320  CREATE PROCEDURE Q9
321      @AssemblyID INT
322  AS
323  BEGIN
324      SELECT ct.Assembly_cost AS Total_Cost
325      FROM Assembly_update au
326      JOIN Assembly_account ct ON au.account_number = ct.account_number
327      WHERE au.assembly_id = @AssemblyID;
328  END;
329
330  -- Query 10 ----------------------------------------------------------------
331  GO
```

Messages

```
9:36:17 PM   Started executing query at Line 316
             Commands completed successfully.
9:36:17 PM   Started executing query at Line 320
             Commands completed successfully.
             Total execution time: 00:00:01.012
```

QUERY 11:

```
405  -- Query 11 ----------------------------------------------------------------
406
407  GO
408  CREATE PROCEDURE Q11
409      @AssemblyID INT
410  AS
411  BEGIN
412      SELECT p.process_id, j.start_date, d.department_number
413      FROM Assembly a
414      JOIN Assigned ass ON a.assembly_id = ass.assembly_id
415      JOIN Job j ON ass.job_number = j.job_number
416      JOIN Process p ON ass.process_id = p.process_id
417      JOIN supervised s ON p.process_id = s.process_id
418      JOIN Department d ON s.department_number = d.department_number
419      WHERE a.assembly_id = @AssemblyID
420      ORDER BY j.start_date;
421  END;
422
```

Messages

```
12:48:02 PM   Started executing query at Line 408
              Commands completed successfully.
              Total execution time: 00:00:00.063
```

QUERY 12:

```
423   -- Query 12 ------------------------------------------------
424   GO
425   CREATE PROCEDURE Q12
426       @CategoryMin INT,
427       @CategoryMax INT
428   AS
429   BEGIN
430       SELECT customer_name, category
431       FROM Customer
432       WHERE category BETWEEN @CategoryMin AND @CategoryMax
433       ORDER BY customer_name;
434   END;
435
```

Messages

12:51:00 PM	Started executing query at Line 425
	Commands completed successfully.
	Total execution time: 00:00:00.079

QUERY 13:

```
382
383   -- Query 13 ------------------------------------------------
384   GO
385   CREATE PROCEDURE Q13
386       @Job_No_from INT,
387       @Job_No_to INT
388   AS
389   BEGIN
390       DELETE FROM Cut_job
391       WHERE job_number BETWEEN @Job_No_from AND @Job_No_to;
392   END;
393
```

Messages

9:38:05 PM	Started executing query at Line 383
	Commands completed successfully.
9:38:05 PM	Started executing query at Line 385
	Commands completed successfully.
	Total execution time: 00:00:00.125

QUERY 14:

```
447   -- Query 14 ----------------------------
448
449   GO
450   CREATE PROCEDURE Q14
451       @JobNumber INT,
452       @NewColor VARCHAR(100)
453   AS
454   BEGIN
455       UPDATE Paint_job
456       SET color = @NewColor
457       WHERE job_number = @JobNumber;
458   END;
```

Messages

12:51:48 PM Started executing query at Line 450
Commands completed successfully.
Total execution time: 00:00:00.062

Task 5.2 - Java source program and screenshots showing its successful compilation

Query 1:

```
public static void main(String[] args) throws SQLException {
    System.out.println("WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM");

    try (final Scanner sc = new Scanner(System.in)) {
        String option = "";
        while (!option.equals("17")) {
            System.out.println(PROMPT);
            option = sc.next();

            switch (option) {
                case "1":
                    try (Connection connection = DriverManager.getConnection(URL)) {
                        Scanner sc1 = new Scanner(System.in);

                        System.out.println("Enter the Customer Name:");
                        String customerName = sc1.nextLine();

                        System.out.println("Enter the Customer Address:");
                        String customerAddress = sc1.nextLine();

                        System.out.println("Enter the Customer Category (1-10):");
                        int category = Integer.parseInt(sc1.nextLine()); // Reading category as a String and parsing to integer

                        try (PreparedStatement stmt = connection.prepareStatement(ENTER_CUSTOMER_PROCEDURE)) {
                            stmt.setString(1, customerName);
                            stmt.setString(2, customerAddress);
                            stmt.setInt(3, category);

                            int rowsAffected = stmt.executeUpdate();
                            if (rowsAffected > 0) {
                                System.out.println("New customer added successfully.");
                            } else {
                                System.out.println("No new customer was added.");
                            }
                        }
                    } catch (SQLException e) {
                        System.out.println("An error occurred while adding a new customer.");
                        e.printStackTrace();
                    }
                    break;
```

Query 2:

```
case "2":

    try {
        // Prompt the user for department information
        System.out.println("Enter the Department number: ");
        int departmentNumber = sc.nextInt();
        sc.nextLine(); // Consume the newline character
        System.out.println("Enter the Department data: ");
        String departmentData = sc.nextLine();

        // Create a SQL connection and prepare the stored procedure
        try (Connection connection = DriverManager.getConnection(URL);
             PreparedStatement statement = connection.prepareStatement(ENTER_DEPARTMENT_PROCEDURE)) {

            // Setting the parameters for the stored procedure
            statement.setInt(1, departmentNumber);
            statement.setString(2, departmentData);

            // Executing the stored procedure
            int rowsAffected = statement.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Department inserted successfully.");
            } else {
                System.out.println("Department insertion failed.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    break;
```

Query 3:

```
        case "3":
            try (Connection connection = DriverManager.getConnection(URL)) {
                System.out.println("Enter Process ID: ");
                int processId = sc.nextInt();
                sc.nextLine(); // Consume the newline character
                System.out.println("Enter Process Data: ");
                String processData = sc.nextLine();
                System.out.println("Enter Department Number: ");
                int departmentNumber = sc.nextInt();
                sc.nextLine(); // Consume the newline character
                System.out.println("Enter Process Type (1 for Paint, 2 for Cut, 3 for Fit): ");
                int processType = sc.nextInt();
                sc.nextLine(); // Consume the newline character

                // Insert process
                try (PreparedStatement processStmt = connection.prepareStatement(insertProcess)) {
                    processStmt.setInt(1, processId);
                    processStmt.setString(2, processData);
                    processStmt.executeUpdate();
                    System.out.println("Process information added successfully.");
                }

                if (processType == 1) { // Paint
                    System.out.println("Enter Paint Type: ");
                    String paintType = sc.nextLine();
                    System.out.println("Enter Painting Method: ");
                    String paintingMethod = sc.nextLine();

                    try (PreparedStatement paintStmt = connection.prepareStatement(insertPaintProcess)) {
                        paintStmt.setInt(1, processId);
                        paintStmt.setString(2, paintType);
                        paintStmt.setString(3, paintingMethod);
                        paintStmt.executeUpdate();
                        System.out.println("Paint process details added successfully.");
                    }
                } else if (processType == 2) { // Cut
                    System.out.println("Enter Cutting Type: ");
                    String cuttingType = sc.nextLine();
                    System.out.println("Enter Machine Type: ");
                    String machineType = sc.nextLine();

                    try (PreparedStatement cutStmt = connection.prepareStatement(insertCutProcess)) {
                        System.out.println("Paint process details added successfully.");
                    }
                } else if (processType == 2) { // Cut
                    System.out.println("Enter Cutting Type: ");
                    String cuttingType = sc.nextLine();
                    System.out.println("Enter Machine Type: ");
                    String machineType = sc.nextLine();

                    try (PreparedStatement cutStmt = connection.prepareStatement(insertCutProcess)) {
                        cutStmt.setInt(1, processId);
                        cutStmt.setString(2, cuttingType);
                        cutStmt.setString(3, machineType);
                        cutStmt.executeUpdate();
                        System.out.println("Cut process details added successfully.");
                    }
                } else if (processType == 3) { // Fit
                    System.out.println("Enter Fit Type: ");
                    String fitType = sc.nextLine();

                    try (PreparedStatement fitStmt = connection.prepareStatement(insertFitProcess)) {
                        fitStmt.setInt(1, processId);
                        fitStmt.setString(2, fitType);
                        fitStmt.executeUpdate();
                        System.out.println("Fit process details added successfully.");
                    }
                } else {
                    System.out.println("Invalid process type selected.");
                }

