Course Name: Database Management Systems

Course Number: CS/DSA – 4513

Section Number: 001

Semester and Year: Fall 2023

Instructor Name: Dr. Le Gruenwald

Author Name: Vasireddy Ujwala

Author Id: 113634633

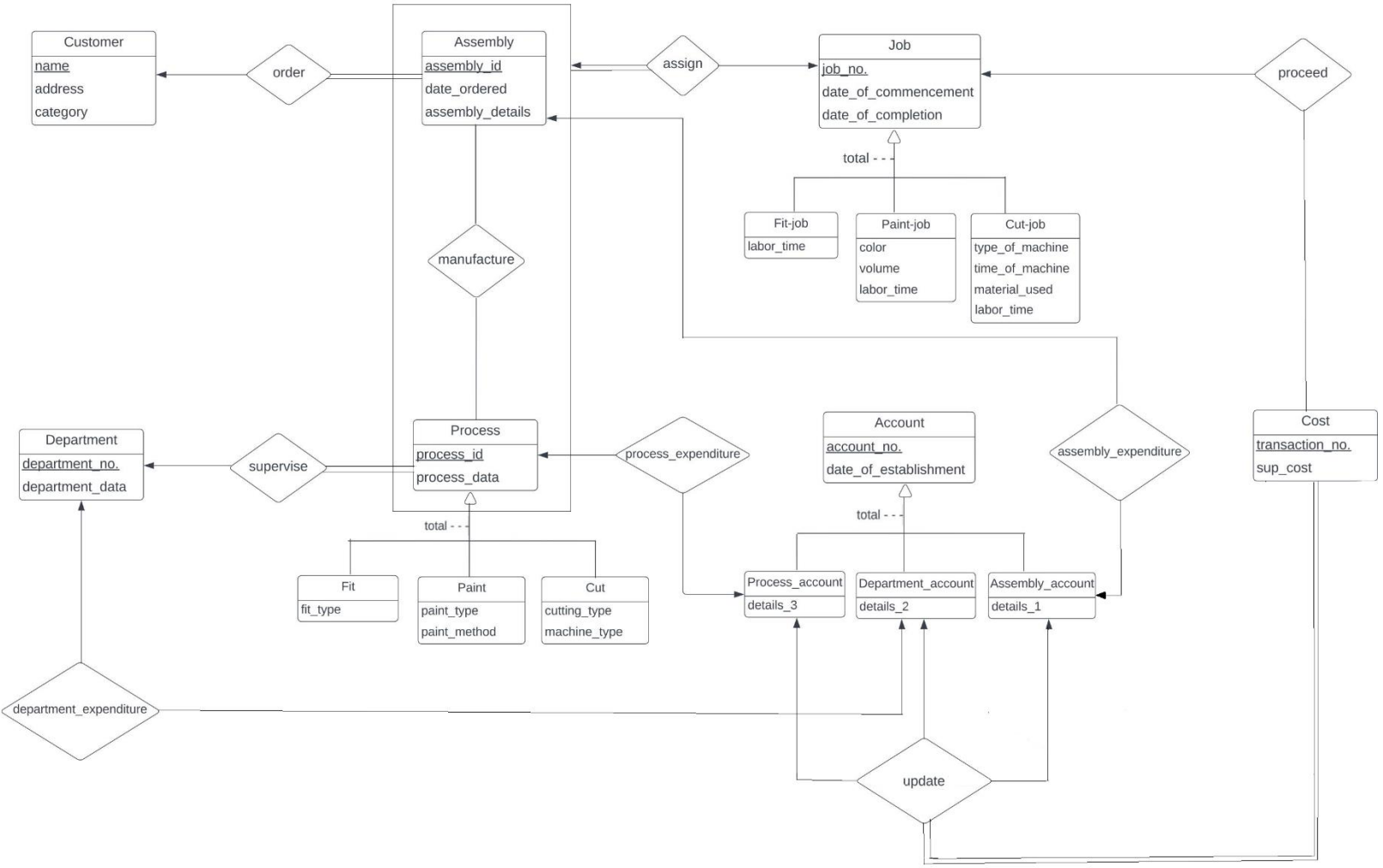
Email-id: Ujwala.Vasireddy-1@ou.edu

Title: A JOB-SHOP ACCOUNTING SYSTEM

UJWALA VASIREDDY
113634633
INDIVIDUAL PROJECT: A JOB-SHOP ACCOUNTING SYSYTEM

UJWALA VASIREDDY
113634633
INDIVIDUAL PROJECT: A JOB-SHOP ACCOUNTING SYSYTEM

Task 1: ER DIAGRAM



Task 2: RELATIONAL DATABASE SCHEMAS

Customer (cname, caddress, category)

Assembly (assembly_id, date_ordered, assembly_details, cname)

Department (department_no, department_data)

Process (process_id, process_data, department_no)

Fit_process (process_id, fit_type)

Paint_process (process_id, paint_type, paint_method)

Cut_process (process_id, cutting_type, machine_type)

Job (job_no, date_of_commencement, date_of_completion)

Fit_job (job_no., fit_labor_time)

Paint_job (job_no., color, volume, paint_labor_time)

Cut_job (job_no., type_of_machine, time_of_machine, material_used, cut_labor_time)

manufacture (assembly_id, process_id, job_no)

Assembly_account (assembly_account_no, date_of_establishment, details_1)

Department_account (department_account_no, date_of_establishment, details_2)

Process_account (process_account_no, date_of_establishment, details_3)

assembly_expenditure (assembly_account_no, assembly_id)

department_expenditure (department_account_no, department_no)

process_expenditure (process_account_no, process_eid)

Cost (transaction_no, sup_cost, assembly_account_no, department_account_no, process_account_no)

proceed (transaction_no, job_no)

TASK 3.1: DISCUSSION OF STORAGE STRUCTURES FOR TABLES

Table Name	Query# Type	Search Key	Query Frequency	Selected file organization	Justifications
Customer	(1) Insertion		30/day	Index Sequential File on search key category	Given that range searches are prevalent in category, the Index Sequential file is the preferred choice for conducting such searches. The Index Sequential file employs a combination of a data file and an index, enabling quick access to specific ranges of data.
	(12) Range Search	category	100/day		
Assembly	(4) Insertion		40/day	Heap File	Heap files are considered advantageous for insertion operations due to their simplicity and efficiency in accommodating dynamic data growth.
Department	(2) Insertion		infrequent	Dynamic Hashing with a hash key job_no	Dynamic hashing is chosen because random search is more frequent and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(10) Random Search	job_no	20/day		
Process	(3) Insertion		infrequent	Dynamic Hashing with a hash key process_id	Dynamic hashing is chosen because random search is more frequent on process_id and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly
	(10) Random Search	department_no	20/day		
	(11) Random Search	process_id	100/day		
Fit_process	(3) Insertion		infrequent	Heap File	Heap files are considered advantageous for insertion operations due to their simplicity and efficiency in accommodating dynamic data growth.
Paint_process				Heap File	Heap files are considered advantageous for insertion operations due to their simplicity and efficiency in

Cut_process	(3) Insertion		infrequent	Heap File	Heap files are considered advantageous for insertion operations due to their simplicity and efficiency in accommodating dynamic data growth.
Job	(6) Insertion		50/day	Dynamic Hashing with a hash key job_no	Dynamic hashing is chosen because random search is more frequent on job_no and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(7) Random Search(update)	job_no	50/day		
	(10) Random Search	date_of_completion	20/day		
	(11) Random Search	job_no	100/day		
Fit_job	(6) Insertion		50/day	Dynamic Hashing with a hash key job_no	Dynamic hashing may be preferable over heap files in scenarios where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a
	(7) Random Search (update)	job_no	50/day		
Paint_job	(6) Insertion		50/day	Dynamic Hashing with a hash key job_no	Dynamic hashing may be preferable over heap files in scenarios where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a specific key.
	(7) Random Search (update)	job_no	50/day		
	(14) Random Search (update)	job_no	1/week		
Cut_job	(6) Insertion		50/day	Dynamic Hashing with a hash key job_no	We don't consider Index Sequential file or a sparse index file here. Dynamic hashing may be preferable over heap files in scenarios where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a specific key.
	(7) Random Search (update)	job_no	50/day		
	(13) Range Search	job_no	1/month		
	(13) Deletion	job_no	1/month		

Table Name	Query# Type	Search Key	Query Frequency	Selected file organization	Justifications
Manufacture	(4) Insertion		40/day	Dynamic Hashing with a hash key assembly_id	Dynamic hashing may be preferable where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a specific key. The search key assembly_id would be really helpful for update operation as well.
	(6) Random Search (update)	job_no	50/day		
	(10) Random Search	process_id	20/day		
	(11) Random Search	assembly_id	100/day		
Assembly_account	(5) Insertion		10/day	Dynamic Hashing with a hash key assembly_account_no	Dynamic hashing is chosen because random search is more frequent on assembly_account_no and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(8) Random Search (update)	assembly_account_no	50/day		
Department_account	(5) Insertion		10/day	Dynamic Hashing with a hash key department_account_no	Dynamic hashing is chosen because random search is more frequent on department_account_no and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(8) Random Search (update)	department_account_	50/day		
Process_account	(5) Insertion		10/day	Dynamic Hashing with a hash key process_account_no	Dynamic hashing is chosen because random search is more frequent on process_account_no and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(8) Random Search (update)	process_account_no	50/day		
Assembly_expenditure	(5) Insertion		10/day	Dynamic Hashing with a hash key assembly_id	Dynamic hashing is chosen because random search is more frequent on assembly_id and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(5) Random Search (update)	assembly_account_no	10/day		
	(9) Random Search	assembly_id	200/day		
Department_expenditure	(5) Insertion		10/day	Dynamic Hashing with a hash key department_account_no	Dynamic hashing may be preferable over heap files in scenarios where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a specific key.
	(5) Random Search (update)	department_account_	10/day		
Process_expenditure	(5) Insertion		10/day	Dynamic Hashing with a hash key process_account_no	Dynamic hashing may be preferable over heap files in scenarios where there is a need for adaptability to dynamic changes in data size and a desire for efficient random search operations based on a
	(5) Random Search (update)	process_account_no	10/day		
Cost	(8) Insertion		50/day	Dynamic Hashing with a hash key assembly_account_no	Dynamic hashing is chosen because random search is more frequent on assembly_account_no and particularly effective for random search with a hash key due to its ability to adapt dynamically to varying amounts of data and distribute it evenly across buckets.
	(9) Random Search	assembly_account_no	200/day		

Task 3.2: STUDY OF RELEVANT DOCUMENTATION

Some other possible indexes I could understand are:

Filtered Indexes:

Relevant for queries involving specific subsets of data. For example, if we frequently query a subset of records based on certain conditions, a filtered index on the relevant columns might improve performance.

Columnstore Indexes:

Suitable for analytical queries that involve aggregating large amounts of data. If we have analytics queries for reporting purposes, a columnstore index on relevant columns can significantly improve query performance.

Memory-Optimized Tables and Indexes:

Suitable for high transaction throughput and low-latency scenarios. If our workload involves frequent insertions and updates, especially for transactional data, memory-optimized tables and indexes can provide improved performance.

Azure SQL Database Hyperscale:

Relevant for very large databases with horizontal scaling requirements. If our dataset grows significantly, Azure SQL Database Hyperscale can offer automatic scaling and support for large volumes of data.

In-Memory OLTP:

Suitable for high-performance transaction processing. If you have queries that require low-latency access to transactional data, especially in scenarios with high concurrency, In-Memory OLTP can be beneficial.

JSON Indexing:

Relevant if your data includes JSON documents and you frequently query JSON data. JSON indexing can improve the performance of queries involving JSON data structures.

Full-Text Indexes:

Suitable for efficient searching of large amounts of text-based data. If you have queries that involve text search operations, full-text indexes can enhance the search performance.

Spatial Indexing:

Relevant if your application deals with spatial data. Spatial indexing can optimize queries related to geographical or geometric data, enhancing the performance of spatial queries.

Auto-Create Statistics and Auto-Update Statistics:

While not indexes, these features are crucial for the query optimizer to generate efficient execution plans. They play a significant role in optimizing query performance by ensuring that the query planner has up-to-date statistics for accurate decision-making.

Bitmapped Index:

In certain scenarios, especially when dealing with categorical data or when queries involve multiple boolean conditions, a bitmapped index can be considered. Bitmapped indexes use bitmaps to represent the presence or absence of values in a column, enabling fast searches for combinations of conditions.

Multilevel Index:

In scenarios where the dataset is large and a single-level index becomes impractical, a multilevel index can be considered. This hierarchical structure allows for more efficient navigation through the index.

Covering Index:

A covering index includes all the columns required to satisfy a query, eliminating the need to access the actual data rows. This can be beneficial for certain queries, especially when dealing with a specific subset of columns.

GiST (Generalized Search Tree):

In databases that support complex data types (e.g., geometric or geographic data), GiST indexing can be considered. GiST allows indexing of complex data structures beyond simple key-value pairs.

BRIN (Block Range INdex):

BRIN indexes are suitable for very large tables with naturally ordered data. They divide the table into blocks and create an index on the minimum and maximum values within each block.

Even though we can’t use file structures like "Index Sequential File" and "Heap File" directly in Azure SQL Database, we do have significant influence over how tables are organized and indexed. By strategically setting up clustered indexes, primary keys, and other indexing features like secondary index, we can steer the database engine to work best for the kinds of queries you're running. It's like customizing the filing system to match exactly how you want to look up information.

Since the clustered indexes are by default in Azure, the non-clustered index I included are:

Table	Secondary Index
Customer	category (because of range search on it)
Process	department_no (because of random search on it)
Job	date_of_completion (because of random search on it)
assembly_expenditure	assembly_id (because of random search on it)
manufacture	job_no (because of random search update on it)
manufacture	assembly_id (because of random search on it)

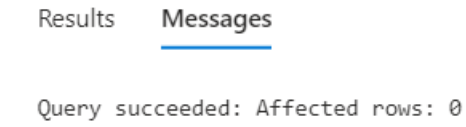
Task 4: SQL statements and screenshots showing the creation of tables in Azure SQL Database

1. Create Customer table:

```
-- Create Customer table
CREATE TABLE Customer (
cname VARCHAR(20) PRIMARY KEY,
caddress VARCHAR(100),
category INT NOT,
CONSTRAINT category_ck CHECK (category BETWEEN 1 AND 10),
);
```

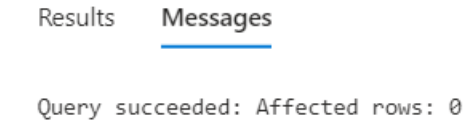
Creating Index on category:

```
CREATE INDEX customer_category ON Customer(category);
```



2. Create Assembly table:

```
-- Create Assembly table
CREATE TABLE Assembly (
assembly_id VARCHAR(20) PRIMARY KEY,
date_ordered DATE,
assembly_details VARCHAR(100),
cname VARCHAR(20) FOREIGN KEY REFERENCES Customer(cname)
);
```



3. Create Department table:

```
--Create Department Table
CREATE TABLE Department (
department_no INT PRIMARY KEY,
department_data VARCHAR(100),
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

4. Create Process table:

```
--Create Process Table
CREATE TABLE Process (
process_id VARCHAR(20) PRIMARY KEY,
process_data VARCHAR(100),
department_no INT,
FOREIGN KEY (department_no) REFERENCES Department(department_no)
);
```

Creating Index on department_no:

```
CREATE INDEX process_department_no ON Process(department_no);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

5. Create Fit_process table:

```
--Create Fit_process Table
CREATE TABLE Fit_process (
process_id VARCHAR(20) FOREIGN KEY REFERENCES Process(process_id),
fit_type VARCHAR(20),
PRIMARY KEY (process_id)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

6. Create Paint_process table:

```
--Create Paint_process Table
CREATE TABLE Paint_process (
process_id VARCHAR(20) FOREIGN KEY REFERENCES Process(process_id),
paint_type VARCHAR(20),
paint_method VARCHAR(20),
PRIMARY KEY (process_id)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

7. Create Cut_process table:

```
--Create Cut_process Table
CREATE TABLE Cut_process (
process_id VARCHAR(20) FOREIGN KEY REFERENCES Process(process_id),
cutting_type VARCHAR(20),
machine_type VARCHAR(20),
PRIMARY KEY (process_id)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

8. Create Job table:

```
--Create Job Table
CREATE TABLE Job (
 job_no INT PRIMARY KEY,
date_of_commencement DATE,
date_of_completion DATE
);
```

Creating an index on date_of_completion:

```
CREATE INDEX job_date_of_completion ON Job(date_of_completion);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

9. Create Fit_job table:

```
--Create Fit_job Table
CREATE TABLE Fit_job (
 job_no INT FOREIGN KEY REFERENCES Job(job_no),
fit_labor_time TIME,
PRIMARY KEY (job_no)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

10. Create Paint_job table:

```
--Create Paint_job Table
CREATE TABLE Paint_job (
 job_no INT FOREIGN KEY REFERENCES Job(job_no),
color VARCHAR(20),
volume FLOAT,
paint_labor_time TIME,
PRIMARY KEY (job_no)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

11. Create Cut_job table:

```
--Create Cut_job Table
CREATE TABLE Cut_job (
 job_no INT FOREIGN KEY REFERENCES Job(job_no),
job_machine_type VARCHAR(20),
machine_time TIME,
material_used VARCHAR(20),
cut_labor_time TIME,
PRIMARY KEY (job_no)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

12. Create manufacture table:

```
--Create manufacture Table
CREATE TABLE manufacture (
assembly_id VARCHAR(20) FOREIGN KEY REFERENCES Assembly(assembly_id),
process_id VARCHAR(20) FOREIGN KEY REFERENCES Process(process_id),
job_no INT FOREIGN KEY REFERENCES Job(job_no),
PRIMARY KEY (assembly_id, process_id)
);
```

Creating an index on assembly_id:

```
CREATE INDEX manufacture_assembly_id ON manufacture(assembly_id);
```

Creating an index on job_no:

```
CREATE INDEX manufacture_job_no ON manufacture(job_no);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

13. Create Assembly_account table:

```
--Create Assembly Account Table
CREATE TABLE Assembly_account (
assembly_account_no INT PRIMARY KEY,
date_of_establishment DATE,
details_1 NUMERIC (10, 2)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

14. Create Department_account table:

```
--Create Department Account Table
CREATE TABLE Department_account (
department_account_no INT PRIMARY KEY,
date_of_establishment DATE,
details_2 NUMERIC (10, 2)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

15. Create Process_account table:

```
--Create Process Account Table
CREATE TABLE Process_account (
process_account_no INT PRIMARY KEY,
date_of_establishment DATE,
details_3 NUMERIC (10, 2)
);
```

Results Messages

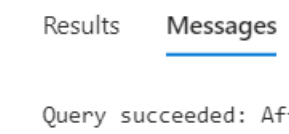
```
Query succeeded: Affected rows: 0
```

16. Create assembly_expenditure table:

```
-- Create Assembly Expenditure Table
CREATE TABLE assembly_expenditure (
    assembly_account_no INT,
    assembly_id VARCHAR(20),
    PRIMARY KEY (assembly_account_no),
    FOREIGN KEY (assembly_account_no) REFERENCES Assembly_account(assembly_account_no),
    FOREIGN KEY (assembly_id) REFERENCES Assembly(assembly_id)
);
```

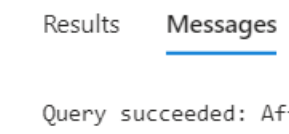
Creating an index on assembly_id:

```
CREATE INDEX assembly_expenditure_assembly_id ON assembly_expenditure(assembly_id);
```



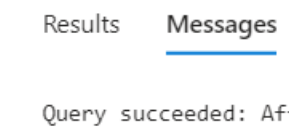
17. Creating department_expenditure table:

```
-- Create Department Expenditure Table
CREATE TABLE department_expenditure (
    department_account_no INT,
    department_no INT,
    PRIMARY KEY (department_account_no),
    FOREIGN KEY (department_account_no) REFERENCES Department_account(department_account_no),
    FOREIGN KEY (department_no) REFERENCES Department(department_no)
);
```



18. Creating process_expenditure table:

```
-- Create Process Expenditure Table
CREATE TABLE process_expenditure (
    process_account_no INT,
    process_id VARCHAR(20),
    PRIMARY KEY (process_account_no),
    FOREIGN KEY (process_account_no) REFERENCES Process_account(process_account_no),
    FOREIGN KEY (process_id) REFERENCES Process(process_id)
);
```



19. Create Cost table:

```
-- Create Cost Table
CREATE TABLE Cost (
    transaction_no INT PRIMARY KEY,
    sup_cost NUMERIC(10, 2),
    assembly_account_no INT,
    department_account_no INT,
    process_account_no INT,
    FOREIGN KEY (assembly_account_no) REFERENCES Assembly_account(assembly_account_no),
    FOREIGN KEY (department_account_no) REFERENCES Department_account(department_account_no),
    FOREIGN KEY (process_account_no) REFERENCES Process_account(process_account_no)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

20. Create proceed table:

```
--Create proceed Table
CREATE TABLE proceed (
transaction_no INT FOREIGN KEY REFERENCES Cost(transaction_no),
job_no INT FOREIGN KEY REFERENCES Job(job_no),
PRIMARY KEY (transaction_no)
);
```

Results Messages

```
Query succeeded: Affected rows: 0
```

Task 5:

Task 5.1 SQL statements and Transact SQL stored procedures

1. Enter a new Customer

Enter a new Customer procedure:

```
CREATE PROCEDURE EnterNewCustomer
    @cname VARCHAR(20),
    @caddress VARCHAR(100),
    @category INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@cname) > 20
        BEGIN
            -- If the name is too long
            SET @errorCode = 50001;
            SET @errorMessage = 'Invalid customer name. Name must be up to 20 characters.';
            RETURN;
        END

         IF @cname LIKE '%[0-9!@#$%^&*(),.?":{}|<>]%'
        BEGIN
            -- If the name contains special characters, raise an error
            SET @errorCode = 50002;
            SET @errorMessage = 'Invalid customer name. Name must be without special characters or numbers.';
            RETURN;
        END

        -- Check if the address is null
        IF LEN(@caddress) = 0
```

```
          -- Check if the address is null
          IF LEN(@caddress) = 0
          BEGIN
              SET @errorCode = 50003;
              SET @errorMessage = 'Invalid customer address. Address must be provided.';
              RETURN;
          END
          -- Check if the address is too long
           IF LEN(@caddress) > 100
          BEGIN
              SET @errorCode = 50004;
              SET @errorMessage = 'Invalid customer address. Address must be up to 100 characters.';
              RETURN;
          END

          IF @category NOT BETWEEN 1 AND 10
          BEGIN
              -- If the category is in the specified range, raise an error
              SET @errorCode = 50005;
              SET @errorMessage = 'Invalid customer category. Category must be an integer between 1 and 10.';
              RETURN;
          END

          -- Data manipulation
          BEGIN
              INSERT INTO Customer (cname, caddress, category)
              VALUES (@cname, @caddress, @category);
          END;
```

```
        END;

    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();

        -- Check if the error is related to a primary key or unique constraint violation
        IF @errorCode = 2627
        BEGIN
            SET @errorMessage = 'A customer with the same name or data already exists.';
        END
        ELSE
        BEGIN
            SET @errorMessage = ERROR_MESSAGE();
        END

        RETURN;
    END CATCH;
END;
```

2. Enter a new department

Enter a new department procedure:

```
CREATE PROCEDURE EnterNewDepartment
    @department_no INT,
    @department_data VARCHAR(100),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF @department_no <= 0
        BEGIN
            -- If the department number is not positive, raise an error
            SET @errorCode = 50006;
            SET @errorMessage = 'Invalid department number. Department number must be a positive integer.';
            RETURN;
        END

        IF LEN(@department_data) > 100
        BEGIN
            -- If the department data is too long, raise an error
            SET @errorCode = 50007;
            SET @errorMessage = 'Invalid department data. Data must be up to 100 characters.';
            RETURN;
        END

        -- Data manipulation
        BEGIN
            INSERT INTO Department (department_no, department_data)
```

```
            RETURN;
        END

        -- Data manipulation
        BEGIN
            INSERT INTO Department (department_no, department_data)
            VALUES (@department_no, @department_data);
        END;


    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();

        -- Check if the error is related to a unique key violation
        IF @errorCode = 2627
        BEGIN
            SET @errorMessage = 'Department number already exists. Please choose a unique department number.';
        END
        ELSE
        BEGIN
            SET @errorMessage = ERROR_MESSAGE();
        END

        RETURN;
    END CATCH;
END;
```

3. Enter a new process_id and its department together with its type and information relevant to the type

Enter new process procedure:

```
CREATE PROCEDURE EnterNewProcess
    @process_id VARCHAR(20),
    @process_data VARCHAR(100),
    @department_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@process_data) = 0 OR LEN(@process_data) > 100
        BEGIN
            SET @errorCode = 50008;
            SET @errorMessage = 'Invaid process data. Process data must be provided upto 100 characters.';
            RETURN;
        END

        IF NOT EXISTS (SELECT 1 FROM Department WHERE department_no = @department_no)
        BEGIN
            SET @errorCode = 50009;
            SET @errorMessage = 'Department does not exist. Please provide a valid department number.';
            RETURN;
        END


        -- Data manipulation
        INSERT INTO Process (process_id, process_data, department_no)
        VALUES (@process_id, @process_data, @department_no);
```

```
        -- Data manipulation
        INSERT INTO Process (process_id, process_data, department_no)
        VALUES (@process_id, @process_data, @department_no);


    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();

        RETURN;
    END CATCH;
END;
```

3.1: Enter a fit process procedure:

```
CREATE PROCEDURE EnterNewFitProcess
    @process_id VARCHAR(20),
    @fit_type VARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@fit_type) = 0 OR LEN(@fit_type) > 20
        BEGIN
            SET @errorCode = 50010;
            SET @errorMessage = 'Invalid input. Fit Type cannot be empty and it must be up to 20 characters.';
            RETURN;
        END

        -- Data manipulation
        INSERT INTO Fit_Process (process_id, fit_type)
        VALUES (@process_id, @fit_type);

    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
        RETURN;
    END CATCH;
END;
```

3.2: Enter a paint process procedure:

```
CREATE PROCEDURE EnterNewPaintProcess
    @process_id VARCHAR(20),
    @paint_type VARCHAR(20),
    @paint_method VARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@paint_type) = 0 OR LEN(@paint_type) > 20
        BEGIN
            SET @errorCode = 50011;
            SET @errorMessage = 'Invalid input. Paint Type cannot be empty and must be up to 20 characters.';
            RETURN;
        END

        IF LEN(@paint_method) = 0 OR LEN(@paint_method) > 20
        BEGIN
            SET @errorCode = 50012;
            SET @errorMessage = 'Invalid input. Paint Method cannot be empty and must be up to 20 characters.';
            RETURN;
        END
        -- Data manipulation
        INSERT INTO Paint_Process (process_id, paint_type, paint_method)
        VALUES (@process_id, @paint_type, @paint_method);
    END TRY
    BEGIN CATCH
        END TRY
        BEGIN CATCH
            -- Handle errors
            SET @errorCode = ERROR_NUMBER();
            SET @errorMessage = ERROR_MESSAGE();
            RETURN;
        END CATCH;
    END;
```

3.3: Enter a cut process procedure:

```sql
CREATE PROCEDURE EnterNewCutProcess
    @process_id VARCHAR(20),
    @cutting_type VARCHAR(20),
    @machine_type VARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@cutting_type) = 0 OR LEN(@cutting_type) > 20
        BEGIN
            SET @errorCode = 50013;
            SET @errorMessage = 'Invalid input. Cutting Type cannot be empty and must be up to 20 characters.';
            RETURN;
        END

         IF LEN(@machine_type) = 0 OR LEN(@machine_type) > 20
        BEGIN
            SET @errorCode = 50014;
            SET @errorMessage = 'Invalid input. Machine Type cannot be empty and must be up to 20 characters.';
            RETURN;
        END

        -- Data manipulation
        INSERT INTO Cut_Process (process_id, cutting_type, machine_type)
        VALUES (@process_id, @cutting_type, @machine_type);


    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
        RETURN;
    END CATCH;
END;
```

4. Enter a new assembly wit its customer_name, assembly_details, assembly_id and date ordered and associate it with one or more processes

4.1. Enter a new Assembly procedure:

```sql
CREATE PROCEDURE EnterNewAssembly
    @assembly_id VARCHAR(20),
    @date_ordered DATE,
    @assembly_details VARCHAR(100),
    @cname VARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Data validation
        IF LEN(@assembly_details) = 0 OR LEN(@assembly_details) > 100
        BEGIN
            SET @errorCode = 50015;
            SET @errorMessage = 'Invalid assembly details. Assembly details must be provided up to 100 characters.';
            RETURN;
        END

        IF NOT EXISTS (SELECT 1 FROM Customer WHERE cname = @cname)
        BEGIN
            SET @errorCode = 50016;
            SET @errorMessage = 'Customer does not exist. Please provide a valid customer name.';
            RETURN;
        END

        -- Data manipulation
        INSERT INTO Assembly (assembly_id, date_ordered, assembly_details, cname)
        VALUES (@assembly_id, @date_ordered, @assembly_details, @cname);
        -- Data manipulation
        INSERT INTO Assembly (assembly_id, date_ordered, assembly_details, cname)
        VALUES (@assembly_id, @date_ordered, @assembly_details, @cname);

    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();

        RETURN;
    END CATCH;
END;
```

4.2: Enter a new manufacture procedure:

```sql
CREATE PROCEDURE EnterNewManufacture
    @assembly_id VARCHAR(20),
    @process_id VARCHAR(20),
    @job_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Check if the combination of assembly_id and process_id already exists
        IF EXISTS (SELECT 1 FROM manufacture WHERE assembly_id = @assembly_id AND process_id = @process_id)
        BEGIN
            SET @errorCode = 50017;
            SET @errorMessage = 'Combination of assembly_id and process_id already exists. Please provide a unique combination.';
            RETURN;
        END

        -- Data manipulation
        INSERT INTO manufacture (assembly_id, process_id, job_no)
        VALUES (@assembly_id, @process_id, @job_no);
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
        RETURN;
    END CATCH;
END;
```

5. Create a new account and associate it with the process, assembly, or department to which it is applicable

5.1: Enter a new assembly account Procedure:

```sql
-- Procedure for Assembly_account
CREATE PROCEDURE InsertAssemblyAccount
    @account_no INT,
    @date_of_establishment DATE,
    @details_1 NUMERIC(10,2),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Assembly_account table
        INSERT INTO Assembly_account (assembly_account_no, date_of_establishment, details_1)
        VALUES (@account_no, @date_of_establishment, @details_1);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Assembly_account record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50018; -- Custom error code for assembly account
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

5.2: Enter a new department account procedure:

```sql
-- Procedure for Department_account
CREATE PROCEDURE InsertDepartmentAccount
    @account_no INT,
    @date_of_establishment DATE,
    @details_2 NUMERIC(10, 2),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Department_account table
        INSERT INTO Department_account (department_account_no, date_of_establishment, details_2)
        VALUES (@account_no, @date_of_establishment, @details_2);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Department_account record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50019; -- Custom error code for department account
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

5.3: Enter a new process account procedure:

```sql
-- Procedure for Process_account
CREATE PROCEDURE InsertProcessAccount
    @account_no INT,
    @date_of_establishment DATE,
    @details_3 NUMERIC(10, 2),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Process_account table
        INSERT INTO Process_account (process_account_no, date_of_establishment, details_3)
        VALUES (@account_no, @date_of_establishment, @details_3);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Process_account record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50020; -- Custom error code for process account
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

5.4: Enter a new assembly expenditure:

```sql
-- Procedure for Assembly_expenditure
CREATE PROCEDURE InsertAssemblyExpenditure
    @assembly_account_no INT,
    @assembly_id NVARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Assembly_expenditure table
        INSERT INTO Assembly_expenditure (assembly_account_no, assembly_id)
        VALUES (@assembly_account_no, @assembly_id);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Assembly_expenditure record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50021; -- Custom error code for assembly expenditure
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

5.5: Enter a new department expenditure procedure:

```sql
-- Procedure for Department_expenditure
CREATE PROCEDURE InsertDepartmentExpenditure
    @department_account_no INT,
    @department_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Department_expenditure table
        INSERT INTO Department_expenditure (department_account_no, department_no)
        VALUES (@department_account_no, @department_no);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Department_expenditure record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50022; -- Custom error code for department expenditure
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

5.6: Enter a new process expenditure procedure:

```sql
-- Procedure for Process_expenditure
CREATE PROCEDURE InsertProcessExpenditure
    @process_account_no INT,
    @process_id NVARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into Process_expenditure table
        INSERT INTO Process_expenditure (process_account_no, process_id)
        VALUES (@process_account_no, @process_id);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Process_expenditure record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = 50023; -- Custom error code for process expenditure
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

6. Enter a new job, given its job_no, assembly_id, process_id and date the job commenced

6.1: Enter a new job procedure:

```sql
-- Procedure for entering a new job
CREATE PROCEDURE EnterNewJob
    @job_no INT,
    @date_of_commencement DATE,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into the Job table
        INSERT INTO Job (job_no, date_of_commencement)
        VALUES (@job_no, @date_of_commencement);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Job record inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

6.2: Enter a fit job procedure:

```sql
--Procedure for inserting details into Fit_job
CREATE PROCEDURE InsertFitJobDetails
    @job_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into the Fit_job table
        INSERT INTO Fit_job (job_no)
        VALUES (@job_no);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Fit_job details inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

6.3: Enter a paint job procedure:

```sql
-- Procedure for inserting details into Paint_job
CREATE PROCEDURE InsertPaintJobDetails
    @job_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into the Paint_job table
        INSERT INTO Paint_job (job_no)
        VALUES (@job_no);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Paint_job details inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

6.4: Enter a cut job procedure:

```sql
--Procedure for inserting details into Cut_job
CREATE PROCEDURE InsertCutJobDetails
    @job_no INT,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(4000) OUTPUT
AS
BEGIN
    BEGIN TRY
        -- Insert the record into the Cut_job table
        INSERT INTO Cut_job (job_no)
        VALUES (@job_no);

        -- Success
        SET @errorCode = 0;
        SET @errorMessage = 'Cut_job details inserted successfully.';
    END TRY
    BEGIN CATCH
        -- Handle errors
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH;
END;
```

6.5: Enter job in manufacture procedure:

```sql
CREATE PROCEDURE InsertManufactureJob (
    @jobNumber INT,
    @assemblyId VARCHAR(20),
    @processId VARCHAR(20),
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(255) OUTPUT
)
AS
BEGIN
    -- Check if the pair is legit existing in the manufacture table
    IF NOT EXISTS (
        SELECT 1
        FROM manufacture
        WHERE assembly_id = @assemblyId
            AND process_id = @processId
    )
    BEGIN
        -- Set error code and message for an invalid pair
        SET @errorCode = 1;
        SET @errorMessage = 'Invalid pair. This assembly_id and process_id combination does not exist in the manufacture table.';
    END
    ELSE
    BEGIN
        -- Update the manufacture table to associate the new job_no with the existing pair
        UPDATE manufacture
        SET job_no = @jobNumber
        WHERE assembly_id = @assemblyId
            AND process_id = @processId;
```

7. At the completion of a job, enter the date it completed and the information relevant to the type of job

7.1. Check for job procedure:

```sql
-- Procedure to check if a job with the specified job number exists
CREATE PROCEDURE CheckJobExistence (
    @jobNumber INT,
    @existsFlag INT OUTPUT
)
AS
BEGIN
    -- Check if the job number exists in the Job table
    IF EXISTS (
        SELECT 1
        FROM Job
        WHERE job_no = @jobNumber
    )
    BEGIN
        -- Set existsFlag to 1 if the job exists
        SET @existsFlag = 1;
    END
    ELSE
    BEGIN
        -- Set existsFlag to 0 if the job does not exist
        SET @existsFlag = 0;
    END
END;

```

7.2: Check job type procedure:

```sql
-- Procedure to determine the type of job (Fit, Paint, or Cut) based on the job number
CREATE PROCEDURE DetermineJob (
    @jobNumber INT,
    @jobType NVARCHAR(20) OUTPUT
)
AS
BEGIN
    -- Determine the job type based on the presence in each table
    IF EXISTS (
        SELECT 1
        FROM Fit_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        SET @jobType = 'Fit Job';
    END
    ELSE IF EXISTS (
        SELECT 1
        FROM Paint_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        SET @jobType = 'Paint Job';
    END
    ELSE IF EXISTS (
        SELECT 1
        FROM Cut_job
        WHERE job_no = @jobNumber
```

```
        SELECT 1
        FROM Cut_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        SET @jobType = 'Cut Job';
    END
    ELSE
    BEGIN
        -- Set a default value if the job type cannot be determined
        SET @jobType = 'Unknown Job';
    END
END;
```

7.3: Enter fit information procedure:

```
-- Procedure to update Fit Job labor time
CREATE PROCEDURE UpdateFitJobLaborTime (
    @jobNumber INT,
    @fitLaborTime TIME,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(255) OUTPUT
)
AS
BEGIN
    -- Check if the job number exists in the Fit Job table
    IF EXISTS (
        SELECT 1
        FROM Fit_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        -- Update the Fit Job table with the new fit_labor_time
        UPDATE Fit_job
        SET fit_labor_time = @fitLaborTime
        WHERE job_no = @jobNumber;

        -- Set success code and message
        SET @errorCode = 0;
        SET @errorMessage = 'Fit Job labor time updated successfully.';
    END
    ELSE
    BEGIN
        -- Set error code and message for job not found
    END
    ELSE
    BEGIN
        -- Set error code and message for job not found
        SET @errorCode = 1;
        SET @errorMessage = 'Fit Job with the specified job number does not exist.';
    END
END;
```

7.4: Enter paint job information procedure:

```
-- Procedure to update Paint Job details
CREATE PROCEDURE UpdatePaintJobDetails (
    @jobNumber INT,
    @color VARCHAR(20),
    @volume DOUBLE,
    @paintLaborTime TIME,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(255) OUTPUT
)
AS
BEGIN
    -- Check if the job number exists in the Paint Job table
    IF EXISTS (
        SELECT 1
        FROM Paint_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        -- Update Paint Job details
        UPDATE Paint_job
        SET color = @color,
            volume = @volume,
            paint_labor_time = @paintLaborTime
        WHERE job_no = @jobNumber;

        -- Set success code and message
        SET @errorCode = 0;
        SET @errorMessage = 'Paint Job details updated successfully.';
        SET @errorCode = 0;
        SET @errorMessage = 'Paint Job details updated successfully.';
    END
    ELSE
    BEGIN
        -- Set error code and message if job number does not exist
        SET @errorCode = 1;
        SET @errorMessage = 'Error: Job with the specified job number does not exist in Paint Job table.';
    END
END;
```

7.5: Enter cut job information procedure:

```sql
-- Procedure to update Cut Job details
CREATE PROCEDURE UpdateCutJobDetails (
    @jobNumber INT,
    @machineType VARCHAR(20),
    @machineTime TIME,
    @materialUsed VARCHAR(50),
    @cutLaborTime TIME,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(255) OUTPUT
)
AS
BEGIN
    -- Check if the job number exists in the Cut Job table
    IF EXISTS (
        SELECT 1
        FROM Cut_job
        WHERE job_no = @jobNumber
    )
    BEGIN
        -- Update Cut Job details
        UPDATE Cut_job
        SET type_of_machine = @machineType,
            time_of_machine = @machineTime,
            material_used = @materialUsed,
            cut_labor_time = @cutLaborTime
        WHERE job_no = @jobNumber;

        -- Set success code and message
```

7.6: Check date of completion procedure:

```sql
CREATE PROCEDURE CheckDateCompletionGreaterThanCommencement (
    @jobNumber INT,
    @dateCompleted DATE,
    @isDateValid BIT OUTPUT
)
AS
BEGIN
    DECLARE @dateCommenced DATE;

    -- Retrieve the date of commencement from the job table
    SELECT @dateCommenced = date_of_commencement
    FROM job
    WHERE job_no = @jobNumber;

    -- Check if the date of completion is greater than the date of commencement
    IF @dateCompleted > @dateCommenced
    BEGIN
        SET @isDateValid = 1; -- Date is valid
    END
    ELSE
    BEGIN
        SET @isDateValid = 0; -- Date is not valid
    END
END;
```

7.7: Enter date of completion procedure:

```sql
CREATE PROCEDURE UpdateCompletionDate
    @jobNumber INT,
    @newCompletionDate DATE,
    @errorCode INT OUTPUT,
    @errorMessage NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        -- Validate that the new completion date is greater than the date of commencement
        IF EXISTS (
            SELECT 1
            FROM job
            WHERE job_no = @jobNumber
                AND @newCompletionDate <= date_of_commencement
        )
        BEGIN
            SET @errorCode = 1; -- You can define your error codes
            SET @errorMessage = 'Error: Date of completion must be greater than the date of commencement.';
        END
        ELSE
        BEGIN
            -- Update the date_of_completion
            UPDATE job
            SET date_of_completion = @newCompletionDate
            WHERE job_no = @jobNumber;

            SET @errorCode = 0;
            SET @errorMessage = 'Date of completion updated successfully.';
        END
    END TRY
    BEGIN CATCH
        SET @errorCode = ERROR_NUMBER();
        SET @errorMessage = ERROR_MESSAGE();
    END CATCH
END;

```

8. Enter a transaction_no and its sup-cost and update all the costs of the affected accounts by adding sup-cost to their current values of details