                // Associate process with department
                try (PreparedStatement superviseStmt = connection.prepareStatement(insertSupervised)) {
                    superviseStmt.setInt(1, processId);
                    superviseStmt.setInt(2, departmentNumber);
                    superviseStmt.executeUpdate();
                    System.out.println("Process associated with department successfully.");
                }

            } catch (SQLException e) {
                System.out.println("An error occurred while processing.");
                e.printStackTrace();
            }
            break;
```

Query 4:

```
    case "4":
        try {
            // Prompt the user for assembly information
            System.out.println("Enter Assembly ID: ");
            int assemblyId = sc.nextInt();
            sc.nextLine(); // Consume the newline character
            System.out.println("Enter Date Ordered (YYYY-MM-DD): ");
            String dateOrdered = sc.nextLine();
            System.out.println("Enter Assembly Details: ");
            String assemblyDetails = sc.nextLine();
            System.out.println("Enter Customer Name: ");
            String customerName = sc.nextLine();

            // Create a SQL connection and prepare the stored procedure for inserting assembly
            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(INSERT_ASSEMBLY_PROCEDURE)) {

                // Set the parameters for the INSERT_ASSEMBLY_PROCEDURE stored procedure
                statement.setInt(1, assemblyId);
                statement.setString(2, dateOrdered);
                statement.setString(3, assemblyDetails);
                statement.setString(4, customerName);

                // Execute the INSERT_ASSEMBLY_PROCEDURE stored procedure to insert the assembly
                int rowsAffected = statement.executeUpdate();

                if (rowsAffected > 0) {
                    System.out.println("Assembly inserted successfully.");
                } else {
                    System.out.println("Assembly insertion failed.");
                }
            }

            // Associate the assembly with processes
            System.out.println("Enter Process IDs to be associated with (comma-separated): ");
            String processIds = sc.nextLine();
            String[] processIdArray = processIds.split(",");

            // Create a SQL connection and prepare the stored procedure for associating processes with the asse
            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(INSERT_PASS_THROUGH_PROCEDURE)) {

                // Set the parameters for the INSERT_PASS_THROUGH_PROCEDURE stored procedure
```

```
                if (rowsAffected > 0) {
                    System.out.println("Assembly inserted successfully.");
                } else {
                    System.out.println("Assembly insertion failed.");
                }
            }

            // Associate the assembly with processes
            System.out.println("Enter Process IDs to be associated with (comma-separated): ");
            String processIds = sc.nextLine();
            String[] processIdArray = processIds.split(",");

            // Create a SQL connection and prepare the stored procedure for associating processes with the assembly
            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(INSERT_PASS_THROUGH_PROCEDURE)) {

                // Set the parameters for the INSERT_PASS_THROUGH_PROCEDURE stored procedure
                for (String processId : processIdArray) {
                    statement.setInt(1, Integer.parseInt(processId));
                    statement.setInt(2, assemblyId);

                    // Execute the INSERT_PASS_THROUGH_PROCEDURE stored procedure to associate processes
                    int rowsAffected = statement.executeUpdate();
                }

                System.out.println("Process associations updated successfully.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        break;
```

Query 5:

```
    case "5":
        try {
            // Prompt for account details
            System.out.println("Enter Account Number: ");
            int accountNumber = sc.nextInt();
            sc.nextLine();
            System.out.println("Enter Date Established (YYYY-MM-DD): ");
            String dateEstablished = sc.nextLine();

            // Connect to database and execute createAccountProcedure
            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(createAccountProcedure)) {

                statement.setInt(1, accountNumber);
                statement.setString(2, dateEstablished);
                statement.executeUpdate();
                System.out.println("Account created successfully.");
            }

            // Additional logic to handle associations with process, assembly, or department
            System.out.println("Choose an option to associate with (1 for Process, 2 for Assembly, 3 for Department): ");
            int associationOption = sc.nextInt();
            sc.nextLine(); // Consume the newline character

            if (associationOption == 1) {
                // Associating with a Process
                //System.out.println("Enter Labor Time (HH:MM:SS): ");
                //String[] laborTimeUnits = sc.nextLine().split(":");
                // int laborTime = Integer.parseInt(laborTimeUnits[0]) * 3600 + Integer.parseInt(laborTimeUnits[1]) * 60 + Inte

                // Connect to database and execute associateWithProcessProcedure
                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(associateWithProcessProcedure)) {

                    statement.setInt(1, accountNumber);
                    statement.executeUpdate();
                    System.out.println("Account associated with the process successfully.");
                }
            } else if (associationOption == 2) {
                // Associating with an Assembly
                // Connect to database and execute associateWithAssemblyProcedure
                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(associateWithAssemblyProcedure)) {
```

```
            if (associationOption == 1) {
                // Associating with a Process
                //System.out.println("Enter Labor Time (HH:MM:SS): ");
                //String[] laborTimeUnits = sc.nextLine().split(":");
                // int laborTime = Integer.parseInt(laborTimeUnits[0]) * 3600 + Integer.parseInt(laborTimeUnits[1]) * 60 + In

                // Connect to database and execute associateWithProcessProcedure
                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(associateWithProcessProcedure)) {

                    statement.setInt(1, accountNumber);
                    statement.executeUpdate();
                    System.out.println("Account associated with the process successfully.");
                }
            } else if (associationOption == 2) {
                // Associating with an Assembly
                // Connect to database and execute associateWithAssemblyProcedure
                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(associateWithAssemblyProcedure)) {

                    statement.setInt(1, accountNumber);
                    statement.executeUpdate();
                    System.out.println("Account associated with the assembly successfully.");
                }
            } else if (associationOption == 3) {
                // Connect to database and execute associateWithDepartmentProcedure
                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(associateWithDepartmentProcedure)) {

                    statement.setInt(1, accountNumber);
                    statement.executeUpdate();
                    System.out.println("Account associated with the department successfully.");
                }
            } else {
                System.out.println("Invalid option. Please select 1 for Process, 2 for Assembly, or 3 for Department.");
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        break;
    // End of case "5"
```

Query 6:

```
case "6":
    try {
        // Prompt for job details
        System.out.println("Enter Job Number: ");
        int jobNumber = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Start Date (YYYY-MM-DD): ");
        String startDate = sc.nextLine();
        System.out.println("Enter End Date (YYYY-MM-DD): ");
        String endDate = sc.nextLine();

        // Connecting to database to execute Q6 procedure
        try (Connection connection = DriverManager.getConnection(URL);
             PreparedStatement statement = connection.prepareStatement(ENTER_NEW_JOB_PROCEDURE)) {

            statement.setInt(1, jobNumber);
            statement.setString(2, startDate);
            statement.setString(3, endDate);
            statement.executeUpdate();
            System.out.println("New job entered successfully.");
        }

        // Prompt for process and assembly association with the job
        System.out.println("Enter Process ID: ");
        int processId = sc.nextInt();
        System.out.println("Enter Assembly ID: ");
        int assemblyId = sc.nextInt();

        // Connect to database and execute Q6_a procedure
        try (Connection connection = DriverManager.getConnection(URL);
             PreparedStatement statement = connection.prepareStatement(ASSOCIATE_JOB_PROCEDURE)) {

            statement.setInt(1, processId);
            statement.setInt(2, assemblyId);
            statement.setInt(3, jobNumber);
            statement.executeUpdate();
            System.out.println("Job associated with process and assembly successfully.");
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    break;
```

Query 7:

```
case "7":
    try {
        System.out.println("Enter Job Number: ");
        int jobNumber = sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Completion Date for the Job (YYYY-MM-DD): ");
        String endDate = sc.nextLine();