8.1. Enter a new transaction procedure:

```sql
CREATE PROCEDURE EnterNewTransaction
    @transaction_no INT,
    @sup_cost NUMERIC(10, 2),
    @assembly_account_no INT,
    @department_account_no INT,
    @process_account_no INT
AS
BEGIN
    -- Update Assembly_account details
    UPDATE Assembly_account
    SET details_1 = details_1 + (@sup_cost)
    WHERE assembly_account_no = @assembly_account_no;

    -- Update Department_account details
    UPDATE Department_account
    SET details_2 = details_2 + (@sup_cost)
    WHERE department_account_no = @department_account_no;

    -- Update Process_account details
    UPDATE Process_account
    SET details_3 = details_3 + (@sup_cost)
    WHERE process_account_no = @process_account_no;

    -- Insert into Cost table
    INSERT INTO Cost (transaction_no, sup_cost, assembly_account_no, department_account_no, process_account_no)
    VALUES (@transaction_no, @sup_cost, @assembly_account_no, @department_account_no, @process_account_no);
END;
```

8.1 Get total code procedure:

```sql
CREATE PROCEDURE GetTotalCost
    @assembly_id VARCHAR(20),
    @total_cost NUMERIC(10, 2) OUTPUT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        SELECT @total_cost = ISNULL(SUM(c.sup_cost), 0)
        FROM Cost c
        WHERE c.assembly_account_no IN (
            SELECT aea.assembly_account_no
            FROM assembly_expenditure aea
            WHERE aea.assembly_id = @assembly_id
        );

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @total_cost = 0;
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

9. Retrieve the total cost incurred on an assembly-id
   Total cost procedure:

```sql
CREATE PROCEDURE GetTotalCost
    @assembly_id VARCHAR(20),
    @total_cost NUMERIC(10, 2) OUTPUT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        SELECT @total_cost = ISNULL(SUM(c.sup_cost), 0)
        FROM Cost c
        WHERE c.assembly_account_no IN (
            SELECT aea.assembly_account_no
            FROM assembly_expenditure aea
            WHERE aea.assembly_id = @assembly_id
        );

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @total_cost = 0;
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

10. Retrieve the total labor time within a department for jobs completed in the department during a given date

```sql
CREATE PROCEDURE GetTotalLaborTimeInDepartment
    @department_no INT,
    @given_date DATE,
    @total_labor_time FLOAT OUTPUT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        SELECT @total_labor_time = ISNULL(SUM(j.fit_labor_time + j.paint_labor_time + cj.cut_labor_time), 0)
        FROM Job j
        LEFT JOIN Fit_job fj ON j.job_no = fj.job_no
        LEFT JOIN Paint_job pj ON j.job_no = pj.job_no
        LEFT JOIN Cut_job cj ON j.job_no = cj.job_no
        WHERE j.date_of_completion = @given_date
            AND EXISTS (
                SELECT 1
                FROM manufacture m
                INNER JOIN Process p ON m.process_id = p.process_id
                WHERE m.job_no = j.job_no
                    AND p.department_no = @department_no
            );

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @total_labor_time = 0;
    END TRY
    BEGIN CATCH
        SET @total_labor_time = 0;
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

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

```sql
CREATE PROCEDURE GetProcessesForAssembly
    @assembly_id VARCHAR(20),
    @process_details NVARCHAR(MAX) OUTPUT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        SELECT @process_details = COALESCE(@process_details + ', ', '') +
                                  p.process_data + ' (Department: ' + CONVERT(NVARCHAR(10), p.department_no) + ')'
        FROM manufacture m
        INNER JOIN Process p ON m.process_id = p.process_id
        INNER JOIN Job j ON m.job_no = j.job_no
        WHERE m.assembly_id = @assembly_id
        ORDER BY j.date_of_completion;

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @process_details = NULL;
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

12. Retrieve the customers whose category is in the given range

```sql
CREATE PROCEDURE GetCustomersInRange
    @min_category INT,
    @max_category INT,
    @customer_details NVARCHAR(MAX) OUTPUT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        SELECT @customer_details = STRING_AGG(cname, ', ')
            FROM Customer
            WHERE category BETWEEN @min_category AND @max_category
            ORDER BY cname;

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @customer_details = NULL;
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

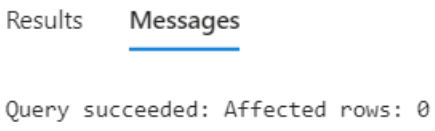
13. Delete all cut-jobs whose job_no is in given range

```sql
CREATE PROCEDURE DeleteCutJobsInRange
    @min_job_no INT,
    @max_job_no INT,
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        DELETE FROM Cut_job
        WHERE job_no BETWEEN @min_job_no AND @max_job_no;

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

14. Change the color of a given paint job

```sql
CREATE PROCEDURE ChangePaintJobColor
    @job_number INT,
    @new_color VARCHAR(20),
    @error_message NVARCHAR(MAX) OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        UPDATE Paint_job
        SET color = @new_color
        WHERE job_no = @job_number;

        SET @error_message = NULL;
    END TRY
    BEGIN CATCH
        SET @error_message = ERROR_MESSAGE();
    END CATCH;
END;
```

Output of all Procedures:



Task 5.2: Java application program that uses JDBC and Azure Database

1. Enter a new customer (30/day).

```
185         case "1":
186             System.out.println("Please enter customer name:");
187             // Consume any remaining newline characters
188             sc.nextLine();
189             // Read customer name
190             final String cname = sc.nextLine();
191
192             System.out.println("Please enter customer address:");
193             // Read customer address
194             final String caddress = sc.nextLine();
195
196             System.out.println("Please enter customer category:");
197             final int category;
198             try {
199                 // Read and parse category as an integer
200                 category = Integer.parseInt(sc.nextLine().trim());
201             } catch (NumberFormatException e) {
202                 // Handle invalid input for category
203                 System.out.println("Error Message: Invalid input for category. Please enter a valid integer.");
204                 break;  // Terminate the loop or take appropriate action
205             }
206
207             System.out.println("Connecting to the database...");
208             try (Connection connection = DriverManager.getConnection(URL)) {
209                 try (CallableStatement statement = connection.prepareCall(ENTER_NEW_CUSTOMER)) {
210                     // Set parameters for the stored procedure
211                     statement.setString(1, cname);
212                     statement.setString(2, caddress);
213                     statement.setInt(3, category);
214
215                     // Register output parameters for error code and message
216                     statement.registerOutParameter(4, Types.INTEGER);
217                     statement.registerOutParameter(5, Types.NVARCHAR);
218
219                     System.out.println("Dispatching the stored procedure...");
220                     // Execute the stored procedure
221                     statement.execute();
222
223                     // Retrieve error code and message from the stored procedure
224                     int aerrorCode = statement.getInt(4);
225                     String aerrorMessage = statement.getString(5);
226
227                     if (aerrorCode != 0) {
228                         // Error occurred, display error message
229                         System.out.println("Error Code: " + aerrorCode);
230                         System.out.println("Error Message: " + aerrorMessage);
231                         System.out.println("Failed to add customer. Returning to the main menu.");
232                     } else {
233                         // No error, success
234                         System.out.println("Done. Customer added.");
235                     }
236                 } catch (SQLException e) {
237                     // Handle SQL exceptions during stored procedure execution
238                     System.out.println("Error: " + e.getMessage());
239                     System.out.println("Failed to add customer. Returning to the main menu.");
240                 }
241             } catch (SQLException e) {
242                 // Handle connection errors
243                 System.out.println("Connection error: " + e.getMessage());
244             }
245             break;
```

2. Enter a new department (infrequent).

```
248             case "2":
249                 // Prompt the user to enter the department number.
250                 System.out.println("Please enter department number:");
251                 // Read the entered department number.
252                 int department_no = sc.nextInt();
253
254                 // Consume the newline character left by previous sc.nextInt().
255                 sc.nextLine();
256
257                 // Prompt the user to enter department data.
258                 System.out.println("Please enter department data:");
259                 // Read the entered department data.
260                 String department_data = sc.nextLine();
261
262                 // Inform the user about the database connection process.
263                 System.out.println("Connecting to the database...");
264
265                 try (Connection connection = DriverManager.getConnection(URL)) {
266                     // Prepare the stored procedure call to add a new department.
267                     try (CallableStatement statement = connection.prepareCall(ENTER_NEW_DEPARTMENT)) {
268                         // Set the input parameters for the stored procedure.
269                         statement.setInt(1, department_no);
270                         statement.setString(2, department_data);
271
272                         // Register output parameters for error code and message.
273                         statement.registerOutParameter(3, Types.INTEGER);
274                         statement.registerOutParameter(4, Types.NVARCHAR);
275
276                         // Inform the user about dispatching the stored procedure.
277                         System.out.println("Dispatching the stored procedure...");
278                         // Execute the stored procedure.
279                         statement.execute();
280
281                         // Retrieve error code and message from the stored procedure.
282                         int berrorCode = statement.getInt(3);
283                         String berrorMessage = statement.getString(4);
284
285                         if (berrorCode != 0) {
286                             // Error occurred, display error message.
287                             System.out.println("Error Code: " + berrorCode);
288                             System.out.println("Error Message: " + berrorMessage);
289                             System.out.println("Failed to add department. Returning to the main menu.");
290                         } else {
291                             // No error, success.
292                             System.out.println("Done. Department added.");
293                         }
294                     } catch (SQLException e) {
295                         // Handle SQL exceptions.
296                         System.out.println("Error: " + e.getMessage());
297                         System.out.println("Failed to add department. Returning to the main menu.");
298                     }
299                 } catch (SQLException e) {
300                     // Handle connection-related exceptions.
301                     System.out.println("Connection error: " + e.getMessage());
302                 }
303                 // Exit the case statement for adding a department.
304                 break;
```

3. Enter a new process-id and its department together with its type and information relevant to the type (infrequent).

```
306         // Case "3" represents the user's choice to add a new process.
307         case "3":
308                 System.out.println("Please enter process ID:");
309                 sc.nextLine();
310                 // read process id
311                 String process_id = sc.nextLine();
312
313                 System.out.println("Please enter process data:");
314                 // read process data
315                 String process_data = sc.nextLine();
316
317                 System.out.println("Please enter department number:");
318
319                 final int department_noc;
320                 try {
321                     department_noc = Integer.parseInt(sc.nextLine().trim());
322                 } catch (NumberFormatException e) {
323                     System.out.println("Error Message: Invalid input for department number. Please enter a valid integer.");
324                     break; // Terminate the loop or take appropriate action
325                 }
326
327                 System.out.println("Connecting to the database...");
328
329                 try (Connection connection = DriverManager.getConnection(URL)) {
330                     try (CallableStatement statement = connection.prepareCall(ENTER_NEW_PROCESS)) {
331                         // Set parameters for the main process
332                         statement.setString(1, process_id);
333                         statement.setString(2, process_data);
334                         statement.setInt(3, department_noc);
335
336                         // Register output parameters for error code and message
337                         statement.registerOutParameter(4, Types.INTEGER);
338                         statement.registerOutParameter(5, Types.NVARCHAR);
339
340                         System.out.println("Dispatching the stored procedure for Process...");
341                         statement.execute();
342
343                         // Retrieve error code and message from the stored procedure
344                         int cerrorCode = statement.getInt(4);
345                         String cerrorMessage = statement.getString(5);
346
347                         if (cerrorCode != 0) {
348                             // Error occurred, display error message
349                             System.out.println("Error Code: " + cerrorCode);
350                             System.out.println("Error Message: " + cerrorMessage);
351                             System.out.println("Failed to add process. Returning to the main menu.");
352                         } else {
353                             // No error, success
354                             System.out.println("Done. Process added.");
355
356                             boolean processDetailsEnteredSuccessfully = false;
357
358                             while (!processDetailsEnteredSuccessfully) {
359                                 // Ask the user to choose the type of process to add
360                                 System.out.println("Choose the type of process to add:");
361                                 System.out.println("1. Fit Process");
362                                 System.out.println("2. Paint Process");
363                                 System.out.println("3. Cut Process");
364
365                                 int processTypeChoice = sc.nextInt();
366
```

```
                switch (processTypeChoice) {
                    case 1:
                        // Now, let's handle the Fit Process
                        System.out.println("Please enter Fit Type:");
                        sc.nextLine(); // Consume the newline character left by previous sc.nextInt()
                        String fit_type = sc.nextLine();

                        try (CallableStatement fitStatement = connection.prepareCall(ENTER_NEW_FIT_PROCESS)) {
                            // Set parameters for the Fit Process
                            fitStatement.setString(1, process_id);
                            fitStatement.setString(2, fit_type);

                            // Register output parameters for error code and message
                            fitStatement.registerOutParameter(3, Types.INTEGER);
                            fitStatement.registerOutParameter(4, Types.NVARCHAR);

                            System.out.println("Dispatching the stored procedure for Fit Process...");
                            fitStatement.execute();

                            // Retrieve error code and message from the stored procedure
                            int errorCodeFitProcess = fitStatement.getInt(3);
                            String errorMessageFitProcess = fitStatement.getString(4);

                            if (errorCodeFitProcess != 0) {
                                // Error occurred, display error message
                                System.out.println("Error Code: " + errorCodeFitProcess);
                                System.out.println("Error Message: " + errorMessageFitProcess);
                                System.out.println("Failed to add Fit Process. Please try again.");
                            } else {
                                // No error, success
                                System.out.println("Done. Fit Process added.");
                                processDetailsEnteredSuccessfully = true;
                            }
                        } catch (SQLException e) {
                            // Handle SQL exceptions
                            System.out.println("Error: " + e.getMessage());
                            System.out.println("Failed to add Fit Process. Returning to process type selection.");
                        }
                        break;

                    case 2:
                        // Now, let's handle the Paint Process
                        System.out.println("Please enter Paint Type:");
                        sc.nextLine(); // Consume the newline character left by previous sc.nextInt()
                        String paint_type = sc.nextLine();

                        System.out.println("Please enter Paint Method:");
                        String paint_method = sc.nextLine();

                        try (CallableStatement paintStatement = connection.prepareCall(ENTER_NEW_PAINT_PROCESS)) {
                            // Set parameters for the Paint Process
                            paintStatement.setString(1, process_id);
                            paintStatement.setString(2, paint_type);
                            paintStatement.setString(3, paint_method);

                            // Register output parameters for error code and message
                            paintStatement.registerOutParameter(4, Types.INTEGER);
                            paintStatement.registerOutParameter(5, Types.NVARCHAR);

                            System.out.println("Dispatching the stored procedure for Paint Process...");
                            paintStatement.execute();
```

```
                // Retrieve error code and message from the stored procedure
                int errorCodePaintProcess = paintStatement.getInt(4);
                String errorMessagePaintProcess = paintStatement.getString(5);

                if (errorCodePaintProcess != 0) {
                    // Error occurred, display error message
                    System.out.println("Error Code: " + errorCodePaintProcess);
                    System.out.println("Error Message: " + errorMessagePaintProcess);
                    System.out.println("Failed to add Paint Process. Please try again.");
                } else {
                    // No error, success
                    System.out.println("Done. Paint Process added.");
                    processDetailsEnteredSuccessfully = true;
                }
            } catch (SQLException e) {
                // Handle SQL exceptions
                System.out.println("Error: " + e.getMessage());
                System.out.println("Failed to add Paint Process. Returning to process type selection.");
            }
            break;

        case 3:
            // Now, let's handle the Cut Process
            System.out.println("Please enter Cutting Type:");
            sc.nextLine(); // Consume the newline character left by previous sc.nextInt()
            String cuttingType = sc.nextLine();

            System.out.println("Please enter Machine Type:");
            String machineType = sc.nextLine();

            try (CallableStatement cutStatement = connection.prepareCall(ENTER_NEW_CUT_PROCESS)) {
                // Set parameters for the Cut Process
                cutStatement.setString(1, process_id);
                cutStatement.setString(2, cuttingType);
                cutStatement.setString(3, machineType);

                // Register output parameters for error code and message
                cutStatement.registerOutParameter(4, Types.INTEGER);
                cutStatement.registerOutParameter(5, Types.NVARCHAR);

                System.out.println("Dispatching the stored procedure for Cut Process...");
                cutStatement.execute();

                // Retrieve error code and message from the stored procedure
                int errorCodeCutProcess = cutStatement.getInt(4);
                String errorMessageCutProcess = cutStatement.getString(5);

                if (errorCodeCutProcess != 0) {
                    // Error occurred, display error message
                    System.out.println("Error Code: " + errorCodeCutProcess);
                    System.out.println("Error Message: " + errorMessageCutProcess);
                    System.out.println("Failed to add Cut Process. Please try again.");
                } else {
                    // No error, success
                    System.out.println("Done. Cut Process added.");
                    processDetailsEnteredSuccessfully = true;
                }
            } catch (SQLException e) {
                // Handle SQL exceptions
                System.out.println("Error: " + e.getMessage());
                System.out.println("Failed to add Cut Process. Returning to process type selection.");
            }
```

```
                    break;

                default:
                    System.out.println("Invalid choice. Returning to the main menu.");
                    break;
            }
        }
    }
} catch (SQLException e) {
    // Handle SQL exceptions
    System.out.println("Error: " + e.getMessage());
    System.out.println("Failed to add process. Returning to the main menu.");
}
} catch (SQLException e) {
    System.out.println("Connection error: " + e.getMessage());
}
break;
```

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).

```
508         // Case "4" represents the user's choice to add a new assembly and manufacture record.
509         case "4":
510                 // Get user input for Assembly table
511                 System.out.println("Please enter assembly ID:");
512                 sc.nextLine();
513                 String assemblyId = sc.nextLine();
514                 // read date ordered
515                 String dateOrdered = getUserInputWithFormat("Please enter date ordered (YYYY-MM-DD):", "yyyy-MM-dd");
516                 // read assembly details
517                 System.out.println("Please enter assembly details:");
518                 String assemblyDetails = sc.nextLine();
519                 // read customer name
520                 System.out.println("Please enter customer name:");
521                 String assemblyName = sc.nextLine();
522
523                 // Connecting to the database and executing stored procedure for Assembly
524                 try (Connection connection = DriverManager.getConnection(URL)) {
525                     try (CallableStatement assemblyStatement = connection.prepareCall(ENTER_NEW_ASSEMBLY)) {
526                         assemblyStatement.setString(1, assemblyId);
527                         assemblyStatement.setString(2, dateOrdered);
528                         assemblyStatement.setString(3, assemblyDetails);
529                         assemblyStatement.setString(4, assemblyName);
530
531                         // Register output parameters for error code and message
532                         assemblyStatement.registerOutParameter(5, Types.INTEGER);
533                         assemblyStatement.registerOutParameter(6, Types.NVARCHAR);
534
535                         System.out.println("Dispatching the stored procedure for Assembly...");
536                         assemblyStatement.execute();
537
538                         // Retrieve error code and message from the stored procedure
539                         int derrorCode = assemblyStatement.getInt(5);
540                         String derrorMessage = assemblyStatement.getString(6);
541
542                         if (derrorCode != 0) {
543                             // Error occurred, display error message
544                             System.out.println("Error Code: " + derrorCode);
545                             System.out.println("Error Message: " + derrorMessage);
546                             System.out.println("Failed to add Assembly record. Returning to the main menu.");
547                             break; // Terminate the loop or take appropriate action
548                         } else {
549                             // No error, success
550                             System.out.println("Done. Assembly record added.");
551                         }
552                     } catch (SQLException e) {
553                         // Handle SQL exceptions
554                         System.out.println("Error: " + e.getMessage());
555                         System.out.println("Failed to add Assembly record. Returning to the main menu.");
556                         break; // Terminate the loop or take appropriate action
557                     }
558
559                     // Ask the user if they want to enter a process for the specific assembly
560                     String userChoice = "yes";
561
562                     while (userChoice.equalsIgnoreCase("yes")) {
563                         System.out.println("Do you want to enter process for this assembly? (yes/no):");
564                         userChoice = sc.nextLine();
565
566                         if (userChoice.equalsIgnoreCase("yes")) {
567                             // Get user input for Manufacture table
568                             System.out.println("Please enter process ID for Manufacture:");
569                             String processIdManufacture = sc.nextLine();
```

```
                        boolean isValidProcessId = false;

                        while (!isValidProcessId) {
                            try (CallableStatement manufactureStatement = connection.prepareCall(ENTER_NEW_MANUFACTURE)) {
                                manufactureStatement.setString(1, assemblyId);
                                manufactureStatement.setString(2, processIdManufacture);
                                manufactureStatement.setNull(3, Types.INTEGER); // Placeholder for job_no (null)

                                // Register output parameters for error code and message
                                manufactureStatement.registerOutParameter(4, Types.INTEGER);
                                manufactureStatement.registerOutParameter(5, Types.NVARCHAR);

                                System.out.println("Dispatching the stored procedure for Manufacture...");
                                manufactureStatement.execute();

                                // Retrieve error code and message from the stored procedure
                                int dderrorCode = manufactureStatement.getInt(4);
                                String dderrorMessage = manufactureStatement.getString(5);

                                if (dderrorCode != 0) {
                                    // Error occurred, display error message
                                    System.out.println("Error Code: " + dderrorCode);
                                    System.out.println("Error Message: " + dderrorMessage);
                                    System.out.println("Failed to add Manufacture record. Please enter process ID again:");

                                    // Prompt user for a new process ID
                                    processIdManufacture = sc.next();
                                } else {
                                    // No error, success
                                    System.out.println("Done. Manufacture record added.");
                                    isValidProcessId = true; // Exit the loop
                                }
                            } catch (SQLException e) {
                                // Handle SQL exceptions
                                System.out.println("Error: " + e.getMessage());
                                System.out.println("Failed to add Manufacture record. Please enter process ID again:");

                                // Prompt user for a new process ID
                                processIdManufacture = sc.next();
                            }
                        }
                    }
                }
            } catch (SQLException e) {
                System.out.println("Connection error: " + e.getMessage());
            }
            break;
```

5. Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day).

```
619
620         // CASE "5": represents the user's choice to add a new account and expenditure
621         case "5":
622                 System.out.println("Enter account number:");
623                 int accountNumber = sc.nextInt();
624
625                 String dateEstablished = getUserInputWithFormat("Enter date of establishment (YYYY-MM-DD):", "yyyy-MM-dd");
626
627                 System.out.println("Choose one of the following types of account:");
628                 System.out.println("1. Assembly Account");
629                 System.out.println("2. Department Account");
630                 System.out.println("3. Process Account");
631
632                 int accountTypeChoice = sc.nextInt();
633
634                 System.out.println("Enter account details:");
635                 sc.nextLine(); // Consume the newline character
636                 BigDecimal accountDetails = new BigDecimal(sc.nextLine()); // Use BigDecimal for NUMERIC(10,2)
637
638                 int eErrorCode;
639                 String eErrorMessage;
640
641                 try (Connection connection = DriverManager.getConnection(URL)) {
642                     switch (accountTypeChoice) {
643                         case 1:
644                             try (CallableStatement assemblyAccountStatement = connection.prepareCall(INSERT_ASSEMBLY_ACCOUNT)) {
645                                 assemblyAccountStatement.setInt(1, accountNumber);
646                                 assemblyAccountStatement.setString(2, dateEstablished);
647                                 assemblyAccountStatement.setBigDecimal(3, accountDetails);
648
649                                 assemblyAccountStatement.registerOutParameter(4, Types.INTEGER);
650                                 assemblyAccountStatement.registerOutParameter(5, Types.NVARCHAR);
651
652                                 System.out.println("Dispatching the stored procedure for Assembly Account...");
653                                 assemblyAccountStatement.execute();
654
655                                 eErrorCode = assemblyAccountStatement.getInt(4);
656                                 eErrorMessage = assemblyAccountStatement.getString(5);
657
658                                 // Check for errors in the assembly account addition
659                                 if (eErrorCode != 0) {
660                                     // Error occurred, display error message
661                                     System.out.println("Error Code: " + eErrorCode);
662                                     System.out.println("Error Message: " + eErrorMessage);
663                                     System.out.println("Failed to add Assembly Account record. Returning to the main menu.");
664                                     break; // Terminate the loop or take appropriate action
665                                 }
666
667                                 // Now, prompt the user for expenditure details
668                                 System.out.println("Enter Assembly ID for Expenditure:");
669                                 String eassemblyId = sc.nextLine();
670
671                                 // Insert expenditure details into the assembly_expenditure table
672                                 try (CallableStatement assemblyExpenditureStatement = connection.prepareCall(INSERT_ASSEMBLY_EXPENDITURE)) {
673                                     assemblyExpenditureStatement.setInt(1, accountNumber);
674                                     assemblyExpenditureStatement.setString(2, eassemblyId);
675
676                                     assemblyExpenditureStatement.registerOutParameter(3, Types.INTEGER);
677                                     assemblyExpenditureStatement.registerOutParameter(4, Types.NVARCHAR);
678
679                                     assemblyExpenditureStatement.execute();
680
```

```
                                int aeErrorCode = assemblyExpenditureStatement.getInt(3);
                                String aeErrorMessage = assemblyExpenditureStatement.getString(4);

                                // Check for errors in the assembly expenditure addition
                                if (aeErrorCode != 0) {
                                    // Error occurred, display error message
                                    System.out.println("Error Code: " + aeErrorCode);
                                    System.out.println("Error Message: " + aeErrorMessage);
                                    System.out.println("Failed to add Assembly Expenditure record. Returning to the main menu.");
                                    break; // Terminate the loop or take appropriate action
                                }
                            } catch (SQLException e) {
                                // Handle SQL exceptions for assembly expenditure statement
                                eErrorCode = 50021; // Custom error code for assembly expenditure
                                eErrorMessage = e.getMessage();

                                // Display error message
                                System.out.println("Error Code: " + eErrorCode);
                                System.out.println("Error Message: " + eErrorMessage);
                                System.out.println("Failed to add Assembly Expenditure record. Returning to the main menu.");
                                break; // Terminate the loop or take appropriate action
                            }

                            System.out.println("Assembly Account and Expenditure records added successfully.");
                        } catch (SQLException e) {
                            // Handle SQL exceptions for assembly account statement
                            eErrorCode = 50018; // Custom error code for assembly account
                            eErrorMessage = e.getMessage();

                            // Display error message
                            System.out.println("Error Code: " + eErrorCode);
                            System.out.println("Error Message: " + eErrorMessage);
                            System.out.println("Failed to add Assembly Account record. Returning to the main menu.");
                            break; // Terminate the loop or take appropriate action
                        }
                        break;

                    case 2:
                        try (CallableStatement departmentAccountStatement = connection.prepareCall(INSERT_DEPARTMENT_ACCOUNT)) {
                            departmentAccountStatement.setInt(1, accountNumber);
                            departmentAccountStatement.setString(2, dateEstablished);
                            departmentAccountStatement.setBigDecimal(3, accountDetails);

                            departmentAccountStatement.registerOutParameter(4, Types.INTEGER); // Assuming this is an INTEGER output parameter
                            departmentAccountStatement.registerOutParameter(5, Types.NVARCHAR);

                            System.out.println("Dispatching the stored procedure for Department Account...");
                            departmentAccountStatement.execute();

                            eErrorCode = departmentAccountStatement.getInt(4);
                            eErrorMessage = departmentAccountStatement.getString(5);

                            // Check for errors in the department account addition
                            if (eErrorCode != 0) {
                                // Error occurred, display error message
                                System.out.println("Error Code: " + eErrorCode);
                                System.out.println("Error Message: " + eErrorMessage);
                                System.out.println("Failed to add Department Account record. Returning to the main menu.");
                                break; // Terminate the loop or take appropriate action
                            }
```

```
741
742                             // Now, prompt the user for expenditure details
743                             System.out.println("Enter Department Number for Expenditure:");
744                             int departmentNumber = sc.nextInt();
745
746                             // Insert expenditure details into the department_expenditure table
747                             try (CallableStatement departmentExpenditureStatement = connection.prepareCall(INSERT_DEPARTMENT_EXPENDITURE)) {
748                                 departmentExpenditureStatement.setInt(1, accountNumber);
749                                 departmentExpenditureStatement.setInt(2, departmentNumber);
750
751                                 departmentExpenditureStatement.registerOutParameter(3, Types.INTEGER);
752                                 departmentExpenditureStatement.registerOutParameter(4, Types.NVARCHAR);
753
754                                 departmentExpenditureStatement.execute();
755
756                                 int deErrorCode = departmentExpenditureStatement.getInt(3);
757                                 String deErrorMessage = departmentExpenditureStatement.getString(4);
758
759                                 // Check for errors in the department expenditure addition
760                                 if (deErrorCode != 0) {
761                                     // Error occurred, display error message
762                                     System.out.println("Error Code: " + deErrorCode);
763                                     System.out.println("Error Message: " + deErrorMessage);
764                                     System.out.println("Failed to add Department Expenditure record. Returning to the main menu.");
765                                     break; // Terminate the loop or take appropriate action
766                                 }
767                             } catch (SQLException e) {
768                                 // Handle SQL exceptions for department expenditure statement
769                                 eErrorCode = 50022; // Custom error code for department expenditure
770                                 eErrorMessage = e.getMessage();
771
772                                 // Display error message
773                                 System.out.println("Error Code: " + eErrorCode);
774                                 System.out.println("Error Message: " + eErrorMessage);
775                                 System.out.println("Failed to add Department Expenditure record. Returning to the main menu.");
776                                 break; // Terminate the loop or take appropriate action
777                             }
778
779                             System.out.println("Department Account and Expenditure records added successfully.");
780                         } catch (SQLException e) {
781                             // Handle SQL exceptions for department account statement
782                             eErrorCode = 50019; // Custom error code for department account
783                             eErrorMessage = e.getMessage();
784
785                             // Display error message
786                             System.out.println("Error Code: " + eErrorCode);
787                             System.out.println("Error Message: " + eErrorMessage);
788                             System.out.println("Failed to add Department Account record. Returning to the main menu.");
789                             break; // Terminate the loop or take appropriate action
790                         }
791                         break;
792
793                     case 3:
794                         try (CallableStatement processAccountStatement = connection.prepareCall(INSERT_PROCESS_ACCOUNT)) {
795                             processAccountStatement.setInt(1, accountNumber);
796                             processAccountStatement.setString(2, dateEstablished);
797                             processAccountStatement.setBigDecimal(3, accountDetails);
798
799                             processAccountStatement.registerOutParameter(4, Types.INTEGER);
800                             processAccountStatement.registerOutParameter(5, Types.NVARCHAR);
```

```
                System.out.println("Dispatching the stored procedure for Process Account...");
                processAccountStatement.execute();

                eErrorCode = processAccountStatement.getInt(4);
                eErrorMessage = processAccountStatement.getString(5);

                // Check for errors in the process account addition
                if (eErrorCode != 0) {
                    // Error occurred, display error message
                    System.out.println("Error Code: " + eErrorCode);
                    System.out.println("Error Message: " + eErrorMessage);
                    System.out.println("Failed to add Process Account record. Returning to the main menu.");
                    break; // Terminate the loop or take appropriate action
                }

                // Now, prompt the user for expenditure details
                System.out.println("Enter Process ID for Expenditure:");
                String processId = sc.nextLine();

                // Insert expenditure details into the process_expenditure table
                try (CallableStatement processExpenditureStatement = connection.prepareCall(INSERT_PROCESS_EXPENDITURE)) {
                    processExpenditureStatement.setInt(1, accountNumber);
                    processExpenditureStatement.setString(2, processId);

                    processExpenditureStatement.registerOutParameter(3, Types.INTEGER);
                    processExpenditureStatement.registerOutParameter(4, Types.NVARCHAR);

                    processExpenditureStatement.execute();

                    int peErrorCode = processExpenditureStatement.getInt(3);
                    String peErrorMessage = processExpenditureStatement.getString(4);

                    // Check for errors in the process expenditure addition
                    if (peErrorCode != 0) {
                        // Error occurred, display error message
                        System.out.println("Error Code: " + peErrorCode);
                        System.out.println("Error Message: " + peErrorMessage);
                        System.out.println("Failed to add Process Expenditure record. Returning to the main menu.");
                        break; // Terminate the loop or take appropriate action
                    }
                } catch (SQLException e) {
                    // Handle SQL exceptions for process expenditure statement
                    eErrorCode = 50023; // Custom error code for process expenditure
                    eErrorMessage = e.getMessage();

                    // Display error message
                    System.out.println("Error Code: " + eErrorCode);
                    System.out.println("Error Message: " + eErrorMessage);
                    System.out.println("Failed to add Process Expenditure record. Returning to the main menu.");
                    break; // Terminate the loop or take appropriate action
                }

                System.out.println("Process Account and Expenditure records added successfully.");
            } catch (SQLException e) {
                // Handle SQL exceptions for process account statement
                eErrorCode = 50020; // Custom error code for process account
                eErrorMessage = e.getMessage();
```

```
                    // Display error message
                    System.out.println("Error Code: " + eErrorCode);
                    System.out.println("Error Message: " + eErrorMessage);
                    System.out.println("Failed to add Process Account record. Returning to the main menu.");
                    break; // Terminate the loop or take appropriate action
                }
                break;

            default:
                System.out.println("Invalid choice. Returning to the main menu.");
                return;
        }

        if (eErrorCode != 0) {
            // Display error message
            System.out.println("Error Code: " + eErrorCode);
            System.out.println("Error Message: " + eErrorMessage);
            System.out.println("Failed to add account. Returning to the main menu.");
        } else {
            System.out.println("Account record added successfully.");
        }
    } catch (SQLException e) {
        // Handle SQL exceptions for connection
        System.out.println("Connection error: " + e.getMessage());
    }
    break;
```

6. Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day).

```
887
888         // case "6": represents the user's choice to add a new job
889         case "6":
890                 // Insert into Job
891                 System.out.println("Please enter job number:");
892                 int jobNumber = sc.nextInt();
893
894                 // Ask the user to enter the date of commencement
895                 String dateCommenced = getUserInputWithFormat("Please enter date the job commenced (YYYY-MM-DD):", "yyyy-MM-dd");
896
897                 try (Connection connection = DriverManager.getConnection(URL)) {
898                     try (CallableStatement jobStatement = connection.prepareCall(ENTER_NEW_JOB)) {
899                         jobStatement.setInt(1, jobNumber);
900                         jobStatement.setString(2, dateCommenced);
901
902                         // Register output parameters for error code and message
903                         jobStatement.registerOutParameter(3, Types.INTEGER);
904                         jobStatement.registerOutParameter(4, Types.NVARCHAR);
905
906                         System.out.println("Dispatching the stored procedure for Job...");
907                         jobStatement.execute();
908
909                         // Retrieve error code and message from the stored procedure
910                         int ferrorCode = jobStatement.getInt(3);
911                         String ferrorMessage= jobStatement.getString(4);
912
913                         if (ferrorCode != 0) {
914                             // Error occurred, display error message
915                             System.out.println("Error Code: " + ferrorCode);
916                             System.out.println("Error Message: " + ferrorMessage);
917                             System.out.println("Failed to add Job record. Returning to the main menu.");
918                         } else {
919                             System.out.println("Job record added successfully.");
920
921                             // Ask the user to choose the type of job
922                             System.out.println("Choose the type of job:");
923                             System.out.println("1. Fit Job");
924                             System.out.println("2. Paint Job");
925                             System.out.println("3. Cut Job");
926                             int jobTypeChoice = sc.nextInt();
927                             sc.nextLine(); // Consume the newline character
928
929                             // Call the procedure to insert details based on the user's choice
930                             switch (jobTypeChoice) {
931                          // Inside the main switch statement
932                             case 1:
933                                 // User's choice for Fit Job
934                                 // Insert into Fit Job
935                                 try (Connection fconnection = DriverManager.getConnection(URL)) {
936                                     try (CallableStatement fitJobStatement = fconnection.prepareCall(INSERT_FIT)) {
937                                         fitJobStatement.setInt(1, jobNumber);
938
939                                         // Register output parameters for error code and message
940                                         fitJobStatement.registerOutParameter(2, Types.INTEGER);
941                                         fitJobStatement.registerOutParameter(3, Types.NVARCHAR);
942
943                                         System.out.println("Dispatching the stored procedure for Fit Job...");
944                                         fitJobStatement.execute();
945
946                                         // Retrieve error code and message from the stored procedure
947                                         int errorCodeFitJob = fitJobStatement.getInt(2);
948                                         String errorMessageFitJob = fitJobStatement.getString(3);
```

```
949
950                                     if (errorCodeFitJob != 0) {
951                                         // Error occurred, display error message
952                                         System.out.println("Error Code: " + errorCodeFitJob);
953                                         System.out.println("Error Message: " + errorMessageFitJob);
954                                         System.out.println("Failed to add Fit Job details. Returning to the main menu.");
955                                     } else {
956                                         System.out.println("Fit Job details added successfully.");
957
958                                         // Ask the user if they want to insert Manufacture details
959                                         System.out.println("Do you want to insert assembly_id and process_id pairs for this job? (yes/no):");
960                                         String userInputManufacture = sc.next().toLowerCase();
961
962                                         if (userInputManufacture.equals("yes")) {
963                                             boolean insertMoreManufacture = true;
964
965                                             while (insertMoreManufacture) {
966                                                 // Ask the user to enter the assembly_id and process_id pair
967                                                 System.out.println("Please enter assembly_id:");
968                                                 String fassemblyId = sc.next();
969
970                                                 System.out.println("Please enter process_id:");
971                                                 String processId = sc.next();
972
973                                                 // Call the procedure to insert details into the Manufacture table
974                                                 try (CallableStatement insertManufactureStatement = connection.prepareCall(INSERT_MANUFACTURE_JOB)) {
975                                                     insertManufactureStatement.setInt(1, jobNumber);
976                                                     insertManufactureStatement.setString(2, fassemblyId);
977                                                     insertManufactureStatement.setString(3, processId);
978
979                                                     // Register output parameters for error code and message
980                                                     insertManufactureStatement.registerOutParameter(4, Types.INTEGER);
981                                                     insertManufactureStatement.registerOutParameter(5, Types.NVARCHAR);
982
983                                                     System.out.println("Dispatching the stored procedure for Manufacture...");
984                                                     insertManufactureStatement.execute();
985
986                                                     // Retrieve error code and message from the stored procedure
987                                                     int errorCodeManufacture = insertManufactureStatement.getInt(4);
988                                                     String errorMessageManufacture = insertManufactureStatement.getString(5);
989
990                                                     if (errorCodeManufacture != 0) {
991                                                         // Error occurred, display error message
992                                                         System.out.println("Error Code: " + errorCodeManufacture);
993                                                         System.out.println("Error Message: " + errorMessageManufacture);
994                                                         System.out.println("Failed to add Manufacture details.");
995
996                                                         // Ask the user if they want to try entering assembly_id and process_id again
997                                                         System.out.println("Do you want to try entering assembly_id and process_id again? (yes/no):");
998                                                         String userInputTryAgain = sc.next().toLowerCase();
999
1000                                                        if (userInputTryAgain.equals("no")) {
1001                                                            insertMoreManufacture = false; // Exit the loop
1002                                                        }
1003                                                    } else {
1004                                                        System.out.println("Manufacture details added successfully.");
1005
1006                                                        // Ask the user if they want to insert more assembly_id and process_id pairs
1007                                                        System.out.println("Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):");
1008                                                        String userInputMorePairs = sc.next().toLowerCase();
1009
1010                                                        if (userInputMorePairs.equals("no")) {
1011                                                            insertMoreManufacture = false; // Exit the loop
```

```
                                                    insertMoreManufacture = false; // Exit the loop
                                                }
                                            }
                                        } catch (SQLException e) {
                                            // Handle SQL exceptions
                                            System.out.println("Error: " + e.getMessage());
                                            System.out.println("Failed to add Manufacture details. Returning to the main menu.");
                                            insertMoreManufacture = false; // Exit the loop on error
                                        }
                                    }
                                }
                            }
                        } catch (SQLException e) {
                            // Handle SQL exceptions
                            System.out.println("Error: " + e.getMessage());
                            System.out.println("Failed to add Fit Job details. Returning to the main menu.");
                        })
                        break;

                case 2:
                    // User's choice for Paint Job
                    // Insert into Paint Job
                    try (Connection fconnection = DriverManager.getConnection(URL)) {
                        try (CallableStatement paintJobStatement = fconnection.prepareCall(INSERT_PAINT)) {
                            paintJobStatement.setInt(1, jobNumber);

                            // Register output parameters for error code and message
                            paintJobStatement.registerOutParameter(2, Types.INTEGER);
                            paintJobStatement.registerOutParameter(3, Types.NVARCHAR);

                            System.out.println("Dispatching the stored procedure for Paint Job...");
                            paintJobStatement.execute();

                            // Retrieve error code and message from the stored procedure
                            int errorCodePaintJob = paintJobStatement.getInt(2);
                            String errorMessagePaintJob = paintJobStatement.getString(3);

                            if (errorCodePaintJob != 0) {
                                // Error occurred, display error message
                                System.out.println("Error Code: " + errorCodePaintJob);
                                System.out.println("Error Message: " + errorMessagePaintJob);
                                System.out.println("Failed to add Paint Job details. Returning to the main menu.");
                            } else {
                                System.out.println("Paint Job details added successfully.");

                                // Ask the user if they want to insert Manufacture details
                                System.out.println("Do you want to insert assembly_id and process_id pairs for this job? (yes/no):");
                                String userInputManufacture = sc.next().toLowerCase();

                                if (userInputManufacture.equals("yes")) {
                                    boolean insertMoreManufacture = true;

                                    while (insertMoreManufacture) {
                                        // Ask the user to enter the assembly_id and process_id pair
                                        System.out.println("Please enter assembly_id:");
                                        String fassemblyId = sc.next();

                                        System.out.println("Please enter process_id:");
                                        String processId = sc.next();

                                        // Call the procedure to insert details into the Manufacture table
                                        try (CallableStatement insertManufactureStatement = connection.prepareCall(INSERT_MANUFACTURE_JOB)) {
                                            insertManufactureStatement.setInt(1, jobNumber);
```

```
                                    insertManufactureStatement.setString(2, fassemblyId);
                                    insertManufactureStatement.setString(3, processId);

                                    // Register output parameters for error code and message
                                    insertManufactureStatement.registerOutParameter(4, Types.INTEGER);
                                    insertManufactureStatement.registerOutParameter(5, Types.NVARCHAR);

                                    System.out.println("Dispatching the stored procedure for Manufacture...");
                                    insertManufactureStatement.execute();

                                    // Retrieve error code and message from the stored procedure
                                    int errorCodeManufacture = insertManufactureStatement.getInt(4);
                                    String errorMessageManufacture = insertManufactureStatement.getString(5);

                                    if (errorCodeManufacture != 0) {
                                        // Error occurred, display error message
                                        System.out.println("Error Code: " + errorCodeManufacture);
                                        System.out.println("Error Message: " + errorMessageManufacture);
                                        System.out.println("Failed to add Manufacture details.");

                                        // Ask the user if they want to try entering assembly_id and process_id again
                                        System.out.println("Do you want to try entering assembly_id and process_id again? (yes/no):");
                                        String userInputTryAgain = sc.next().toLowerCase();

                                        if (userInputTryAgain.equals("no")) {
                                            insertMoreManufacture = false; // Exit the loop
                                        }
                                    } else {
                                        System.out.println("Manufacture details added successfully.");

                                        // Ask the user if they want to insert more assembly_id and process_id pairs
                                        System.out.println("Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):");
                                        String userInputMorePairs = sc.next().toLowerCase();

                                        if (userInputMorePairs.equals("no")) {
                                            insertMoreManufacture = false; // Exit the loop
                                        }
                                    }
                                } catch (SQLException e) {
                                    // Handle SQL exceptions
                                    System.out.println("Error: " + e.getMessage());
                                    System.out.println("Failed to add Manufacture details. Returning to the main menu.");
                                    insertMoreManufacture = false; // Exit the loop on error
                                }
                            }
                        }
                    }
                } catch (SQLException e) {
                    // Handle SQL exceptions
                    System.out.println("Error: " + e.getMessage());
                    System.out.println("Failed to add Paint Job details. Returning to the main menu.");
                }
            }
            break;


        case 3:
            // User's choice for Cut Job
            // Insert into Cut Job
            try (Connection ffconnection = DriverManager.getConnection(URL)) {
                try (CallableStatement cutJobStatement = ffconnection.prepareCall(INSERT_CUT)) {
                    cutJobStatement.setInt(1, jobNumber);

                        // Register output parameters for error code and message
                        cutJobStatement.registerOutParameter(2, Types.INTEGER);
                        cutJobStatement.registerOutParameter(3, Types.NVARCHAR);

                        System.out.println("Dispatching the stored procedure for Cut Job...");
                        cutJobStatement.execute();

                        // Retrieve error code and message from the stored procedure
                        int errorCodeCutJob = cutJobStatement.getInt(2);
                        String errorMessageCutJob = cutJobStatement.getString(3);

                        if (errorCodeCutJob != 0) {
                            // Error occurred, display error message
                            System.out.println("Error Code: " + errorCodeCutJob);
                            System.out.println("Error Message: " + errorMessageCutJob);
                            System.out.println("Failed to add Cut Job details. Returning to the main menu.");
                        } else {
                            System.out.println("Cut Job details added successfully.");

                            // Ask the user if they want to insert Manufacture details
                            System.out.println("Do you want to insert assembly_id and process_id pairs for this job? (yes/no):");
                            String userInputManufacture = sc.next().toLowerCase();

                            if (userInputManufacture.equals("yes")) {
                                boolean insertMoreManufacture = true;

                                while (insertMoreManufacture) {
                                    // Ask the user to enter the assembly_id and process_id pair
                                    System.out.println("Please enter assembly_id:");
                                    String fffassemblyId = sc.next();

                                    System.out.println("Please enter process_id:");
                                    String processId = sc.next();

                                    // Call the procedure to insert details into the Manufacture table
                                    try (CallableStatement insertManufactureStatement = connection.prepareCall(INSERT_MANUFACTURE_JOB)) {
                                        insertManufactureStatement.setInt(1, jobNumber);
                                        insertManufactureStatement.setString(2, fffassemblyId);
                                        insertManufactureStatement.setString(3, processId);

                                        // Register output parameters for error code and message
                                        insertManufactureStatement.registerOutParameter(4, Types.INTEGER);
                                        insertManufactureStatement.registerOutParameter(5, Types.NVARCHAR);

                                        System.out.println("Dispatching the stored procedure for Manufacture...");
                                        insertManufactureStatement.execute();

                                        // Retrieve error code and message from the stored procedure
                                        int errorCodeManufacture = insertManufactureStatement.getInt(4);
                                        String errorMessageManufacture = insertManufactureStatement.getString(5);

                                        if (errorCodeManufacture != 0) {
                                            // Error occurred, display error message
                                            System.out.println("Error Code: " + errorCodeManufacture);
                                            System.out.println("Error Message: " + errorMessageManufacture);
                                            System.out.println("Failed to add Manufacture details.");

                                            // Ask the user if they want to try entering assembly_id and process_id again
                                            System.out.println("Do you want to try entering assembly_id and process_id again? (yes/no):");
                                            String userInputTryAgain = sc.next().toLowerCase();
```

```
                                                if (userInputTryAgain.equals("no")) {
                                                    insertMoreManufacture = false; // Exit the loop
                                                }
                                            } else {
                                                System.out.println("Manufacture details added successfully.");

                                                // Ask the user if they want to insert more assembly_id and process_id pairs
                                                System.out.println("Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):");
                                                String userInputMorePairs = sc.next().toLowerCase();

                                                if (userInputMorePairs.equals("no")) {
                                                    insertMoreManufacture = false; // Exit the loop
                                                }
                                            }
                                        } catch (SQLException e) {
                                            // Handle SQL exceptions
                                            System.out.println("Error: " + e.getMessage());
                                            System.out.println("Failed to add Manufacture details. Returning to the main menu.");
                                            insertMoreManufacture = false; // Exit the loop on error
                                        }
                                    }
                                }
                            }
                        } catch (SQLException e) {
                            // Handle SQL exceptions
                            System.out.println("Error: " + e.getMessage());
                            System.out.println("Failed to add Cut Job details. Returning to the main menu.");
                        }
                    }
                    break;



                        default:
                            System.out.println("Invalid job type choice. Returning to the main menu.");
                            break;
                    }
                }
            } catch (SQLException e) {
                // Handle SQL exceptions
                System.out.println("Error: " + e.getMessage());
                System.out.println("Failed to add job. Returning to the main menu.");
            }
            break;
```