        // Connect to database and execute completeJobProcedure
        try (Connection connection = DriverManager.getConnection(URL);
             PreparedStatement statement = connection.prepareStatement(completeJobProcedure)) {

            statement.setInt(1, jobNumber);  // Set job_number parameter
            statement.setString(2, endDate); // Set end_date parameter
            statement.executeUpdate();
            System.out.println("Job completion date recorded successfully.");
        }

        // Prompt for job type
        System.out.println("Enter job type (1 for Cut, 2 for Paint, 3 for Fit): ");
        int jobType = sc.nextInt();
        sc.nextLine();

        if (jobType == 1) {
            // Cut job details

            System.out.println("Enter Machine Type: ");
            String machineType = sc.nextLine();
            System.out.println("Enter Cut Time Used (HH:MM:SS): ");
            String cutTimeUsed = sc.nextLine();
            System.out.println("Enter Cut Material Used: ");
            String cutMaterialUsed = sc.nextLine();
            System.out.println("Enter Cut Labor Time (HH:MM:SS): ");
            String cutLaborTime = sc.nextLine();

            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(insertCutJobProcedure)) {

                statement.setInt(1, jobNumber);
                statement.setString(2, machineType);
                statement.setString(3, cutTimeUsed);
                statement.setString(4, cutMaterialUsed);
                statement.setString(5, cutLaborTime);
```

```
            System.out.println("Enter Machine Type: ");
            String machineType = sc.nextLine();
            System.out.println("Enter Cut Time Used (HH:MM:SS): ");
            String cutTimeUsed = sc.nextLine();
            System.out.println("Enter Cut Material Used: ");
            String cutMaterialUsed = sc.nextLine();
            System.out.println("Enter Cut Labor Time (HH:MM:SS): ");
            String cutLaborTime = sc.nextLine();

            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(insertCutJobProcedure)) {

                statement.setInt(1, jobNumber);
                statement.setString(2, machineType);
                statement.setString(3, cutTimeUsed);
                statement.setString(4, cutMaterialUsed);
                statement.setString(5, cutLaborTime);
                statement.executeUpdate();
                System.out.println("Cut job details recorded successfully.");
            }
        } else if (jobType == 2) {
            // Paint job details

            System.out.println("Enter Paint Color: ");
            String paintColor = sc.nextLine();
            System.out.println("Enter Paint Volume: ");
            int paintVolume = sc.nextInt();
            sc.nextLine();
            System.out.println("Enter Paint Labor Time (HH:MM:SS): ");
            String paintLaborTime = sc.nextLine();

            try (Connection connection = DriverManager.getConnection(URL);
                 PreparedStatement statement = connection.prepareStatement(insertPaintJobProcedure)) {

                statement.setInt(1, jobNumber);
                statement.setString(2, paintColor);
                statement.setInt(3, paintVolume);
                statement.setString(4, paintLaborTime);
                statement.executeUpdate();
                System.out.println("Paint job details recorded successfully.");
            }

        } else if (jobType == 3) {
```

```
                System.out.println("Enter Paint Labor Time (HH:MM:SS): ");
                String paintLaborTime = sc.nextLine();

                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(insertPaintJobProcedure)) {

                    statement.setInt(1, jobNumber);
                    statement.setString(2, paintColor);
                    statement.setInt(3, paintVolume);
                    statement.setString(4, paintLaborTime);
                    statement.executeUpdate();
                    System.out.println("Paint job details recorded successfully.");
                }

            } else if (jobType == 3) {
                // Fit job details

                System.out.println("Enter Fit Labor Time (HH:MM:SS): ");
                String fitLaborTime = sc.nextLine();

                try (Connection connection = DriverManager.getConnection(URL);
                     PreparedStatement statement = connection.prepareStatement(insertFitJobProcedure)) {

                    statement.setInt(1, jobNumber);
                    statement.setString(2, fitLaborTime);
                    statement.executeUpdate();
                    System.out.println("Fit job details recorded successfully.");
                }
            } else {
                System.out.println("Invalid job type selected.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        break;
    // End of case "7"
```

Query 8:

```
    case "8":

        try (Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Enter Transaction Number: ");
            int transactionNumber = sc.nextInt();
            System.out.println("Enter Supplementary Cost: ");
            double supCost = sc.nextDouble();
            sc.nextLine();
            // Step 1: Insert the transaction
            try (PreparedStatement transactionStmt = connection.prepareStatement(insertTransaction)) {
                transactionStmt.setInt(1, transactionNumber);
                transactionStmt.setDouble(2, supCost);
                transactionStmt.executeUpdate();
                System.out.println("Transaction inserted successfully.");
            }

            // Step 2: Update costs in Process, Department, and Assembly cost tables
            try (PreparedStatement processCostStmt = connection.prepareStatement(updateProcessCost);
                 PreparedStatement departmentCostStmt = connection.prepareStatement(updateDepartmentCost);
                 PreparedStatement assemblyCostStmt = connection.prepareStatement(updateAssemblyCost)) {

                processCostStmt.setInt(1, transactionNumber);
                processCostStmt.setDouble(2, supCost);
                processCostStmt.executeUpdate();

                departmentCostStmt.setInt(1, transactionNumber);
                departmentCostStmt.setDouble(2, supCost);
                departmentCostStmt.executeUpdate();

                assemblyCostStmt.setInt(1, transactionNumber);
                assemblyCostStmt.setDouble(2, supCost);
                assemblyCostStmt.executeUpdate();

                System.out.println("Costs updated successfully in Process, Department, and Assembly cost tables.");
            }

            // Step 3: Link the transaction with processes, assemblies, and departments
            System.out.println("Enter Process ID: ");
            int processId = sc.nextInt();
            System.out.println("Enter Job Number: ");
            int jobNo = sc.nextInt();
            try (PreparedStatement linkProcessStmt = connection.prepareStatement(linkTransactionProcess)) {
                linkProcessStmt.setInt(1, processId);
```

```
            departmentCostStmt.executeUpdate();

            assemblyCostStmt.setInt(1, transactionNumber);
            assemblyCostStmt.setDouble(2, supCost);
            assemblyCostStmt.executeUpdate();

            System.out.println("Costs updated successfully in Process, Department, and Assembly cost tables.");
        }

        // Step 3: Link the transaction with processes, assemblies, and departments
        System.out.println("Enter Process ID: ");
        int processId = sc.nextInt();
        System.out.println("Enter Job Number: ");
        int jobNo = sc.nextInt();
        try (PreparedStatement linkProcessStmt = connection.prepareStatement(linkTransactionProcess)) {
            linkProcessStmt.setInt(1, processId);
            linkProcessStmt.setInt(2, jobNo);
            linkProcessStmt.setInt(3, transactionNumber);
            linkProcessStmt.setDouble(4, supCost);
            linkProcessStmt.executeUpdate();
            System.out.println("Transaction linked to process successfully.");
        }


    } catch (SQLException e) {
        System.out.println("An error occurred during the operation.");
        e.printStackTrace();
    }
    break;
```

Query 9:

```
case "9":
    System.out.println("Enter Assembly ID: ");
    int assemblyId = sc.nextInt();

    try (final Connection connection = DriverManager.getConnection(URL);
         final PreparedStatement statement = connection.prepareStatement(get_total_cost)) {

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

        statement.setInt(1, assemblyId);

        final ResultSet resultSet = statement.executeQuery();

        System.out.printf("Executing the stored procedure...\n");
        System.out.println("Total Cost:");

        // Unpacking the result set returned by the database and print out the total cost
        if (resultSet.next()) {
            System.out.println(String.format("Total cost incurred on assembly ID %d is: %.2f",
                    assemblyId, resultSet.getDouble(1)));
        } else {
            System.out.println("No cost data found for the specified assembly ID.");
        }
    } catch (SQLException e) {
        System.out.println("An error occurred while executing the query.");
        e.printStackTrace();
    }
    break;
```

Query 10:

```
case "10":
    System.out.println("Enter Department Number: ");
    int departmentNumber = sc.nextInt();
    sc.nextLine();