7. At the completion of a job, enter the date it completed and the information relevant to the type of job (50/day).

```
        case "7":
            // Ask the user to enter the job number
            System.out.println("Please enter the job number:");
            int updateJobNumber = sc.nextInt();

            // Loop to handle errors and ask for the correct date of completion
            while (true) {
                // Check if the job number exists and determine the job type
                try (Connection checkConnection = DriverManager.getConnection(URL)) {
                    try (CallableStatement checkStatement = checkConnection.prepareCall(CHECK_JOB_EXISTENCE)) {
                        checkStatement.setInt(1, updateJobNumber);
                        checkStatement.registerOutParameter(2, Types.INTEGER);

                        // Execute the stored procedure to check job existence
                        checkStatement.execute();

                        // Retrieve the existsFlag from the stored procedure
                        int existsFlag = checkStatement.getInt(2);

                        if (existsFlag != 0) {
                            boolean isValidDate = false;
                            String dateCompleted = null;

                            while (!isValidDate) {
                                // Job exists, ask for date of completion
                                dateCompleted = getUserInputWithFormat("Please enter date the job completed (YYYY-MM-DD):", "yyyy-MM-dd");

                                // Update the completion date for the job
                                try (CallableStatement updateCompletionDateStatement = checkConnection.prepareCall(UPDATE_COMPLETION_DATE)) {
                                    updateCompletionDateStatement.setInt(1, updateJobNumber);
                                    updateCompletionDateStatement.setString(2, dateCompleted);
                                    updateCompletionDateStatement.registerOutParameter(3, Types.INTEGER);
                                    updateCompletionDateStatement.registerOutParameter(4, Types.NVARCHAR);

                                    // Execute the stored procedure to update the completion date
                                    updateCompletionDateStatement.execute();

                                    // Retrieve error code and message from the stored procedure
                                    int errorCodeUpdateCompletionDate = updateCompletionDateStatement.getInt(3);
                                    String errorMessageUpdateCompletionDate = updateCompletionDateStatement.getString(4);

                                    if (errorCodeUpdateCompletionDate != 0) {
                                        // Error occurred, display error message
                                        System.out.println("Error Code: " + errorCodeUpdateCompletionDate);
                                        System.out.println("Error Message: " + errorMessageUpdateCompletionDate);
                                        System.out.println("Failed to update completion date.");

                                        // Prompt user for a correct date
                                        isValidDate = false;
                                    } else {
                                        System.out.println("Completion date updated successfully.");
                                        isValidDate = true; // Exit the loop as the update was successful
                                    }
                                } catch (SQLException e) {
                                    // Handle SQL exceptions
                                    System.out.println("Error: " + e.getMessage());
                                    System.out.println("Failed to update completion date.");
                                    isValidDate = false; // Prompt user for a correct date
                                }
                            }
```

```
                // Determine the type of job (Fit, Paint, or Cut)
                String jobType;
                try (CallableStatement determineTypeStatement = checkConnection.prepareCall(DETERMINE_JOB_TYPE)) {
                    determineTypeStatement.setInt(1, updateJobNumber);
                    determineTypeStatement.registerOutParameter(2, Types.NVARCHAR);

                    // Execute the stored procedure to determine job type
                    determineTypeStatement.execute();

                    // Retrieve the job type from the stored procedure
                    jobType = determineTypeStatement.getString(2);

                    // Additional logic based on job type
                    switch (jobType) {
                        case "Fit Job":
                            // Ask the user to enter the new fit_labor_time
                            System.out.println("This is a Fit Job:");
                            System.out.println("Please enter the new fit labor time (HH:mm:ss):");
                            String fitLaborTimeString = sc.next();
                            LocalTime fitLaborTime = LocalTime.parse(fitLaborTimeString);

                            // Call the stored procedure to update Fit Job labor time
                            try (CallableStatement updateFitJobStatement = checkConnection.prepareCall(UPDATE_FIT_JOB_LABOR_TIME)) {
                                updateFitJobStatement.setInt(1, updateJobNumber);
                                updateFitJobStatement.setTime(2, Time.valueOf(fitLaborTime));
                                updateFitJobStatement.registerOutParameter(3, Types.INTEGER);
                                updateFitJobStatement.registerOutParameter(4, Types.NVARCHAR);

                                // Execute the stored procedure to update Fit Job labor time
                                updateFitJobStatement.execute();

                                // Retrieve error code and message from the stored procedure
                                int errorCodeUpdateFitJob = updateFitJobStatement.getInt(3);
                                String errorMessageUpdateFitJob = updateFitJobStatement.getString(4);

                                if (errorCodeUpdateFitJob != 0) {
                                    // Error occurred, display error message
                                    System.out.println("Error Code: " + errorCodeUpdateFitJob);
                                    System.out.println("Error Message: " + errorMessageUpdateFitJob);
                                    System.out.println("Failed to update Fit Job labor time.");
                                } else {
                                    System.out.println("Fit Job labor time updated successfully.");
                                }
                            } catch (SQLException e) {
                                // Handle SQL exceptions
                                System.out.println("Error: " + e.getMessage());
                                System.out.println("Failed to update Fit Job labor time.");
                            }
                            break;

                        case "Paint Job":
                            // Ask the user to enter the new details for Paint Job
                            System.out.println("This is a Paint Job:");
                            System.out.println("Please enter the new color:");
                            String paintColor = sc.next();

                            System.out.println("Please enter the new volume:");
                            double paintVolume = sc.nextDouble();

                            System.out.println("Please enter the new paint labor time (HH:mm:ss):");
                            String paintLaborTimeString = sc.next();
                            LocalTime paintLaborTime = LocalTime.parse(paintLaborTimeString);
```

```
                                    // Call the stored procedure to update Paint Job details
                                    try (CallableStatement updatePaintJobStatement = checkConnection.prepareCall(UPDATE_PAINT_JOB_DETAILS))
                                        updatePaintJobStatement.setInt(1, updateJobNumber);
                                        updatePaintJobStatement.setString(2, paintColor);
                                        updatePaintJobStatement.setDouble(3, paintVolume);
                                        updatePaintJobStatement.setTime(4, Time.valueOf(paintLaborTime));
                                        updatePaintJobStatement.registerOutParameter(5, Types.INTEGER);
                                        updatePaintJobStatement.registerOutParameter(6, Types.NVARCHAR);

                                        // Execute the stored procedure to update Paint Job details
                                        updatePaintJobStatement.execute();

                                        // Retrieve error code and message from the stored procedure
                                        int errorCodeUpdatePaintJob = updatePaintJobStatement.getInt(5);
                                        String errorMessageUpdatePaintJob = updatePaintJobStatement.getString(6);

                                        if (errorCodeUpdatePaintJob != 0) {
                                            // Error occurred, display error message
                                            System.out.println("Error Code: " + errorCodeUpdatePaintJob);
                                            System.out.println("Error Message: " + errorMessageUpdatePaintJob);
                                            System.out.println("Failed to update Paint Job details.");
                                        } else {
                                            System.out.println("Paint Job details updated successfully.");
                                        }
                                    } catch (SQLException e) {
                                        // Handle SQL exceptions
                                        System.out.println("Error: " + e.getMessage());
                                        System.out.println("Failed to update Paint Job details.");
                                    }
                                    break;

                                case "Cut Job":
                                    // Ask the user to enter the new details for Cut Job
                                    System.out.println("This is a Cut Job:");
                                    System.out.println("Please enter the new type of machine:");
                                    String machineType = sc.next();

                                    System.out.println("Please enter the new time of machine (HH:mm:ss):");
                                    String machineTimeString = sc.next();
                                    LocalTime machineTime = LocalTime.parse(machineTimeString);

                                    System.out.println("Please enter the new material used:");
                                    String machineUsed = sc.next();

                                    System.out.println("Please enter the new cut labor time (HH:mm:ss):");
                                    String cutLaborTimeString = sc.next();
                                    LocalTime cutLaborTime = LocalTime.parse(cutLaborTimeString);

                                    // Call the stored procedure to update Cut Job details
                                    try (CallableStatement updateCutJobStatement = checkConnection.prepareCall(UPDATE_CUT_JOB_DETAILS)) {
                                        updateCutJobStatement.setInt(1, updateJobNumber);
                                        updateCutJobStatement.setString(2, machineType);
                                        updateCutJobStatement.setTime(3, Time.valueOf(machineTime));
                                        updateCutJobStatement.setString(4, machineUsed);
                                        updateCutJobStatement.setTime(5, Time.valueOf(cutLaborTime));
                                        updateCutJobStatement.registerOutParameter(6, Types.INTEGER);
                                        updateCutJobStatement.registerOutParameter(7, Types.NVARCHAR);

                                        // Execute the stored procedure to update Cut Job details
                                        updateCutJobStatement.execute();

                                           // Retrieve error code and message from the stored procedure
                                           int errorCodeUpdateCutJob = updateCutJobStatement.getInt(6);
                                           String errorMessageUpdateCutJob = updateCutJobStatement.getString(7);

                                           if (errorCodeUpdateCutJob != 0) {
                                               // Error occurred, display error message
                                               System.out.println("Error Code: " + errorCodeUpdateCutJob);
                                               System.out.println("Error Message: " + errorMessageUpdateCutJob);
                                               System.out.println("Failed to update Cut Job details.");
                                           } else {
                                               System.out.println("Cut Job details updated successfully.");
                                           }
                                       } catch (SQLException e) {
                                           // Handle SQL exceptions
                                           System.out.println("Error: " + e.getMessage());
                                           System.out.println("Failed to update Cut Job details.");
                                       }
                                       break;

                                   default:
                                       System.out.println("Invalid job type. Returning to the main menu.");
                                       break;
                               }
                           }

                           break; // Exit the loop as the update was successful
                       } else {
                           // Error: Job does not exist
                           System.out.println("Error: Job with the specified job number does not exist. Returning to the main menu.");
                           break; // Exit the loop as the job does not exist
                       }
                   } catch (SQLException e) {
                       // Handle SQL exceptions
                       System.out.println("Error: " + e.getMessage());
                       System.out.println("Failed to update job details. Returning to the main menu.");
                       break; // Exit the loop in case of a SQL exception
                   }
               }
           }
           break;
```

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 (50/day).

```
            // Prompt the user to enter transaction details for a specific case (case "8")
        case "8":
            System.out.println("Please enter transaction-no:");
            sc.nextLine();
            // read transaction-no
            int transaction_no = sc.nextInt();

            System.out.println("Please enter sup-cost:");
            // read sup-cost
            double sup_cost = sc.nextDouble();

            System.out.println("Do you want to enter assembly account? (yes/no):");
            String enterAssembly = sc.next().toLowerCase();
            int assembly_account_no = 0;

            // Check if the user wants to enter an assembly account
            if (enterAssembly.equals("yes")) {
                System.out.println("Please enter assembly account no:");
                assembly_account_no = sc.nextInt();
            }

            System.out.println("Do you want to enter department account? (yes/no):");
            String enterDepartment = sc.next().toLowerCase();
            int department_account_no = 0;

            // Check if the user wants to enter a department account
            if (enterDepartment.equals("yes")) {
                System.out.println("Please enter department account no:");
                department_account_no = sc.nextInt();
            }

            System.out.println("Do you want to enter process account? (yes/no):");
            String enterProcess = sc.next().toLowerCase();
            int process_account_no = 0;

            // Check if the user wants to enter a process account
            if (enterProcess.equals("yes")) {
                System.out.println("Please enter process account no:");
                process_account_no = sc.nextInt();
            }

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

            int accountsUpdatedCount = 0;

            // Establish a database connection
            try (Connection connection = DriverManager.getConnection(URL)) {
                try (CallableStatement statement = connection.prepareCall(ENTER_NEW_TRANSACTION)) {
                    // Set parameters for the transaction
                    statement.setInt(1, transaction_no);
                    statement.setDouble(2, sup_cost);

                    // Check and set assembly account if provided
                    if (assembly_account_no != 0) {
                        statement.setInt(3, assembly_account_no);
                        accountsUpdatedCount++;
                        System.out.println("Updating assembly account.");
                    } else {
                        statement.setNull(3, Types.INTEGER);
                    }
```

```
                    // Check and set department account if provided
                    if (department_account_no != 0) {
                        statement.setInt(4, department_account_no);
                        accountsUpdatedCount++;
                        System.out.println("Updating department account.");
                    } else {
                        statement.setNull(4, Types.INTEGER);
                    }

                    // Check and set process account if provided
                    if (process_account_no != 0) {
                        statement.setInt(5, process_account_no);
                        accountsUpdatedCount++;
                        System.out.println("Updating process account.");
                    } else {
                        statement.setNull(5, Types.INTEGER);
                    }

                    System.out.println("Dispatching the stored procedure for Transaction...");

                    // Execute the stored procedure for the transaction
                    boolean accountsUpdated = statement.execute();

                    // Display the number of accounts updated
                    if (accountsUpdatedCount > 0) {
                        System.out.println("Cost updated for " + accountsUpdatedCount + " account(s).");
                    } else {
                        System.out.println("No cost update.");
                    }
                } catch (SQLException e) {
                    // Handle SQL exceptions
                    System.out.println("Error: " + e.getMessage());
                    System.out.println("Failed to update costs. Returning to the main menu.");
                }
            } catch (SQLException e) {
                System.out.println("Connection error: " + e.getMessage());
            }
            break;
```

9. Retrieve the total cost incurred on an assembly-id (200/day).

```
        // Prompt the user to enter an assembly id to retrieve total cost (case "9")
        case "9":
            System.out.println("Enter an assembly id to retrieve total cost:");
            String assemblyIdi = sc.next();

            // Establish a database connection
            try (Connection connection = DriverManager.getConnection(URL)) {
                // Retrieve total cost from the Cost table using a stored procedure
                try (CallableStatement costStatement = connection.prepareCall(GET_TOTAL_COST)) {
                    costStatement.setString(1, assemblyIdi);
                    costStatement.registerOutParameter(2, Types.DOUBLE);
                    costStatement.registerOutParameter(3, Types.NVARCHAR);

                    // Execute the stored procedure to get total cost
                    costStatement.execute();

                    // Retrieve total cost and error message from the stored procedure
                    double totalCost = costStatement.getDouble(2);
                    String errorMessage = costStatement.getString(3);

                    // Display the total cost or an error message, if any
                    if (errorMessage != null) {
                        System.out.println("Error Message: " + errorMessage);
                    } else {
                        System.out.println("Total cost incurred on assembly_id " + assemblyIdi + ": " + totalCost);
                    }
                } catch (SQLException e) {
                    // Handle SQL exceptions for the cost statement
                    int errorCode23 = 50025; // Custom error code
                    String errorMessage23 = e.getMessage();
                    System.out.println("Error Code: " + errorCode23);
                    System.out.println("Error Message: " + errorMessage23);
                    System.out.println("Failed to retrieve total cost. Returning to the main menu.");
                }
            } catch (SQLException e) {
                // Handle SQL exceptions for the database connection
                System.out.println("Connection error: " + e.getMessage());
            }
            break;
```