    System.out.println("Enter Completion Date (YYYY-MM-DD): ");
    String completionDate = sc.nextLine();

    try (final Connection connection = DriverManager.getConnection(URL);
         final PreparedStatement statement = connection
                                             .prepareStatement(get_total_labor_time)) {
        System.out.println("Dispatching the query...");

        statement.setInt(1, departmentNumber);
        statement.setString(2, completionDate);

        System.out.printf("Executing the stored procedure...\n");

        final ResultSet resultSet = statement.executeQuery();

        System.out.println("Total Labor Time:");

        if (resultSet.next()) {
            System.out.println(String.format("%d hours",
                                             resultSet.getInt("Total_Labor_Time")));
        } else {
            System.out.println("No labor time data found for the specified department and date.");
        }
    } catch (SQLException e) {
        System.out.println("An error occurred while executing the query.");
        e.printStackTrace();
    }
    break;
// End of case "10"
```

Query 11:

```
    case "11":
        try (Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Enter the Assembly ID for which you want process information:");
            int assId = sc.nextInt();

            try (PreparedStatement stmt = connection.prepareStatement(retrieveProcessesForAssembly)) {
                stmt.setInt(1, assId);

                ResultSet resultSet = stmt.executeQuery();
                boolean hasResults = false;

                System.out.println("\nRetrieving Process Information for Assembly ID: " + assId);
                System.out.println("-------------------------------------------------------------");
                System.out.printf("%-12s %-15s %-20s\n", "Process ID", "Start Date", "Department Number");
                System.out.println("-------------------------------------------------------------");

                while (resultSet.next()) {
                    hasResults = true;
                    int processId = resultSet.getInt("process_id");
                    Date startDate = resultSet.getDate("start_date");
                    int deptNumber = resultSet.getInt("department_number"); // Renamed variable to avoid duplication

                    System.out.printf("%-12d %-15s %-20d\n", processId, startDate.toString(), deptNumber);
                }

                if (!hasResults) {
                    System.out.println("No processes found for Assembly ID: " + assId);
                }
                System.out.println("-------------------------------------------------------------");
            }
        } catch (SQLException e) {
            System.out.println("An error occurred while retrieving process information.");
            e.printStackTrace();
        }
        break;
```

Query 12:

```
    case "12":
        try (Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("Enter the minimum category value:");
            int categoryMin = sc.nextInt();
            System.out.println("Enter the maximum category value:");
            int categoryMax = sc.nextInt();

            try (PreparedStatement stmt = connection.prepareStatement(retrieveCustomersByCategoryRange)) {
                stmt.setInt(1, categoryMin);
                stmt.setInt(2, categoryMax);

                ResultSet resultSet = stmt.executeQuery();

                System.out.println("Customers in Category Range (" + categoryMin + " to " + categoryMax + "):");
                while (resultSet.next()) {
                    String customerName = resultSet.getString("customer_name");
                    System.out.println("Customer Name: " + customerName);
                }
            }
        } catch (SQLException e) {
            System.out.println("An error occurred while retrieving customer information.");
            e.printStackTrace();
        }

        break;
```

Query 13:

```
    case "13":
        try (Connection connection = DriverManager.getConnection(URL)) {
            System.out.println("\n--- Delete Cut Jobs ---");
            System.out.println("Enter the starting job number of the range:");
            int jobNoFrom = sc.nextInt();
            System.out.println("Enter the ending job number of the range:");
            int jobNoTo = sc.nextInt();

            try (PreparedStatement stmt = connection.prepareStatement(deleteCutJobs)) {
                stmt.setInt(1, jobNoFrom);
                stmt.setInt(2, jobNoTo);

                int rowsAffected = stmt.executeUpdate();
                System.out.println("\n" + rowsAffected + " cut job(s) deleted successfully.");
                System.out.println("-----------------------");
            }
        } catch (SQLException e) {
            System.out.println("\nError: Unable to delete cut jobs.");
            System.out.println("-----------------------");
            e.printStackTrace();
        }
        break;
```

Query 14:

```
case "14":
    try (Connection connection = DriverManager.getConnection(URL)) {
        System.out.println("Enter the Job Number of the paint job you want to change:");
        int jobNumber = sc.nextInt();
        sc.nextLine(); // Consume the newline character left after nextInt
        System.out.println("Enter the New Color for the paint job:");
        String newColor = sc.nextLine();

        try (PreparedStatement stmt = connection.prepareStatement(changePaintJobColor)) {
            stmt.setInt(1, jobNumber);
            stmt.setString(2, newColor);

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("The color of the paint job has been successfully updated.");
            } else {
                System.out.println("No paint job was updated. Please check the Job Number.");
            }
        }
    } catch (SQLException e) {
        System.out.println("An error occurred while updating the paint job color.");
        e.printStackTrace();
    }
    break;
```

After Compiling:

```
WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details (
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100/
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
```

Successful Insertion of Data into DB relations:

```
274
275   SELECT * FROM Customer
276
```

Results | Messages

	customer_name	customer_address	category
1	Alice Johnson	123 Maple Street, Springfield	5
2	Bob Smith	456 Oak Avenue, River City	4
3	Caroline Davis	789 Pine Road, Mountain Town	3
4	David Wilson	321 Birch Lane, Lakeside	2
5	Emily Clark	654 Elm Court, Sunshine Vill…	1

```
274
275   SELECT * FROM Department
276
```

Results | Messages

	department_number	department_data
1	101	Research and Development
2	102	Environmental Conservation
3	103	Robotics and Automation
4	104	Aerospace Engineering
5	105	Marine Biology Research

```
275  SELECT * FROM Process
276
```

Results | Messages

	process_id	process_data
1	101	High Precision Fitting
2	102	Automotive Painting
3	103	Metal Cutting
4	104	Component Assembly
5	105	Protective Coating
6	106	Wood Cutting
7	107	Precision Engineering
8	108	Aircraft Painting
9	109	Stone Carving
10	110	Electronic Assembly

```
275  SELECT * FROM Fit_process
276
```

Results | Messages

	process_id	fit_type
1	101	Laser Alignment
2	104	Modular Assembly
3	107	Micro-Fitting
4	110	Circuit Board Assembly

```
275  SELECT * FROM Cut_process
276
```

Results | Messages

	process_id	cutting_type	machine_type
1	103	Plasma	CNC Plasma Cutter
2	106	Circular Saw	Automated Saw
3	109	Laser Cutting	3D Laser Cutter

```
275    SELECT * FROM Paint_process
276
```

Results | Messages

	process_id	paint_type	painting_method
1	102	Gloss Finish	Spray
2	105	Anti-Corrosion	Dip Coating
3	108	Thermal Resistant	Airbrush

Results grid

```
275    SELECT * FROM Assembly
276
```

Results | Messages

	assembly_id	date_ordered	details
1	2001	2023-03-01	Robot Arm Assembly
2	2002	2023-03-02	Satellite Communication System
3	2003	2023-03-03	Deep Sea Exploration Module
4	2004	2023-03-04	Automated Farming Equipment
5	2005	2023-03-05	Electric Vehicle Battery Pack
6	2006	2023-03-06	Solar Power Inverter
7	2007	2023-03-07	AR Gaming Console
8	2008	2023-03-08	Industrial 3D Printer
9	2009	2023-03-09	Smart Home Security System
10	2010	2023-03-10	Autonomous Drone

```
282    SELECT * FROM Assembly
```

Results | Messages

	assembly_id	date_ordered	details
1	2001	2023-03-01	Robot Arm Assembly
2	2002	2023-03-02	Satellite Communication System
3	2003	2023-03-03	Deep Sea Exploration Module
4	2004	2023-03-04	Automated Farming Equipment
5	2005	2023-03-05	Electric Vehicle Battery Pack
6	2006	2023-03-06	Solar Power Inverter
7	2007	2023-03-07	AR Gaming Console
8	2008	2023-03-08	Industrial 3D Printer
9	2009	2023-03-09	Smart Home Security System
10	2010	2023-03-10	Autonomous Drone

```
282  SELECT * FROM pass_through
283
```

Results | Messages

	process_id	assembly_id
1	101	2001
2	102	2002
3	103	2003
4	104	2004
5	105	2005
6	106	2006
7	107	2007
8	108	2008
9	109	2009
10	110	2010

```
282  SELECT * FROM Account
283
```

Results | Messages

	account_number	date_established
1	1001	2023-01-01
2	1002	2023-01-02
3	1003	2023-01-03
4	1004	2023-01-04
5	1005	2023-01-05
6	1006	2023-01-06
7	1007	2023-01-07
8	1008	2023-01-08
9	1009	2023-01-09
10	1010	2023-01-10

```
282  SELECT * FROM Process_account
283
```

Results | Messages

	account_number	Process_cost
1	1002	300
2	1005	350
3	1008	250

```
282  SELECT * FROM Department_account
```

Results | Messages

	account_number	Dept_cost
1	1003	400
2	1006	450
3	1009	500

```
282  SELECT * FROM Assembly_account
```

Results | Messages

	account_number	Assembly_cost
1	1001	500
2	1004	600
3	1007	550
4	1010	700

```
282  SELECT * FROM Job
```

Results | Messages

	job_number	start_date	end_date
1	5001	2023-03-01	2023-03-05
2	5002	2023-03-02	2023-03-06
3	5003	2023-03-03	2023-03-07
4	5004	2023-03-04	2023-03-08
5	5005	2023-03-05	2023-03-09
6	5006	2023-03-06	2023-03-10
7	5007	2023-03-07	2023-03-11
8	5008	2023-03-08	2023-03-12
9	5009	2023-03-09	2023-03-13
10	5010	2023-03-10	2023-03-14

```
282  SELECT * FROM Cut_job
283
```

Results | Messages

	job_number	machine_type	time_used	material_used	labor_time
1	5007	Machine A	01:30:00	Steel	02:00:00
2	5008	Machine B	02:00:00	Aluminum	01:45:00
3	5009	Machine C	01:15:00	Copper	01:30:00
4	5010	Machine D	03:00:00	Plastic	02:15:00

```
282  SELECT * FROM Fit_job
```

Results | Messages

	job_number	machine_type	time_used	material_used	labor_time
1	5007	Machine A	01:30:00	Steel	02:00:00
2	5008	Machine B	02:00:00	Aluminum	01:45:00
3	5009	Machine C	01:15:00	Copper	01:30:00
4	5010	Machine D	03:00:00	Plastic	02:15:00

```
282  SELECT * FROM Paint_job
283
```

Results | Messages

	job_number	color	volume	labor_time
1	5001	Red	10	02:30:00
2	5002	Blue	15	03:45:00
3	5003	Green	8	01:45:00

```
282  SELECT * FROM Cost_transaction
283
```

Results | Messages

	transaction_number	sup_cost
1	101	500
2	102	150
3	103	300
4	104	250
5	105	600
6	106	400
7	107	350
8	108	200
9	109	450
10	110	550

```
282  SELECT * FROM Assembly_account
283
```

Results | Messages

	account_number	Assembly_cost
1	1001	500
2	1004	600
3	1007	550
4	1010	700

```
282  SELECT * FROM Process_account
283
```

Results | Messages

	account_number	Process_cost
1	1002	300
2	1005	350
3	1008	250

Results grid

```
282    SELECT * FROM Department_account
283
```

Results | Messages

	account_number	Dept_cost
1	1003	400
2	1006	450
3	1009	500

Task 6:

6.1 Screenshot showing the testing of query 1

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details (
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100/
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!

1
Enter the Customer Name:
Alice Johnson
Enter the Customer Address:
123 Maple Street, Springfield
Enter the Customer Category (1-10):
5
New customer added successfully.
```

```
281
282    SELECT * FROM Customer
```

Results | Messages

	customer_name	customer_address	category
1	Alice Johnson	123 Maple Street, Springfield	5

6.2 Screenshot showing the testing of query 2:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of detai
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
2
Enter the Department number:
101
Enter the Department data:
Research and Development
Department inserted successfully.

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
```

```
695
696    SELECT* from Department
```

Results | Messages

	department_n…	department_data
1	101	Research and Development

6.3 Screenshot showing the testing of query 3:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more proces
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their curre
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
3
Enter Process ID:
101
Enter Process Data:
High Precision Fitting
Enter Department Number:
102
Enter Process Type (1 for Paint, 2 for Cut, 3 for Fit):
1
Process information added successfully.
Enter Paint Type:
Anti-Corroson
Enter Painting Method:
Dip Coating
Paint process details added successfully.
```

```
695    SELECT * FROM Process
```

Results | Messages

	process_id	process_data
1	101	High Precision Fitting

```
695    SELECT * FROM Paint_process
```

Results | Messages

	process_id	paint_type	painting_method
1	101	Anti-Corroson	Dip Coating

6.4 Screenshot showing the testing of query 4:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the departme
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
4
Enter Assembly ID:
2002
Enter Date Ordered (YYYY-MM-DD):
2023-03-02
Enter Assembly Details:
Satellite Communication System
Enter Customer Name:
Bob Smith
Assembly inserted successfully.
Enter Process IDs to be associated with (comma-separated):
102
```

```
282    SELECT * FROM Assembly
```

Results | Messages

	assembly_id	date_ordered	details
1	2002	2023-03-02	Satellite Communication System

6.5 Screenshot showing the testing of query 5:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department respon
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
5
Enter Account Number:
1001
Enter Date Established (YYYY-MM-DD):
2023-01-01
Account created successfully.
Choose an option to associate with (1 for Process, 2 for Assembly, 3 for Department):
2
Enter the cost:
500.0
```

```
282  SELECT * FROM Account
283
```

Results | Messages

	account_number	date_established
1	1001	2023-01-01

```
282  SELECT * FROM Assembly_account
283
```

Results | Messages

	account_number	Assembly_cost
1	1001	500

6.6 Screenshot showing the testing of query 6:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent)
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-c
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the depai
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
6
Enter Job Number:
5001
Enter Start Date (YYYY-MM-DD):
2023-03-01
Enter End Date (YYYY-MM-DD):
2023-03-05
New job entered successfully.
Enter Process ID:
101
Enter Assembly ID:
2001
Job associated with process and assembly successfully.

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent)
```

```
282    SELECT * FROM Job
283
```

Results | Messages

	job_number	start_date	end_date
1	5001	2023-03-01	2023-03-05

6.7 Screenshot showing the testing of query 7:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrec
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/d
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
7
Enter Job Number:
5005
Enter Completion Date for the Job (YYYY-MM-DD):
2023-05-04
Job completion date recorded successfully.
Enter job type (1 for Cut, 2 for Paint, 3 for Fit):
1
Enter Machine Type:
Machine A
Enter Cut Time Used (HH:MM:SS):
01:30:00
Enter Cut Material Used:
Steel
Enter Cut Labor Time (HH:MM:SS):
02:00:00
Cut job details recorded successfully.

Please select one of the options below:
```

```
282    SELECT * FROM Cut_job
283
```

Results | Messages

	job_number	machine_type	time_used	material_used	labor_time
1	5005	Machine A	01:30:00	Steel	02:00:00

6.8 Screenshot showing the testing of query 8:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to th
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered ar
5) Create a new account and associate it with the process, assembly, or department to which it is
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the typ
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected acco
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
8
Enter Transaction Number:
101
Enter Supplementary Cost:
500.0
Transaction inserted successfully.
```