10. Retrieve the total labor time within a department for jobs completed in the department during a given date (20/day).

```
1616            // Prompt the user to enter department number and date to retrieve total labor time (case "10")
1617            case "10":
1618                System.out.println("Enter department number:");
1619                String departmentNumberCase10 = sc.next();
1620
1621                System.out.println("Enter the date (YYYY-MM-DD):");
1622                String givenDateCase10 = sc.next();
1623
1624                // Establish a database connection
1625                try (Connection connection = DriverManager.getConnection(URL)) {
1626                    // Retrieve total labor time from Process, Manufacture, and Job tables using a stored procedure
1627                    try (CallableStatement totalLaborTimeStatement = connection.prepareCall(GET_TOTAL_LABOR_TIME_IN_DEPARTMENT))
1628                        totalLaborTimeStatement.setString(1, departmentNumberCase10);
1629                        totalLaborTimeStatement.setString(2, givenDateCase10);
1630                        totalLaborTimeStatement.registerOutParameter(3, Types.FLOAT);
1631                        totalLaborTimeStatement.registerOutParameter(4, Types.NVARCHAR);
1632
1633                        // Execute the stored procedure to get total labor time
1634                        totalLaborTimeStatement.execute();
1635
1636                        // Retrieve total labor time and error message from the stored procedure
1637                        float totalLaborTime = totalLaborTimeStatement.getFloat(3);
1638                        String errorMessageCase10 = totalLaborTimeStatement.getString(4);
1639
1640                        // Display the total labor time or an error message, if any
1641                        if (errorMessageCase10 != null) {
1642                            System.out.println("Error Message: " + errorMessageCase10);
1643                        } else {
1644                            System.out.println("Total labor time in department " + departmentNumberCase10 +
1645                                    " for jobs completed on " + givenDateCase10 + ": " + totalLaborTime + " minutes");
1646                        }
1647                    } catch (SQLException e) {
1648                        // Handle SQL exceptions for the total labor time statement
1649                        int errorCodeCase10 = 50020; // Custom error code
1650                        String errorMessageCase10 = e.getMessage();
1651                        System.out.println("Error Code: " + errorCodeCase10);
1652                        System.out.println("Error Message: " + errorMessageCase10);
1653                        System.out.println("Failed to retrieve total labor time. Returning to the main menu.");
1654                    }
1655                } catch (SQLException e) {
1656                    // Handle SQL exceptions for the database connection
1657                    System.out.println("Connection error: " + e.getMessage());
1658                }
1659                break;
1660
```

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/day).

```
// Case 11
        // Prompt the user to enter assembly_id to retrieve processes (case "11")
       case "11":
           System.out.println("Enter assembly_id to retrieve processes:");
           String asssemblyId = sc.next();

           // Establish a database connection
           try (Connection connection = DriverManager.getConnection(URL)) {
               // Retrieve processes for the given assembly_id using a stored procedure
               try (CallableStatement processesStatement = connection.prepareCall(GET_PROCESSES_FOR_ASSEMBLY)) {
                   processesStatement.setString(1, asssemblyId);
                   processesStatement.registerOutParameter(2, Types.NVARCHAR);
                   processesStatement.registerOutParameter(3, Types.NVARCHAR);

                   // Execute the stored procedure to get processes
                   processesStatement.execute();

                   // Retrieve process details and error message from the stored procedure
                   String processDetails = processesStatement.getString(2);
                   String errorMessageCase11 = processesStatement.getString(3);

                   // Display the processes or an error message, if any
                   if (errorMessageCase11 != null) {
                       System.out.println("Error Message: " + errorMessageCase11);
                   } else {
                       System.out.println("Processes for assembly_id " + asssemblyId + ":\n" + processDetails);
                   }
               } catch (SQLException e) {
                   // Handle SQL exceptions for the processes statement
                   int errorCodeCase11 = 50020; // Custom error code
                   String errorMessageCase11 = e.getMessage();
                   System.out.println("Error Code: " + errorCodeCase11);
                   System.out.println("Error Message: " + errorMessageCase11);
                   System.out.println("Failed to retrieve processes. Returning to the main menu.");
               }
           } catch (SQLException e) {
               // Handle SQL exceptions for the database connection
               System.out.println("Connection error: " + e.getMessage());
           }
           break;
```

12. Retrieve the customers (in name order) whose category is in a given range (100/day)

```
// Case 12
        // Prompt the user to enter the minimum and maximum category to retrieve customers (case "12")
       case "12":
           System.out.println("Enter the minimum category:");
           int minCategory = sc.nextInt();

           System.out.println("Enter the maximum category:");
           int maxCategory = sc.nextInt();

           // Establish a database connection
           try (Connection connection = DriverManager.getConnection(URL)) {
               // Retrieve customers in the given category range using a stored procedure
               try (CallableStatement customersStatement = connection.prepareCall(GET_CUSTOMERS_IN_RANGE)) {
                   customersStatement.setInt(1, minCategory);
                   customersStatement.setInt(2, maxCategory);
                   customersStatement.registerOutParameter(3, Types.NVARCHAR);
                   customersStatement.registerOutParameter(4, Types.NVARCHAR);

                   // Execute the stored procedure to get customers
                   customersStatement.execute();

                   // Retrieve customer details and error message from the stored procedure
                   String customerDetails = customersStatement.getString(3);
                   String errorMessageCase12 = customersStatement.getString(4);

                   // Display the customers or an error message, if any
                   if (errorMessageCase12 != null) {
                       System.out.println("Error Message: " + errorMessageCase12);
                   } else {
                       System.out.println("Customers in the category range " + minCategory + " to " + maxCategory + ":\n" + customerDetails);
                   }
               } catch (SQLException e) {
                   // Handle SQL exceptions for the customers statement
                   int errorCodeCase12 = 50021; // Custom error code
                   String errorMessageCase12 = e.getMessage();
                   System.out.println("Error Code: " + errorCodeCase12);
                   System.out.println("Error Message: " + errorMessageCase12);
                   System.out.println("Failed to retrieve customers. Returning to the main menu.");
               }
           } catch (SQLException e) {
               // Handle SQL exceptions for the database connection
               System.out.println("Connection error: " + e.getMessage());
           }
           break;
```

13. Delete all cut-jobs whose job-no is in a given range (1/month).

```
// Case 13
        // Prompt the user to enter the range for job numbers to delete cut-jobs (case "13")
       case "13":
           System.out.println("Enter the range for job numbers (minJobNo and maxJobNo):");
           int minJobNo = sc.nextInt();
           int maxJobNo = sc.nextInt();

           // Establish a database connection
           try (Connection connection = DriverManager.getConnection(URL)) {
               // Delete cut-jobs within the specified job number range using a stored procedure
               try (CallableStatement deleteCutJobsStatement = connection.prepareCall(DELETE_CUT_JOBS_IN_RANGE)) {
                   deleteCutJobsStatement.setInt(1, minJobNo);
                   deleteCutJobsStatement.setInt(2, maxJobNo);
                   deleteCutJobsStatement.registerOutParameter(3, Types.NVARCHAR);

                   // Execute the stored procedure to delete cut-jobs
                   deleteCutJobsStatement.execute();

                   // Retrieve error message from the stored procedure
                   String errorMessageCase13 = deleteCutJobsStatement.getString(3);

                   // Display success message or an error message, if any
                   if (errorMessageCase13 != null) {
                       System.out.println("Error Message: " + errorMessageCase13);
                   } else {
                       System.out.println("Cut-jobs within the specified range have been deleted.");
                   }
               } catch (SQLException e) {
                   // Handle SQL exceptions for delete cut-jobs statement
                   int errorCodeCase13 = 50023; // Custom error code
                   String errorMessageCase13 = e.getMessage();
                   System.out.println("Error Code: " + errorCodeCase13);
                   System.out.println("Error Message: " + errorMessageCase13);
                   System.out.println("Failed to delete cut-jobs. Returning to the main menu.");
               }
           } catch (SQLException e) {
               // Handle SQL exceptions for the database connection
               System.out.println("Connection error: " + e.getMessage());
           }
           break;
```

14. Change the color of a given paint job (1/week)

```
        // Prompt the user to enter job number and new color for a paint job (case "14")
       case "14":
           System.out.println("Enter job number for paint job:");
           int jobNumberw = sc.nextInt();

           System.out.println("Enter new color:");
           String newColor = sc.next();

           // Establish a database connection
           try (Connection connection = DriverManager.getConnection(URL)) {
               // Change the color of the given paint job using a stored procedure
               try (CallableStatement changeColorStatement = connection.prepareCall(CHANGE_PAINT_JOB_COLOR)) {
                   changeColorStatement.setInt(1, jobNumberw);
                   changeColorStatement.setString(2, newColor);
                   changeColorStatement.registerOutParameter(3, Types.NVARCHAR);

                   // Execute the stored procedure to change the color
                   changeColorStatement.execute();

                   // Retrieve error message from the stored procedure
                   String errorMessageCase14 = changeColorStatement.getString(3);

                   // Display success message or an error message, if any
                   if (errorMessageCase14 != null) {
                       System.out.println("Error Message: " + errorMessageCase14);
                   } else {
                       System.out.println("Color changed successfully for paint job " + jobNumberw);
                   }
               } catch (SQLException e) {
                   // Handle SQL exceptions for changing color statement
                   int errorCodeCase14 = 50022; // Custom error code
                   String errorMessageCase14 = e.getMessage();
                   System.out.println("Error Code: " + errorCodeCase14);
                   System.out.println("Error Message: " + errorMessageCase14);
                   System.out.println("Failed to change color. Returning to the main menu.");
               }
           } catch (SQLException e) {
               // Handle SQL exceptions for the database connection
               System.out.println("Connection error: " + e.getMessage());
           }
           break;
```

15. Import.file:

```
1840            // Prompt the user to enter the input file name for adding customers (case "15")
1841            case "15":
1842                System.out.println("Please enter the input file name:");
1843                String inputFileName = sc.next();
1844
1845                // Print current working directory for debugging
1846                System.out.println("Current Directory: " + System.getProperty("user.dir"));
1847
1848                // Open the file and read customer data until the file is empty
1849                try (Scanner fileScanner = new Scanner(new File(inputFileName))) {
1850                    while (fileScanner.hasNext()) {
1851                        String cname15 = fileScanner.nextLine();
1852                        String caddress15 = fileScanner.nextLine();
1853                        int category15;
1854
1855                        try {
1856                            // Attempt to parse category as an integer
1857                            category15 = Integer.parseInt(fileScanner.nextLine().trim());
1858                        } catch (NumberFormatException e) {
1859                            // Handle invalid input for category
1860                            System.out.println("Error Message: Invalid input for category. Please enter a valid integer.");
1861                            continue; // Move to the next iteration
1862                        }
1863
1864                        // Establish a database connection
1865                        try (Connection connection = DriverManager.getConnection(URL)) {
1866                            // Add a new customer from the file using a stored procedure
1867                            try (CallableStatement statement = connection.prepareCall(ENTER_NEW_CUSTOMER)) {
1868                                statement.setString(1, cname15);
1869                                statement.setString(2, caddress15);
1870                                statement.setInt(3, category15);
1871                                statement.registerOutParameter(4, Types.INTEGER);
1872                                statement.registerOutParameter(5, Types.NVARCHAR);
1873
1874                                // Execute the stored procedure to add the customer
1875                                statement.execute();
1876
1877                                // Retrieve error code and message from the stored procedure
1878                                int errorCode = statement.getInt(4);
1879                                String errorMessage = statement.getString(5);
1880
1881                                // Display success message or an error message, if any
1882                                if (errorCode != 0) {
1883                                    System.out.println("Error Code: " + errorCode);
1884                                    System.out.println("Error Message: " + errorMessage);
1885                                    System.out.println("Failed to add customer from file.");
1886                                } else {
1887                                    System.out.println("Customer added from file: " + cname15);
1888                                }
1889                            } catch (SQLException e) {
1890                                System.out.println("Error: " + e.getMessage());
1891                                System.out.println("Failed to add customer from file.");
1892                            }
1893                        } catch (SQLException e) {
1894                            System.out.println("Connection error: " + e.getMessage());
1895                        }
1896                    }
1897                } catch (FileNotFoundException e) {
1898                    System.out.println("File not found: " + inputFileName);
1899                }
1900                break;
1901
```

16. export.file:

```
1904
1905  // Case 16: Export customers to a data file based on category range
1906           // Prompt the user to enter the output file name and category range for exporting customers (case "16")
1907          case "16":
1908              System.out.println("Please enter the output file name:");
1909              String outputFileName = sc.next();
1910              System.out.println("Please enter the category range (e.g., 1-5):");
1911              String categoryRange = sc.next();
1912
1913              // Parse the category range
1914              String[] range = categoryRange.split("-");
1915              int minCategoryy = Integer.parseInt(range[0]);
1916              int maxCategoryy = Integer.parseInt(range[1]);
1917
1918              // Establish a database connection and export customers to a file
1919              try (Connection connection = DriverManager.getConnection(URL);
1920                   Statement statement = connection.createStatement();
1921                   ResultSet resultSet = statement.executeQuery("SELECT cname, caddress, category FROM Customer " +
1922                           "WHERE category BETWEEN " + minCategoryy + " AND " + maxCategoryy + " ORDER BY cname");
1923                   PrintWriter writer = new PrintWriter(outputFileName)) {
1924
1925                  while (resultSet.next()) {
1926                      String cname16 = resultSet.getString("cname");
1927                      String caddress16 = resultSet.getString("caddress");
1928                      int category16 = resultSet.getInt("category");
1929
1930                      // Write customer details to the output file
1931                      writer.println(cname16);
1932                      writer.println(caddress16);
1933                      writer.println(category16);
1934
1935                      System.out.println("Exported customer: " + cname16);
1936                  }
1937                  System.out.println("Customers exported to file: " + outputFileName);
1938              } catch (SQLException | FileNotFoundException e) {
1939                  // Handle SQL and file-related exceptions
1940                  System.out.println("Error: " + e.getMessage());
1941                  System.out.println("Failed to export customers to file.");
1942              }
1943              break;
1944
```

17. Quit option:

```
1956
1957           // Handle case "17" - Do nothing, the while loop will terminate upon the next iteration
1958          case "17":
1959              System.out.println("Exiting!"); // Display exit message
1960              break;
1961
```

Error Handling for 3 different queries:

Query 1:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
abc
Please enter customer address:
aaa
Please enter customer category:

Error Message: Invalid input for category. Please enter a valid integer.

Please select one of the options below:
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
A234
Please enter customer address:
ssdgh
Please enter customer category:
1
Connecting to the database...
Dispatching the stored procedure...
Error Code: 50002
Error Message: Invalid customer name. Name must be without special characters or numbers.
Failed to add customer. Returning to the main menu.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
abc
Please enter customer address:
ashj
Please enter customer category:
dhf
Error Message: Invalid input for category. Please enter a valid integer.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Swarupa
Please enter customer address:
ash
Please enter customer category:
3
Connecting to the database...
Dispatching the stored procedure...
Error Code: 2627
Error Message: A customer with the same name or data already exists.
Failed to add customer. Returning to the main menu.
```

Query 6:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1001
Please enter date the job commenced (YYYY-MM-DD):
2020-01-01
Dispatching the stored procedure for Job...
Error Code: 2627
Error Message: Violation of PRIMARY KEY constraint 'PK__Job__6E3281CA587D4168'. Cannot insert duplicate key in object 'dbo.Job'. The duplicate key value is (1001).
Failed to add Job record. Returning to the main menu.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1990
Please enter date the job commenced (YYYY-MM-DD):
299
Invalid date format. Please enter the date in yyyy-MM-dd format.
Please enter date the job commenced (YYYY-MM-DD):
```

Query 7:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
109
Error: Job with the specified job number does not exist. Returning to the main menu.

Please select one of the options below:
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1001
Please enter date the job completed (YYYY-MM-DD):
2000-11-11
Error Code: 1
Error Message: Error: Date of completion must be greater than the date of commencement.
Failed to update completion date.
Please enter date the job completed (YYYY-MM-DD):
```

Task 6: Java Program Execution

Testing of query 1: 5 queries

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


Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Swarupa
Please enter customer address:
123 Main Street, Anytown, USA
Please enter customer category:
2
Connecting to the database...
Dispatching the stored procedure...
Done. Customer added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Siva
Please enter customer address:
456 Oak Avenue, Cityville, CA
Please enter customer category:
5
Connecting to the database...
Dispatching the stored procedure...
Done. Customer added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Santhu
Please enter customer address:
789 Maple Lane, Suburbia, TX
Please enter customer category:
6
Connecting to the database...
Dispatching the stored procedure...
Done. Customer added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Malli
Please enter customer address:
101 Pine Road, Countryside, NY
Please enter customer category:
9
Connecting to the database...
Dispatching the stored procedure...
Done. Customer added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
1
Please enter customer name:
Rupa
Please enter customer address:
202 Cedar Boulevard, Metropolis, FL
Please enter customer category:
6
Connecting to the database...
Dispatching the stored procedure...
Done. Customer added.
```

Output of query 1:

dbo.Customer

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

cname	caddress	category
Malli	101 Pine Road, Countryside, NY	9
Rupa	202 Cedar Boulevard, Metropolis, FL	6
Santhu	789 Maple Lane, Suburbia, TX	6
Siva	456 Oak Avenue, Cityville, CA	5
Swarupa	123 Main Street, Anytown, USA	2

Testing of query 2: 5 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
2
Please enter department number:
101
Please enter department data:
data1
Connecting to the database...
Dispatching the stored procedure...
Done. Department added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
2
Please enter department number:
102
Please enter department data:
data2
Connecting to the database...
Dispatching the stored procedure...
Done. Department added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
2
Please enter department number:
103
Please enter department data:
data3
Connecting to the database...
Dispatching the stored procedure...
Done. Department added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
2
Please enter department number:
104
Please enter department data:
data4
Connecting to the database...
Dispatching the stored procedure...
Done. Department added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
2
Please enter department number:
105
Please enter department data:
data5
Connecting to the database...
Dispatching the stored procedure...
Done. Department added.
```

Output of query 2:

dbo.Department

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

department_no	department_data
101	data1
102	data2
103	data3
104	data4
105	data5

Testing of query 3: 10 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P01
Please enter process data:
pdata1
Please enter department number:
101
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
3
Please enter Cutting Type:
ctype1
Please enter Machine Type:
mtype1
Dispatching the stored procedure for Cut Process...
Done. Cut Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P02
Please enter process data:
pdata2
Please enter department number:
102
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
1
Please enter Fit Type:
ftype2
Dispatching the stored procedure for Fit Process...
Done. Fit Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P03
Please enter process data:
pdata3
Please enter department number:
103
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
2
Please enter Paint Type:
ptype3
Please enter Paint Method:
pmethod3
Dispatching the stored procedure for Paint Process...
Done. Paint Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P04
Please enter process data:
pdata4
Please enter department number:
104
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
1
Please enter Fit Type:
ftype4
Dispatching the stored procedure for Fit Process...
Done. Fit Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P05
Please enter process data:
pdata5
Please enter department number:
105
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
1
Please enter Fit Type:
ftype5
Dispatching the stored procedure for Fit Process...
Done. Fit Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P06
Please enter process data:
pdata6
Please enter department number:
105
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
2
Please enter Paint Type:
ptype6
Please enter Paint Method:
pmethod6
Dispatching the stored procedure for Paint Process...
Done. Paint Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P07
Please enter process data:
pdata7
Please enter department number:
105
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
2
Please enter Paint Type:
ptype7
Please enter Paint Method:
pmethod7
Dispatching the stored procedure for Paint Process...
Done. Paint Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P08
Please enter process data:
pdata8
Please enter department number:
104
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
3
Please enter Cutting Type:
ctype8
Please enter Machine Type:
mtype8
Dispatching the stored procedure for Cut Process...
Done. Cut Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P09
Please enter process data:
pdata9
Please enter department number:
103
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
3
Please enter Cutting Type:
ctype9
Please enter Machine Type:
mtype9
Dispatching the stored procedure for Cut Process...
Done. Cut Process added.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
3
Please enter process ID:
P10
Please enter process data:
pdata10
Please enter department number:
102
Connecting to the database...
Dispatching the stored procedure for Process...
Done. Process added.
Choose the type of process to add:
1. Fit Process
2. Paint Process
3. Cut Process
1
Please enter Fit Type:
ftype10
Dispatching the stored procedure for Fit Process...
Done. Fit Process added.
```