```
282    SELECT * FROM Cost_transaction
283
```

Results | Messages

	transaction_number	sup_cost
1	101	500

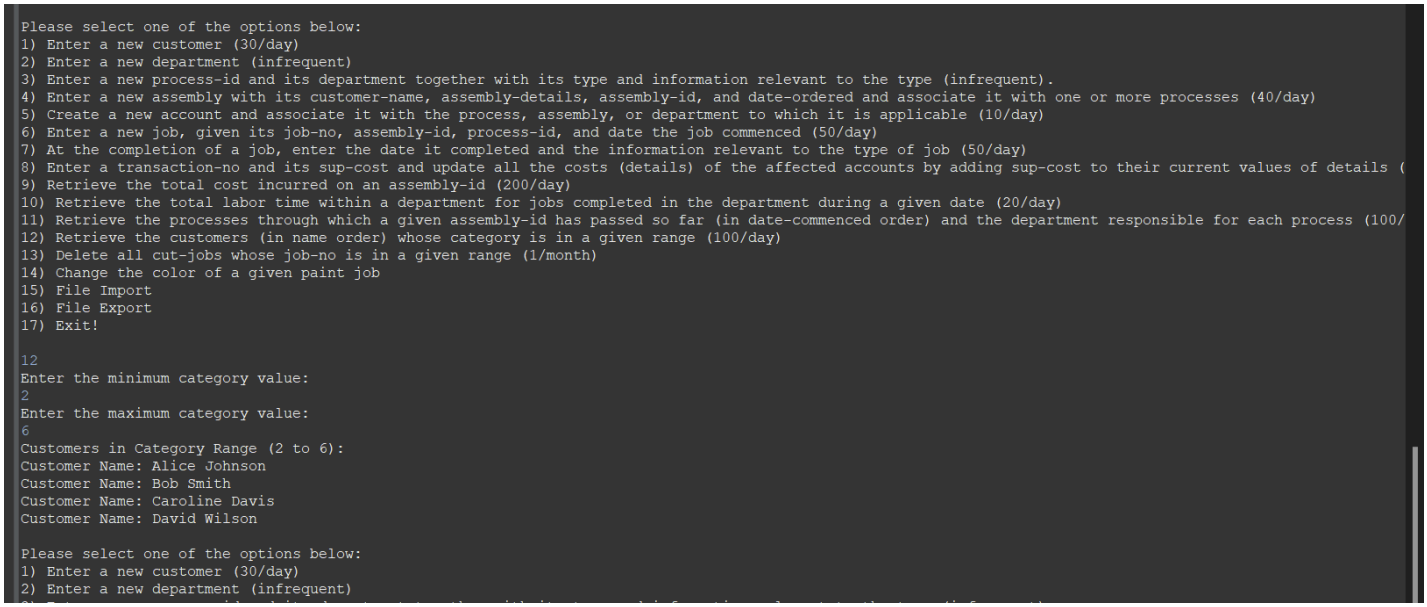
6.8 Screenshot showing the testing of query 9:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with or
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cos
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the departm
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
9
Enter Assembly ID:
2001
Dispatching the query...
Executing the stored procedure...
Cost: 300.0
```

6.11 Screenshot showing the testing of query 11:

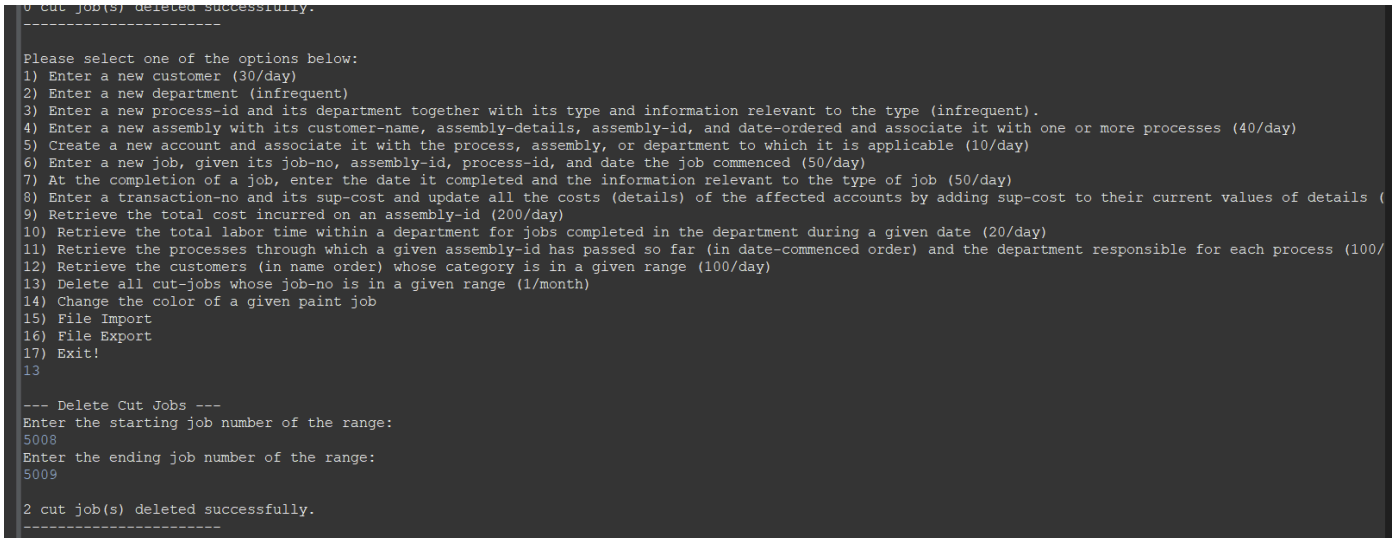
```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details (
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100/
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
11
Enter the Assembly ID for which you want process information:
2005

Retrieving Process Information for Assembly ID: 2005
------------------------------------------------------------
Process ID   Start Date      Department Number
------------------------------------------------------------
105          2023-03-05      102
------------------------------------------------------------
```

6.12 Screenshot showing the testing of query 12



6.13 Screenshot showing the testing of query 13:

QUERY 13:



Before

```
696    SELECT* from Cut_job
```

Results | Messages

	job_number	machine_type	time_used	material_used	labor_time
1	5007	Machine A	01:30:00	Steel	02:00:00
2	5008	Machine B	02:00:00	Aluminum	01:45:00
3	5009	Machine C	01:15:00	Copper	01:30:00
4	5010	Machine D	03:00:00	Plastic	02:15:00

After

```
696    SELECT* from Cut_job
```

Results | Messages

	job_number	machine_type	time_used	material_used	labor_time
1	5007	Machine A	01:30:00	Steel	02:00:00
2	5010	Machine D	03:00:00	Plastic	02:15:00

6.14 Screenshot showing the testing of query 14

QUERY 14:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
14
Enter the Job Number of the paint job you want to change:
5003
Enter the New Color for the paint job:
Yellow
The color of the paint job has been successfully updated.
```

Before:

```
282    SELECT * FROM Paint_job
283
```

Results | Messages

	job_number	color	volume	labor_time
1	5001	Red	10	02:30:00
2	5002	Blue	15	03:45:00
3	5003	Green	8	01:45:00

After:

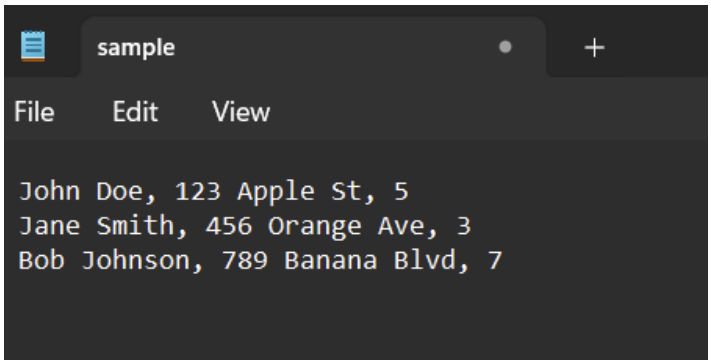
```
696    SELECT* from Paint_job
```

Results | Messages

	job_number	color	volume	labor_time
1	5001	Red	10	02:30:00
2	5002	Blue	15	03:45:00
3	5003	Yellow	8	01:45:00

6.15 Screenshot showing the testing of query 15:(import)

```
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more proc
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their cur
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsibl
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
15
Enter the file name:
sample.txt
File import complete!