Outputs of query 3:

dbo.Process

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

process_id	process_data	department_no
P01	pdata1	101
P02	pdata2	102
P03	pdata3	103
P04	pdata4	104
P05	pdata5	105
P06	pdata6	105
P07	pdata7	105
P08	pdata8	104
P09	pdata9	103
P10	pdata10	102

dbo.Fit_process

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

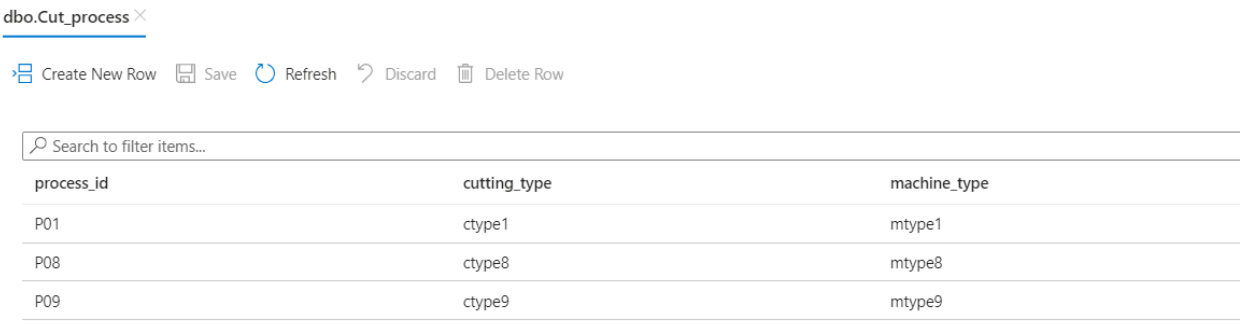
process_id	fit_type
P02	ftype2
P04	ftype4
P05	ftype5
P10	ftype10

dbo.Paint_process

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

process_id	paint_type	paint_method
P03	ptype3	pmethod3
P06	ptype6	pmethod6
P07	ptype7	pmethod7



dbo.Cut_process

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

process_id	cutting_type	machine_type
P01	ctype1	mtype1
P08	ctype8	mtype8
P09	ctype9	mtype9

Testing of query 4: 10 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A01
Please enter date ordered (YYYY-MM-DD):
2020-11-11
Please enter assembly details:
adetails1
Please enter customer name:
Siva
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A02
Please enter date ordered (YYYY-MM-DD):
2017-10-10
Please enter assembly details:
adetails2
Please enter customer name:
Siva
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P02
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P03
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
no


Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A03
Please enter date ordered (YYYY-MM-DD):
2018-09-09
Please enter assembly details:
adetails3
Please enter customer name:
Siva
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A04
Please enter date ordered (YYYY-MM-DD):
2019-12-19
Please enter assembly details:
adetails4
Please enter customer name:
Swarupa
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P05
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P06
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A05
Please enter date ordered (YYYY-MM-DD):
2018-11-26
Please enter assembly details:
adetails5
Please enter customer name:
Santhu
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A06
Please enter date ordered (YYYY-MM-DD):
2015-03-25
Please enter assembly details:
adetails6
Please enter customer name:
Santhu
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P07
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P08
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P09
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A07
Please enter date ordered (YYYY-MM-DD):
2021-04-14
Please enter assembly details:
adetails7
Please enter customer name:
Malli
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A08
Please enter date ordered (YYYY-MM-DD):
2018-02-03
Please enter assembly details:
adetails8
Please enter customer name:
Rupa
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A09
Please enter date ordered (YYYY-MM-DD):
2009-10-10
Please enter assembly details:
adetails9
Please enter customer name:
Malli
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P01
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P02
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
no
```

```
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
4
Please enter assembly ID:
A10
Please enter date ordered (YYYY-MM-DD):
2010-12-19
Please enter assembly details:
adetails10
Please enter customer name:
Santhu
Dispatching the stored procedure for Assembly...
Done. Assembly record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P05
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
yes
Please enter process ID for Manufacture:
P07
Dispatching the stored procedure for Manufacture...
Done. Manufacture record added.
Do you want to enter process for this assembly? (yes/no):
no
```

Outputs of query 4:

```
1   SELECT assembly_id, FORMAT(date_ordered, 'yyyy-MM-dd') AS date_ordered, assembly_details, cname
2   FROM Assembly;
3
```

dbo.Assembly | Query 3

Run | Cancel query | Save query | Export data as | Show all

Results | Messages

Search to filter items...

assembly_id	date_ordered	assembly_details	cname
A01	2020-11-11	adetails1	Siva
A02	2017-10-10	adetails2	Siva
A03	2018-09-09	adetails3	Siva
A04	2019-12-19	adetails4	Swarupa
A05	2018-11-26	adetails5	Santhu
A06	2015-03-25	adetails6	Santhu
A07	2021-04-14	adetails7	Malli
A08	2018-02-03	adetails8	Rupa
A09	2009-10-10	adetails9	Malli
A10	2010-12-19	adetails10	Santhu

dbo.manufacture

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

assembly_id	process_id	job_no
A02	P02	
A02	P03	
A04	P05	
A04	P06	
A06	P07	
A06	P08	
A06	P09	
A09	P01	
A09	P02	
A10	P05	
A10	P07	

Testing of query 5: 10 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
1111
Enter date of establishment (YYYY-MM-DD):
1990-03-27
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
1
Enter account details:
1000.00
Dispatching the stored procedure for Assembly Account...
Enter Assembly ID for Expenditure:
A01
Assembly Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
2222
Enter date of establishment (YYYY-MM-DD):
1991-04-24
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
2
Enter account details:
2000.00
Dispatching the stored procedure for Department Account...
Enter Department Number for Expenditure:
102
Department Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
3333
Enter date of establishment (YYYY-MM-DD):
1992-01-28
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
3
Enter account details:
3000.00
Dispatching the stored procedure for Process Account...
Enter Process ID for Expenditure:
P03
Process Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
4444
Enter date of establishment (YYYY-MM-DD):
1993-09-09
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
1
Enter account details:
4000.00
Dispatching the stored procedure for Assembly Account...
Enter Assembly ID for Expenditure:
A04
Assembly Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
5555
Enter date of establishment (YYYY-MM-DD):
1994-08-01
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
2
Enter account details:
5000.00
Dispatching the stored procedure for Department Account...
Enter Department Number for Expenditure:
105
Department Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
6666
Enter date of establishment (YYYY-MM-DD):
1995-02-26
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
3
Enter account details:
6000.00
Dispatching the stored procedure for Process Account...
Enter Process ID for Expenditure:
P06
Process Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
7777
Enter date of establishment (YYYY-MM-DD):
1996-07-06
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
1
Enter account details:
7000.00
Dispatching the stored procedure for Assembly Account...
Enter Assembly ID for Expenditure:
A07
Assembly Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
8888
Enter date of establishment (YYYY-MM-DD):
1997-06-07
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
1
Enter account details:
8000.00
Dispatching the stored procedure for Assembly Account...
Enter Assembly ID for Expenditure:
A07
Assembly Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
9999
Enter date of establishment (YYYY-MM-DD):
1998-12-11
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
3
Enter account details:
9000.00
Dispatching the stored procedure for Process Account...
Enter Process ID for Expenditure:
P03
Process Account and Expenditure records added successfully.
Account record added successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
5
Enter account number:
10000
Enter date of establishment (YYYY-MM-DD):
1999-11-12
Choose one of the following types of account:
1. Assembly Account
2. Department Account
3. Process Account
2
Enter account details:
10000.00
Dispatching the stored procedure for Department Account...
Enter Department Number for Expenditure:
102
Department Account and Expenditure records added successfully.
Account record added successfully.
```

Outputs of query 5:

```
1  SELECT assembly_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_1
2  FROM Assembly_account;
3
```

dbo.Assembly_account | Query 4

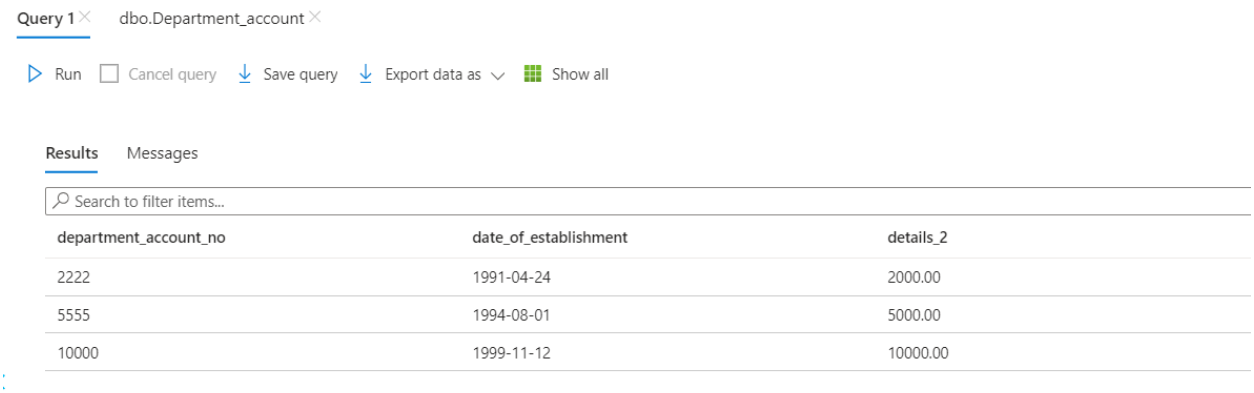
Run | Cancel query | Save query | Export data as | Show all

Results | Messages

Search to filter items...

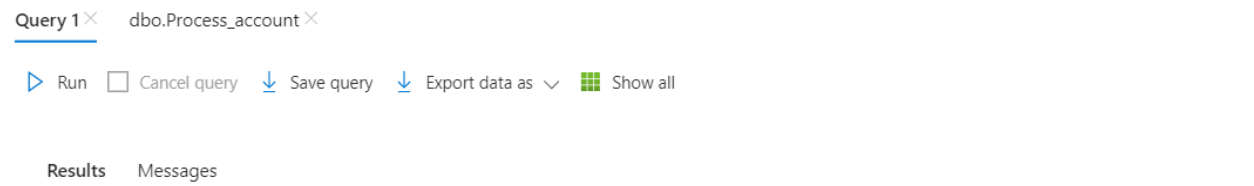
assembly_account_no	date_of_establishment	details_1
1111	1990-03-27	1000.00
4444	1993-09-09	4000.00
7777	1996-07-06	7000.00
8888	1997-06-07	8000.00

```
1  SELECT department_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_2
2  FROM Department_account;
3
```



department_account_no	date_of_establishment	details_2
2222	1991-04-24	2000.00
5555	1994-08-01	5000.00
10000	1999-11-12	10000.00

```
1  SELECT process_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_3
2  FROM Process_account;
3
```



process_account_no	date_of_establishment	details_3
3333	1992-01-28	3000.00
6666	1995-02-26	6000.00
9999	1998-12-11	9000.00

Testing of query 6: 10 queries

```
(11) Retrieve the processes through which a given assembly-id has passed so far and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1001
Please enter date the job commenced (YYYY-MM-DD):
2020-01-01
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
3
Dispatching the stored procedure for Cut Job...
Cut Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A02
Please enter process_id:
P02
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A09
Please enter process_id:
P02
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1002
Please enter date the job commenced (YYYY-MM-DD):
2019-02-02
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
2
Dispatching the stored procedure for Paint Job...
Paint Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A02
Please enter process_id:
P03
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A10
Please enter process_id:
P05
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
(11) Retrieve the processes through which a given assembly-id has passed so far and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1003
Please enter date the job commenced (YYYY-MM-DD):
2018-03-03
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
2
Dispatching the stored procedure for Paint Job...
Paint Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A04
Please enter process_id:
P05
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A09
Please enter process_id:
P01
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1004
Please enter date the job commenced (YYYY-MM-DD):
2017-04-04
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
3
Dispatching the stored procedure for Cut Job...
Cut Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A10
Please enter process_id:
P07
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1005
Please enter date the job commenced (YYYY-MM-DD):
2016-05-05
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
1
Dispatching the stored procedure for Fit Job...
Fit Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A06
Please enter process_id:
P07
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no

(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1006
Please enter date the job commenced (YYYY-MM-DD):
2015-06-06
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
1
Dispatching the stored procedure for Fit Job...
Fit Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A06
Please enter process_id:
P08
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A06
Please enter process_id:
P09
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1007
Please enter date the job commenced (YYYY-MM-DD):
2014-07-07
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
1
Dispatching the stored procedure for Fit Job...
Fit Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1008
Please enter date the job commenced (YYYY-MM-DD):
2013-08-08
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
2
Dispatching the stored procedure for Paint Job...
Paint Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
no
```

```
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1009
Please enter date the job commenced (YYYY-MM-DD):
2012-09-09
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
2
Dispatching the stored procedure for Paint Job...
Paint Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
yes
Please enter assembly_id:
A04
Please enter process_id:
P06
Dispatching the stored procedure for Manufacture...
Manufacture details added successfully.
Do you want to insert more assembly_id and process_id pairs for this job? (yes/no):
no
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
6
Please enter job number:
1010
Please enter date the job commenced (YYYY-MM-DD):
2021-10-10
Dispatching the stored procedure for Job...
Job record added successfully.
Choose the type of job:
1. Fit Job
2. Paint Job
3. Cut Job
3
Dispatching the stored procedure for Cut Job...
Cut Job details added successfully.
Do you want to insert assembly_id and process_id pairs for this job? (yes/no):
no
```

Outputs of query 6:

```
1  SELECT job_no, FORMAT(date_of_commencement, 'yyyy-MM-dd') AS date_of_commencement, date_of_completion
2  FROM Job;
3
```

dbo.Job | Query 2

Run | Cancel query | Save query | Export data as | Show all

Results | Messages

Search to filter items...

job_no	date_of_commencement	date_of_completion
1001	2020-01-01	
1002	2019-02-02	
1003	2018-03-03	
1004	2017-04-04	
1005	2016-05-05	
1006	2015-06-06	
1007	2014-07-07	
1008	2013-08-08	
1009	2012-09-09	
1010	2021-10-10	

dbo.Fit_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	fit_labor_time
1005	
1006	
1007	

dbo.Paint_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	color	volume	paint_labor_time
1002			
1003			
1008			
1009			

UJWALA VASIREDDY
113634633
INDIVIDUAL PROJECT: A JOB-SHOP ACCOUNTING SYSYTEM

dbo.Cut_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	type_of_machine	time_of_machine	material_used	cut_labor_time
1001				
1004				
1010				

dbo.manufacture

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

assembly_id	process_id	job_no
A02	P02	1001
A09	P02	1001
A02	P03	1002
A10	P05	1002
A04	P05	1003
A09	P01	1003
A10	P07	1004
A06	P07	1005
A06	P08	1006
A06	P09	1006
A04	P06	1009

Testing of query 7: 10 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1001
Please enter date the job completed (YYYY-MM-DD):
2000-11-11
Error Code: 1
Error Message: Error: Date of completion must be greater than the date of commencement.
Failed to update completion date.
Please enter date the job completed (YYYY-MM-DD):
2021-01-01
Completion date updated successfully.
This is a Cut Job:
Please enter the new type of machine:
type1
Please enter the new time of machine (HH:mm:ss):
01:01:01
Please enter the new material used:
material1
Please enter the new cut labor time (HH:mm:ss):
10:10:10
Cut Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1002
Please enter date the job completed (YYYY-MM-DD):
2020-02-02
Completion date updated successfully.
This is a Paint Job:
Please enter the new color:
White
Please enter the new volume:
243.89
Please enter the new paint labor time (HH:mm:ss):
20:20:20
Paint Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1003
Please enter date the job completed (YYYY-MM-DD):
2019-03-03
Completion date updated successfully.
This is a Paint Job:
Please enter the new color:
Black
Please enter the new volume:
4599.999
Please enter the new paint labor time (HH:mm:ss):
13:03:03
Paint Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1004
Please enter date the job completed (YYYY-MM-DD):
2018-04-04
Completion date updated successfully.
This is a Cut Job:
Please enter the new type of machine:
type2
Please enter the new time of machine (HH:mm:ss):
02:02:02
Please enter the new material used:
material2
Please enter the new cut labor time (HH:mm:ss):
20:20:20
Cut Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1005
Please enter date the job completed (YYYY-MM-DD):
2017-05-05
Completion date updated successfully.
This is a Fit Job:
Please enter the new fit labor time (HH:mm:ss):
15:15:05
Fit Job labor time updated successfully.


Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1006
Please enter date the job completed (YYYY-MM-DD):
2016-06-06
Completion date updated successfully.
This is a Fit Job:
Please enter the new fit labor time (HH:mm:ss):
16:06:06
Fit Job labor time updated successfully.


Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1007
Please enter date the job completed (YYYY-MM-DD):
2015-07-07
Completion date updated successfully.
This is a Fit Job:
Please enter the new fit labor time (HH:mm:ss):
17:07:07
Fit Job labor time updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1008
Please enter date the job completed (YYYY-MM-DD):
2014-08-08
Completion date updated successfully.
This is a Paint Job:
Please enter the new color:
Blue
Please enter the new volume:
421.678
Please enter the new paint labor time (HH:mm:ss):
18:09:09
Paint Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1009
Please enter date the job completed (YYYY-MM-DD):
2013-09-09
Completion date updated successfully.
This is a Paint Job:
Please enter the new color:
Black
Please enter the new volume:
67.3
Please enter the new paint labor time (HH:mm:ss):
19:09:09
Paint Job details updated successfully.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
7
Please enter the job number:
1010
Please enter date the job completed (YYYY-MM-DD):
2012-10-10
Error Code: 1
Error Message: Error: Date of completion must be greater than the date of commencement.
Failed to update completion date.
Please enter date the job completed (YYYY-MM-DD):
2023-03-03
Completion date updated successfully.
This is a Cut Job:
Please enter the new type of machine:
type3
Please enter the new time of machine (HH:mm:ss):
03:03:30
Please enter the new material used:
material3
Please enter the new cut labor time (HH:mm:ss):
13:30:30
Cut Job details updated successfully.
```

Outputs of query 7:

```
1   SELECT job_no, FORMAT(date_of_commencement, 'yyyy-MM-dd') AS date_of_commencement,
2   FORMAT(date_of_completion, 'yyyy-MM-dd') AS date_of_completion
3   from Job;
```

dbo.Job | Query 4

Run | Cancel query | Save query | Export data as | Show all

Results | Messages

Search to filter items...

job_no	date_of_commencement	date_of_completion
1001	2020-01-01	2021-01-01
1002	2019-02-02	2020-02-02
1003	2018-03-03	2019-03-03
1004	2017-04-04	2018-04-04
1005	2016-05-05	2017-05-05
1006	2015-06-06	2016-06-06
1007	2014-07-07	2015-07-07
1008	2013-08-08	2014-08-08
1009	2012-09-09	2013-09-09
1010	2021-10-10	2023-03-03

dbo.Fit_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	fit_labor_time
1005	15:15:05
1006	16:06:06
1007	17:07:07

dbo.Paint_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	color	volume	paint_labor_time
1002	White	243.89	20:20:20
1003	Black	4599.999	13:03:03
1008	Blue	421.678	18:09:09
1009	Black	67.3	19:09:09

dbo.Cut_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	type_of_machine	time_of_machine	material_used	cut_labor_time
1001	type1	01:01:01	material1	10:10:10
1004	type2	02:02:02	material2	20:20:20
1010	type3	03:03:30	material3	13:30:30

Testing of query 8: 10 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
1110
Please enter sup-cost:
10.10
Do you want to enter assembly account? (yes/no):
yes
Please enter assembly account no:
1111
Do you want to enter department account? (yes/no):
yes
Please enter department account no:
2222
Do you want to enter process account? (yes/no):
no
Connecting to the database...
Updating assembly account.
Updating department account.
Dispatching the stored procedure for Transaction...
Cost updated for 2 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
2220
Please enter sup-cost:
20.20
Do you want to enter assembly account? (yes/no):
no
Do you want to enter department account? (yes/no):
no
Do you want to enter process account? (yes/no):
no
Connecting to the database...
Dispatching the stored procedure for Transaction...
No cost update.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
3330
Please enter sup-cost:
30.30
Do you want to enter assembly account? (yes/no):
yes
Please enter assembly account no:
4444
Do you want to enter department account? (yes/no):
no
Do you want to enter process account? (yes/no):
yes
Please enter process account no:
3333
Connecting to the database...
Updating assembly account.
Updating process account.
Dispatching the stored procedure for Transaction...
Cost updated for 2 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
4440
Please enter sup-cost:
40.40
Do you want to enter assembly account? (yes/no):
yes
Please enter assembly account no:
7777
Do you want to enter department account? (yes/no):
yes
Please enter department account no:
5555
Do you want to enter process account? (yes/no):
yes
Please enter process account no:
6666
Connecting to the database...
Updating assembly account.
Updating department account.
Updating process account.
Dispatching the stored procedure for Transaction...
Cost updated for 3 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
5550
Please enter sup-cost:
50.50
Do you want to enter assembly account? (yes/no):
no
Do you want to enter department account? (yes/no):
no
Do you want to enter process account? (yes/no):
yes
Please enter process account no:
6666
Connecting to the database...
Updating process account.
Dispatching the stored procedure for Transaction...
Cost updated for 1 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
6660
Please enter sup-cost:
60.60
Do you want to enter assembly account? (yes/no):
yes
Please enter assembly account no:
8888
Do you want to enter department account? (yes/no):
yes
Please enter department account no:
5555
Do you want to enter process account? (yes/no):
no
Connecting to the database...
Updating assembly account.
Updating department account.
Dispatching the stored procedure for Transaction...
Cost updated for 2 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
7770
Please enter sup-cost:
70.70
Do you want to enter assembly account? (yes/no):
yes
Please enter assembly account no:
8888
Do you want to enter department account? (yes/no):
no
Do you want to enter process account? (yes/no):
no
Connecting to the database...
Updating assembly account.
Dispatching the stored procedure for Transaction...
Cost updated for 1 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
8880
Please enter sup-cost:
80.80
Do you want to enter assembly account? (yes/no):
no
Do you want to enter department account? (yes/no):
no
Do you want to enter process account? (yes/no):
yes
Please enter process account no:
3333
Connecting to the database...
Updating process account.
Dispatching the stored procedure for Transaction...
Cost updated for 1 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
9990
Please enter sup-cost:
90.90
Do you want to enter assembly account? (yes/no):
no
Do you want to enter department account? (yes/no):
yes
Please enter department account no:
10000
Do you want to enter process account? (yes/no):
no
Connecting to the database...
Updating department account.
Dispatching the stored procedure for Transaction...
Cost updated for 1 account(s).
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
8
Please enter transaction-no:
1001
Please enter sup-cost:
100.10
Do you want to enter assembly account? (yes/no):
no
Do you want to enter department account? (yes/no):
yes
Please enter department account no:
2222
Do you want to enter process account? (yes/no):
yes
Please enter process account no:
9999
Connecting to the database...
Updating department account.
Updating process account.
Dispatching the stored procedure for Transaction...
Cost updated for 2 account(s).
```