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more proc
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
```



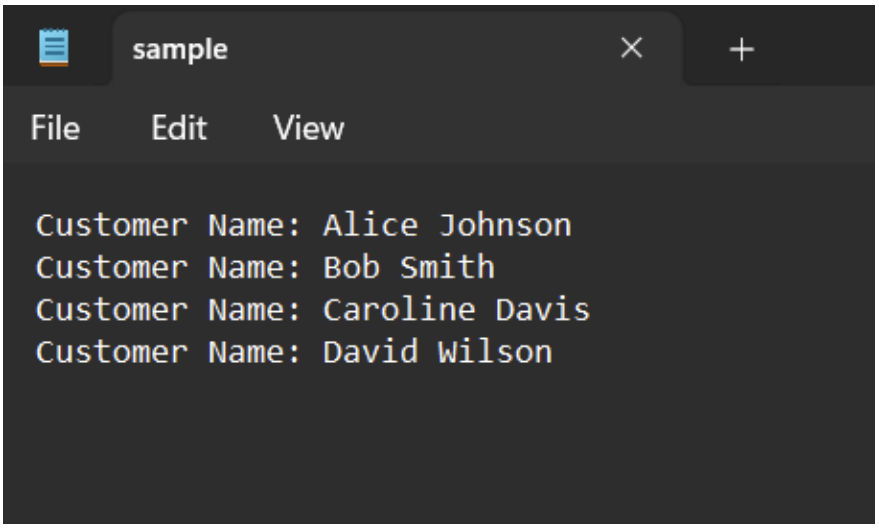

```
282  SELECT * FROM Customer
283
```

Results | Messages

	customer_name	customer_address	category
1	Alice Johnson	123 Maple Street, Springfield	5
2	Bob Johnson	789, Banana Blvd	7
3	Bob Smith	456 Oak Avenue, River City	4
4	Caroline Davis	789 Pine Road, Mountain Town	3
5	David Wilson	321 Birch Lane, Lakeside	2
6	Emily Clark	654 Elm Court, Sunshine Village	1
7	Jane Smith	456 Orange Ave	3
8	John Doe	123, Apple St	5

6.15 Screenshot showing the testing of query 16:(export)

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequ
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/da
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (2
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
16
Enter the minimum category value:
2
Enter the maximum category value:
6
Enter the output file name:
sample
Customers in Category Range (2 to 6) exported to C:/Users/Kaustubh/Downloads/sample
```



6.16 Testing types of errors:

Error Detection – 1: Primary Key Violation (Duplicate Insertion)

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details (
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department responsible for each process (100/
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
1
Enter the Customer Name:
John Wick
Enter the Customer Address:
2 street
Enter the Customer Category (1-10):
1
An error occurred while adding a new customer.
com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__Customer__5B894ACABC942D8E'. Cannot insert duplicate key in object '
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparedState
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedStatem
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7627)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:3912)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:268)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:242)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLServerPreparedStatement.java:486
        at IP.main(IP.java:140)

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
```

Error Detection – 2: Syntax Error while executing Stored Procedure

```
15) File Import
16) File Export
17) Exit!
2
Enter the Department number:
1
Enter the Department data:
Action
com.microsoft.sqlserver.jdbc.SQLServerException: Procedure or function 'Q2' expects parameter '@department_data', which was not supplied.
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:265)
        at com.microsoft.sqlserver.jdbc@11.2.0.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1673)

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more processes (40/day)
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
```

IP.java

```
26      String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.databas
27      HOSTNAME, DBNAME, USERNAME, PASSWORD);
28
29      // Stored Procedure templates
30      //Query1
31      final static String ENTER_CUSTOMER_PROCEDURE = "EXEC Q1 @customer_name = ?, @customer_address = ?, @category = ?";
32      //Query2
33      final static String ENTER_DEPARTMENT_PROCEDURE = "EXEC Q2 @department_number = ?, @john_wick = ?";
```

Error Detection – 3: Date Error: It detects that the date is not possible

```
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
4
Enter Assembly ID:
123
Enter Date Ordered (YYYY-MM-DD):
2040-13-35
Enter Assembly Details:
Sample
Enter Customer Name:
John Wick
com.microsoft.sqlserver.jdbc.SQLServerException: Error converting data type nvarchar to date.

Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequer
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it w
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/da
```

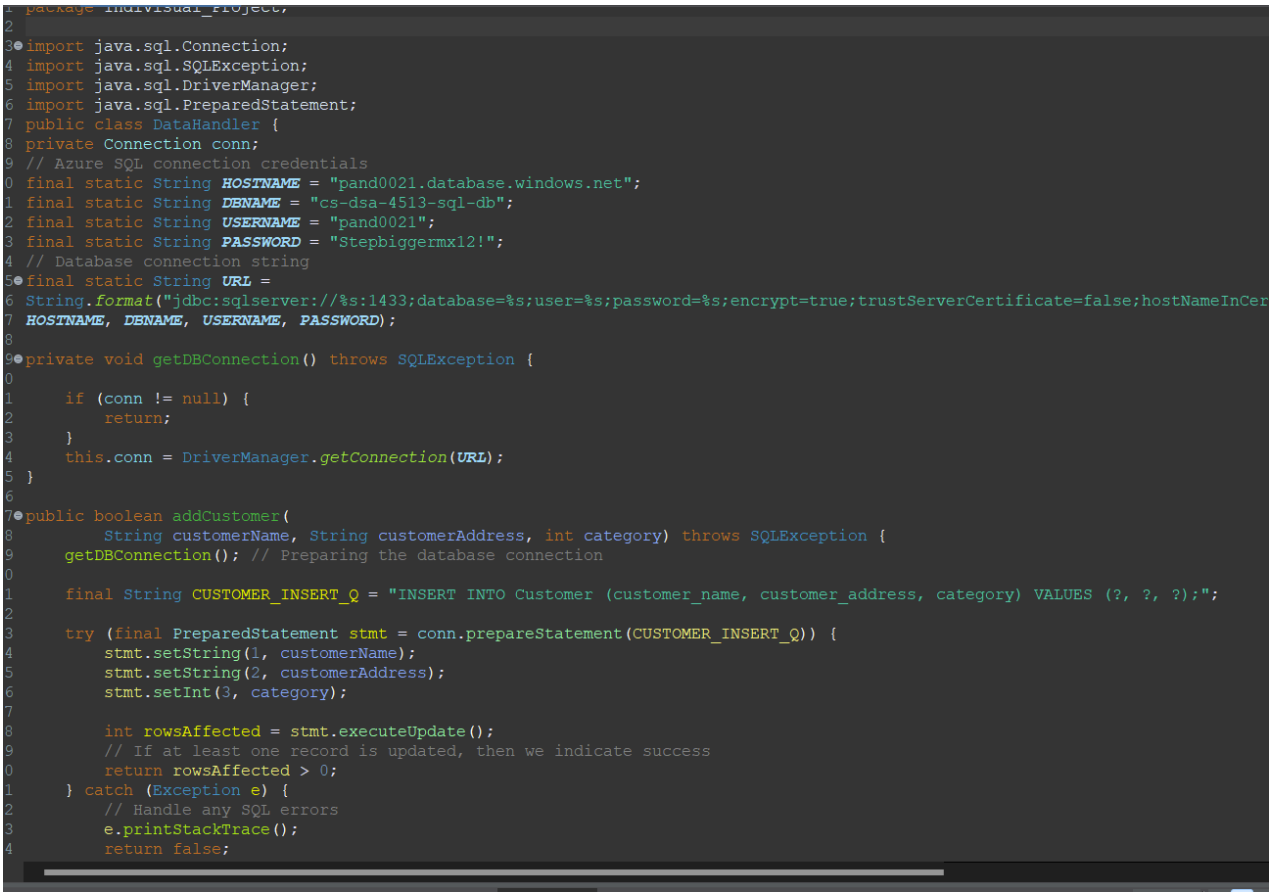
6.17 Screenshot showing the testing of query 17:

```
Please select one of the options below:
1) Enter a new customer (30/day)
2) Enter a new department (infrequent)
3) Enter a new process-id and its department together with its type and information relevant to the type (infrequent).
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or more p
5) Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day)
6) Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
7) At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day)
8) Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their
9) Retrieve the total cost incurred on an assembly-id (200/day)
10) Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day)
11) Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department respons
12) Retrieve the customers (in name order) whose category is in a given range (100/day)
13) Delete all cut-jobs whose job-no is in a given range (1/month)
14) Change the color of a given paint job
15) File Import
16) File Export
17) Exit!
17
Exiting the program. Goodbye!
```

Task 7. Web database application and its execution

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

DataHandler for Adding Customer:



```
package individual_project;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class DataHandler {
private Connection conn;
// Azure SQL connection credentials
final static String HOSTNAME = "pand0021.database.windows.net";
final static String DBNAME = "cs-dsa-4513-sql-db";
final static String USERNAME = "pand0021";
final static String PASSWORD = "Stepbiggermx12!";
// Database connection string
final static String URL =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCer
HOSTNAME, DBNAME, USERNAME, PASSWORD);

private void getDBConnection() throws SQLException {

    if (conn != null) {
        return;
    }
    this.conn = DriverManager.getConnection(URL);
}

public boolean addCustomer(
        String customerName, String customerAddress, int category) throws SQLException {
    getDBConnection(); // Preparing the database connection

    final String CUSTOMER_INSERT_Q = "INSERT INTO Customer (customer_name, customer_address, category) VALUES (?, ?, ?);";

    try (final PreparedStatement stmt = conn.prepareStatement(CUSTOMER_INSERT_Q)) {
        stmt.setString(1, customerName);
        stmt.setString(2, customerAddress);
        stmt.setInt(3, category);

        int rowsAffected = stmt.executeUpdate();
        // If at least one record is updated, then we indicate success
        return rowsAffected > 0;
    } catch (Exception e) {
        // Handle any SQL errors
        e.printStackTrace();
        return false;
```

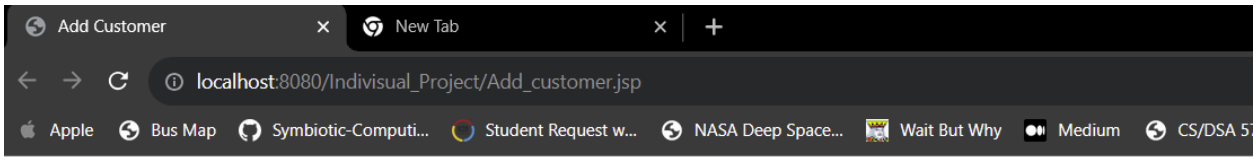
Add_cust.jsp file:

```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="Indivisual_Project.DataHandler"%>
    <%@ page import="java.sql.ResultSet"%>
    <%@ page import="java.sql.SQLException"%>
    <%
    // Instantiate the handler responsible for database operations
    DataHandler handler = new DataHandler();

    // Retrieve attribute values passed from the form
    String customerName = request.getParameter("customer_name");
    String customerAddress = request.getParameter("customer_address");
    String categoryString = request.getParameter("category");

    // Simple validation to ensure required fields are filled
    if (customerName.equals("") || customerAddress.equals("") || categoryString.equals("")) {
        // Redirect back to the form if validation fails
        response.sendRedirect("add_customer_form.jsp");
    } else {
        int category = Integer.parseInt(categoryString);
        boolean success = false;
        try {
            // Perform the insert operation
            success = handler.addCustomer(customerName, customerAddress, category);
        } catch(SQLException e) {
            // Handle SQL exception if any
            e.printStackTrace();
        }
        if (!success) { // If the insert operation failed
            %>
                <h2>There was a problem inserting the customer</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The Customer:</h2>
```

Add_customer webpage:

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

        <form action="add_cust.jsp" method="post">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border="1">
                <tr>
                    <th colspan="2">Enter the Customer Data:</th>
                </tr>
                <tr>
                    <td>Customer Name:</td>
                    <td><div style="text-align: center;">
                        <input type="text" name="customer_name">
                    </div></td>
                </tr>
                <tr>
                    <td>Customer Address:</td>
                    <td><div style="text-align: center;">
                        <input type="text" name="customer_address">
                    </div></td>
                </tr>
                <tr>
                    <td>Category:</td>
                    <td><div style="text-align: center;">
                        <input type="number" name="category" min="1">
                    </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>
```

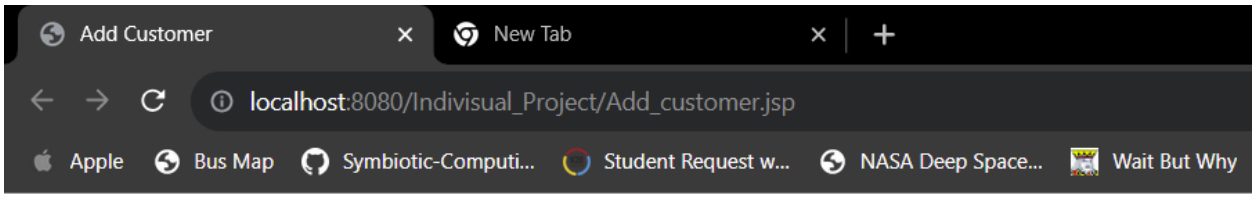
After compiling the code:



Add Customer

Enter the Customer Data:	
Customer Name:	
Customer Address:	
Category:	
Clear	Insert

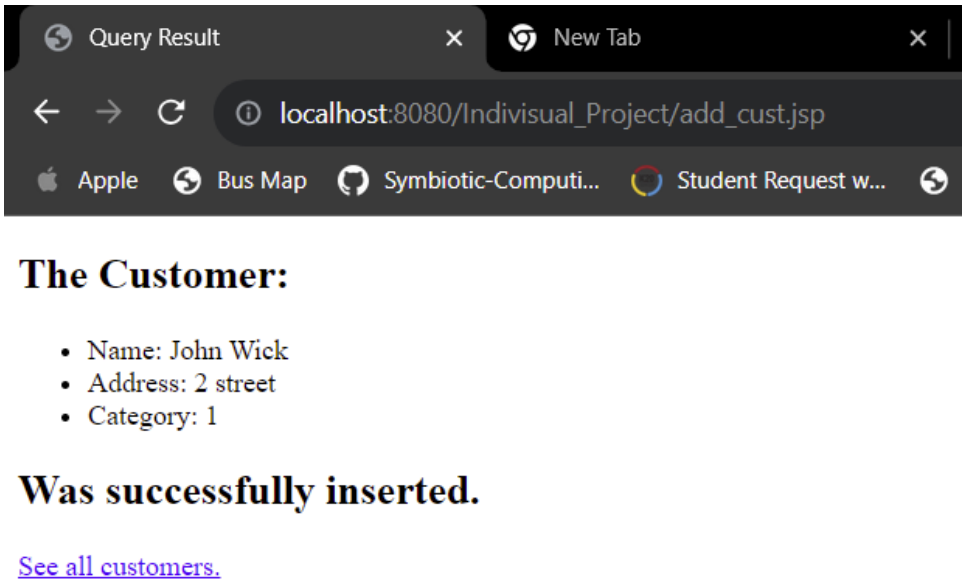
Entering values:



Add Customer

Enter the Customer Data:	
Customer Name:	John Wick
Customer Address:	2 street
Category:	1
Clear	Insert

Result:



Verification:

```
282  SELECT * FROM Customer
283
284
285
```

Results | Messages

	customer_name	customer_address	category
1	Alice Johnson	123 Maple Street, Springfield	5
2	Bob Smith	456 Oak Avenue, River City	4
3	Caroline Davis	789 Pine Road, Mountain Town	3
4	David Wilson	321 Birch Lane, Lakeside	2
5	Emily Clark	654 Elm Court, Sunshine Village	1
6	John Wick	2 street	1
7	Kaustubh	1 street	3