Outputs of query 8:

dbo.Cost

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

transaction_no	sup_cost	assembly_account_no	department_account_no	process_account_no
1001	100.10		2222	9999
1110	10.10	1111	2222	
2220	20.20			
3330	30.30	4444		3333
4440	40.40	7777	5555	6666
5550	50.50			6666
6660	60.60	8888	5555	
7770	70.70	8888		
8880	80.80			3333
9990	90.90		10000	

```
1  SELECT assembly_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_1
2  FROM Assembly_account;
3
```

dbo.Assembly_account | Query 16

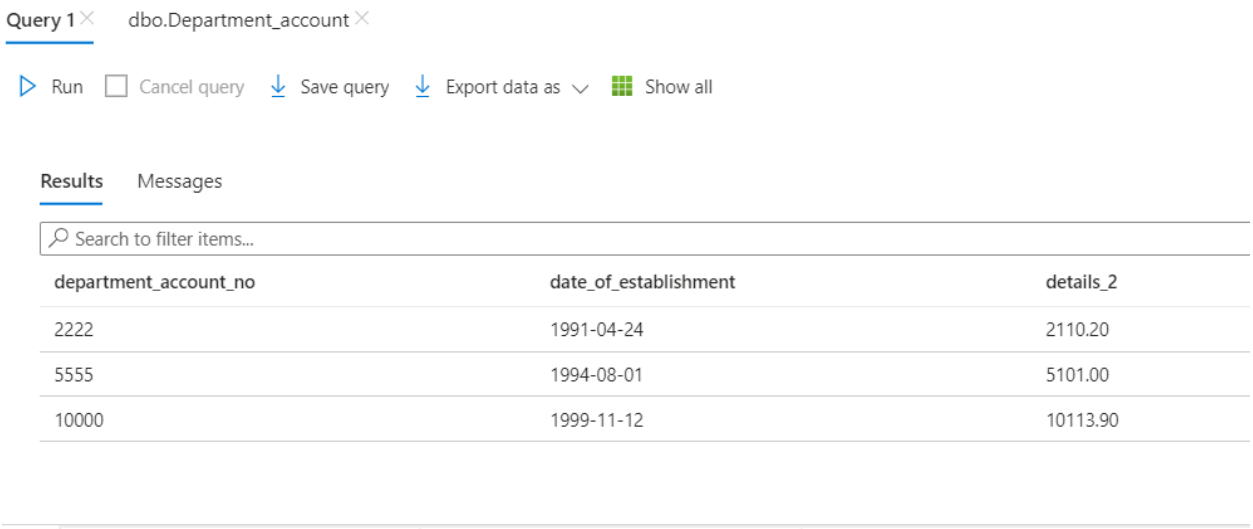
Run | Cancel query | Save query | Export data as | Show all

Results | Messages

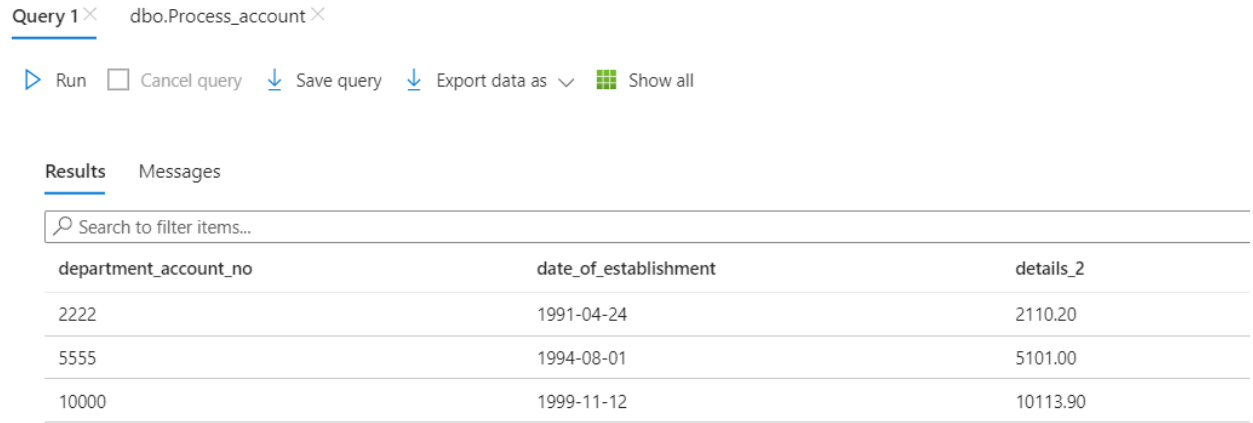
Search to filter items...

assembly_account_no	date_of_establishment	details_1
1111	1990-03-27	1010.10
4444	1993-09-09	4030.30
7777	1996-07-06	7040.40
8888	1997-06-07	8131.30

```
1  SELECT department_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_2
2  FROM department_account;
3
```




```
1   SELECT process_account_no, FORMAT(date_of_establishment, 'yyyy-MM-dd') AS date_of_establishment, details_3
2   FROM process_account;
3
```



Testing and output of query 9: 3 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
9
Enter an assembly id to retrieve total cost:
A01
Total cost incurred on assembly id A01: 10.1
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
9
Enter an assembly id to retrieve total cost:
A04
Total cost incurred on assembly_id A04: 30.3
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
9
Enter an assembly id to retrieve total cost:
A07
Total cost incurred on assembly_id A07: 171.7
```

Testing and output of query 10: 3 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
10
Enter department number:
101
Enter the date (YYYY-MM-DD):
2021-01-01
Total labor time in department 101 for jobs completed on 2021-01-01: 0.0 minutes
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
10
Enter department number:
105
Enter the date (YYYY-MM-DD):
2017-05-05
Total labor time in department 105 for jobs completed on 2017-05-05: 915.0 minutes
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
10
Enter department number:
103
Enter the date (YYYY-MM-DD):
2020-02-02
Total labor time in department 103 for jobs completed on 2020-02-02: 1220.0 minutes
```

Testing and output of query 11: 3 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
11
Enter assembly_id to retrieve processes:
A10
Processes for assembly_id A10:
Process ID: P07, Department: data5
Process ID: P05, Department: data5
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
11
Enter assembly_id to retrieve processes:
A02
Processes for assembly_id A02:
Process ID: P03, Department: data3
Process ID: P02, Department: data2
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
11
Enter assembly_id to retrieve processes:
A06
Processes for assembly_id A06:
Process ID: P08, Department: data4
Process ID: P09, Department: data3
Process ID: P07, Department: data5
```

Testing and output of query 12: 3 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
12
Enter the minimum category:
3
Enter the maximum category:
8
Customers in the category range 3 to 8:
Rupa - 202 Cedar Boulevard, Metropolis, FL\nSanthu - 789 Maple Lane, Suburbia, TX\nSiva - 456 Oak Avenue, Cityville, CA
```

```
Color changed successfully for paint job 1002

Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
12
Enter the minimum category:
1
Enter the maximum category:
4
Customers in the category range 1 to 4:
Swarupa - 123 Main Street, Anytown, USA
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
12
Enter the minimum category:
3
Enter the maximum category:
4
Customers in the category range 3 to 4:
null
```

Testing of query 13: 3 queries

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
13
Enter the range for job numbers (minJobNo and maxJobNo):
1002
1004
Cut-jobs within the specified range have been deleted.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
13
Enter the range for job numbers (minJobNo and maxJobNo):
1001
1002
Cut-jobs within the specified range have been deleted.
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
13
Enter the range for job numbers (minJobNo and maxJobNo):
1002
1004
Cut-jobs within the specified range have been deleted.
```

Output of query 13:

dbo.Paint_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	color	volume	paint_labor_time
1002	Silver	243.89	20:20:20
1003	Black	4599.999	13:03:03
1008	Red	421.678	18:09:09
1009	Blue	67.3	19:09:09

Testing of query 14:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
14
Enter job number for paint job:
1002
Enter new color:
Silver
Color changed successfully for paint job 1002
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
14
Enter job number for paint job:
1008
Enter new color:
Red
Color changed successfully for paint job 1008
```

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
14
Enter job number for paint job:
1009
Enter new color:
Blue
Color changed successfully for paint job 1009
```

Outputs of query 14:

dbo.Cut_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

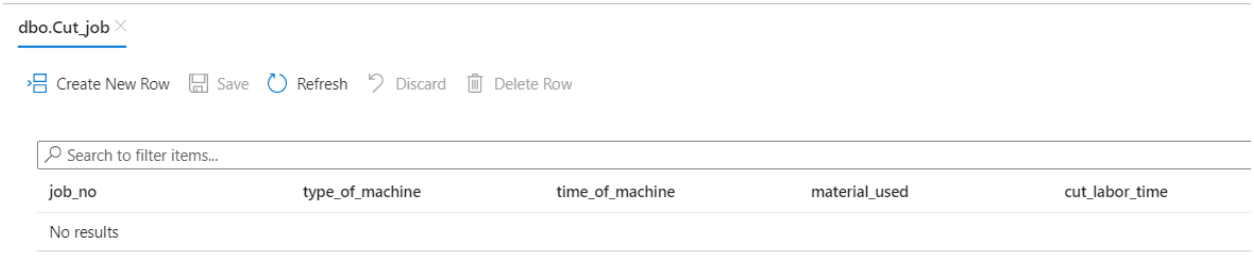
job_no	type_of_machine	time_of_machine	material_used	cut_labor_time
1004	type2	02:02:02	material2	20:20:20
1010	type3	03:03:30	material3	13:30:30

dbo.Cut_job

Create New Row | Save | Refresh | Discard | Delete Row

Search to filter items...

job_no	type_of_machine	time_of_machine	material_used	cut_labor_time
1010	type3	03:03:30	material3	13:30:30



Testing of query 15:

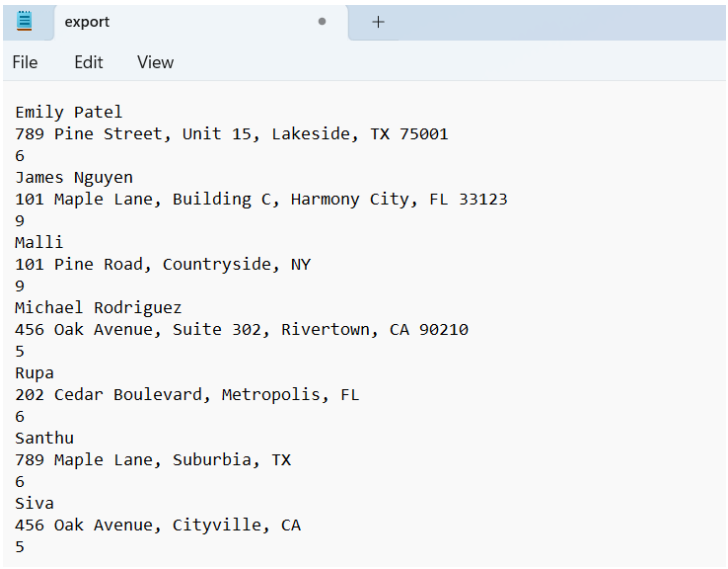
```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
15
Please enter the input file name:
import.txt
Customer added from file: Alice Johnson
Customer added from file: Michael Rodriguez
Customer added from file: Emily Patel
Customer added from file: James Nguyen
Customer added from file: Sophia Kim
```

Output of query 15:

Testing of query 16:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
16
Please enter the output file name:
export.txt
Please enter the category range (like 1-5):
5-9
Exported customer: Emily Patel
Exported customer: James Nguyen
Exported customer: Malli
Exported customer: Michael Rodriguez
Exported customer: Rupa
Exported customer: Santhu
Exported customer: Siva
Customers exported to file: export.txt
```

Output for query 16:



Testing and output for query 17:

```
Please select one of the options below:
(1) Enter a new Customer:
(2) Enter a new Department:
(3) Enter a new Process and information related to the type:
(4) Enter a new Assembly and associate it with one or more processes:
(5) Enter a new Account and associate it with the one which it is applicable:
(6) Enter a new Job:
(7) Enter the date of completion for a job and information of its type:
(8) Enter a new Cost Transaction and update all the affected accounts:
(9) Retrieve the total cost incurred on an assembly-id:
(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 and the department responsible for each process:
(12) Retrieve the customers whose category is in a given range:
(13) Delete all cut-jobs whose job-no is in a given range:
(14) Change the color of a given paint job:
(15) Import: enter new customers from a data file until the file is empty:
(16) Export: Retrieve the customers whose category is in a given range and output them to a data file:
(17) Quit
17
Exiting!
```

Task 7: Web Database Application and its Execution

DataHandler.java:

```
package jsp_azure;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class DataHandler {
    private Connection conn;
// Azure SQL connection credentials
    private String server = "vasi0011-sql-server.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "vasi0011";
    private String password = "*****";
// Resulting connection string
    final private String url =
            String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginT
            server, database, username, password);
// Initialize and save the database connection
    // Initialize and save the database connection
    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;
        }
        this.conn = DriverManager.getConnection(url);
    }

    // Return the result of selecting everything from the Customer table
    public ResultSet getAllCustomers() throws SQLException {
        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery = "SELECT * FROM Customer;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Execute the query
        return stmt.executeQuery();
    }

    // Return the result of selecting Customers with their category in the given range from Customer table
    public ResultSet retrieveCustomers(int lower_b, int upper_b) throws SQLException {
        getDBConnection(); // Prepare the database connection
```

```
    // Return the result of selecting Customers with their category in the given range from Customer table
    public ResultSet retrieveCustomers(int lower_b, int upper_b) throws SQLException {
        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery = "SELECT cname AS name, category FROM Customer" +
                " WHERE category >= ? AND category <= ? ORDER BY 1 ;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setInt(1, lower_b);
        stmt.setInt(2, upper_b);

        // Execute the query
        return stmt.executeQuery();
    }

    // Inserts a record into the Customer table with the given attribute values
    public boolean addCustomer(String cname, String caddress, int category) throws SQLException {
        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement for adding a new customer
        final String sqlQuery = "INSERT INTO Customer (cname, caddress, category) VALUES (?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, cname);
        stmt.setString(2, caddress);
        stmt.setInt(3, category);

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

```

welcome.jsp:

```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>Welcome to Job-Shop Accounting</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 20px;
            padding: 20px;
            background-color: #f4f4f4;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        h1 {
            color: #333;
        }

        .card-container {
            display: flex;
            justify-content: space-around;
            margin-top: 20px;
        }

        .card {
            width: 400px;
            padding: 20px;
            background-color: #eee; /* Gray background color */
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
            text-align: center;
        }

        h2 {
            color: #333;
        }
        h2 {
            color: #333;
        }

        a {
            display: block;
            margin: 10px 0;
            padding: 10px;
            background-color: #3498db;
            color: #fff;
            text-decoration: none;
            text-align: center;
            border-radius: 5px;
            transition: background-color 0.3s;
        }

        a:hover {
            background-color: #2980b9;
        }
    </style>
</head>
<body>
    <h1>Welcome to Job-Shop Accounting</h1>

    <div class="card-container">
        <!-- Add New Customer Card -->
        <div class="card">
            <h2>Add New Customer</h2>
            <a href="add_new_customer_form.jsp">Go to Add New Customer</a>
        </div>

        <!-- Retrieve Customers Card -->
        <div class="card">
            <h2>Retrieve Customers</h2>
            <a href="retrieve_customer_form.jsp">Go to Retrieve Customers</a>
        </div>
    </div>
</body>
</html>
```

add_new_customer.jsp:

```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>Add New Customer Result</title>
</head>
<body>
    <%@page import="jsp_azure.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>

    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler1 = new DataHandler();

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

    /*
    * If the user hasn't filled out all the cname, address and category. This is very simple checking.
    */
    if (cname == null || address == null || category == null || cname.trim().isEmpty() || address.trim().isEmpty() || category.trim().isEmpty()) {
        response.sendRedirect("add_new_customer_form.jsp");
    } else {
        try {
            int category1 = Integer.parseInt(category);

            // Now perform the query with the data from the form.
            boolean success = handler1.addCustomer(cname, address, category1);

            if (!success) {
    %>
                <h2>There was a problem inserting the customer.</h2>
    <%
            } else {
    %>
```

```jsp
            } else {
     %>
                <h2>The Customer Details:</h2>
                <ul>
                    <li>Customer Name: <%=cname%></li>
                    <li>Address: <%=address%></li>
                    <li>Category: <%=category1%></li>
                </ul>
                <h2>Was successfully inserted.</h2>
                <a href="get_all_customers.jsp">See all Customers.</a>
     <%
            }
        } catch (NumberFormatException e) {
     %>
            <h2>Error: Invalid input for category. Please enter a valid integer.</h2>
            <a href="add_new_customer_form.jsp">Go back to the form</a>
     <%
        }
    }
     %>
</body>
</html>
```

add_new_customer_forms.jsp:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Add Customer</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f4f4f4;
        }

        .card {
            width: 70%;
            background-color: #fff;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
            padding: 20px;
        }

        h2 {
            text-align: center;
        }

        table {
            margin: auto;
            border-collapse: collapse;
            width: 100%;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }

        input[type="text"] {
            width: calc(100% - 16px);
            padding: 8px;
```

```
            padding: 8px;
            box-sizing: border-box;
        }

        input[type="reset"], input[type="submit"] {
            padding: 10px;
            width: calc(50% - 6px);
            box-sizing: border-box;
        }

        input[type="reset"] {
            background-color: #f44336;
            color: white;
            border: none;
        }

        input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        div {
            text-align: center;
        }

        .error {
            color: #f44336;
        }
    </style>
</head>
<body>
    <div class="card">
        <h2>Add Customer</h2>

        <!-- Form for collecting user input for the new customer record. -->
        <form action="add_new_customer.jsp" method="post">
            <!-- The form organized in an HTML table for better clarity. -->
            <table>
                <tr>
                    <th colspan="2">Enter the Customer Data:</th>
                </tr>
                <tr>
```

```
                <tr>
                    <td>Customer Name:</td>
                    <td>
                        <div>
                            <input type="text" name="cname" required>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>Customer Address:</td>
                    <td>
                        <div>
                            <input type="text" name="address" required>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>Category:</td>
                    <td>
                        <div>
                            <input type="text" name="category" required>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td colspan="2">
                        <div>
                            <input type="reset" value="Clear">
                            <input type="submit" value="Insert">
                        </div>
                    </td>
                </tr>
            </table>
        </form>

        <!-- Display error messages, if any -->
        <%
            String errorMessage = (String)request.getAttribute("errorMessage");
            if (errorMessage != null && !errorMessage.isEmpty()) {
        %>
                <div class="error"><%=errorMessage%></div>
        <%

                    <td colspan="2">
                        <div>
                            <input type="reset" value="Clear">
                            <input type="submit" value="Insert">
                        </div>
                    </td>
                </tr>
            </table>
        </form>

        <!-- Display error messages, if any -->
        <%
            String errorMessage = (String)request.getAttribute("errorMessage");
            if (errorMessage != null && !errorMessage.isEmpty()) {
        %>
                <div class="error"><%=errorMessage%></div>
        <%
            }
        %>
    </div>
</body>
</html>
```

get_all_customers.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Customers</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f4f4f4;
        }

        .card {
            width: 70%;
            background-color: #fff;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
            padding: 20px;
        }

        h4 {
            margin: 0;
        }

        table {
            margin: auto;
            border-collapse: collapse;
            width: 100%;
            margin-top: 20px;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }
    </style>
</head>
```

```
    </style>
</head>
<body>
    <%@page import="jsp_azure.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>

    <%
        // We instantiate the data handler here, and get all the customers from the database
        final DataHandler handler = new DataHandler();
        final ResultSet customers = handler.getAllCustomers();
    %>

    <div class="card">
        <h2>All Customers</h2>

        <!-- The table for displaying all the customer records -->
        <table>
            <tr> <!-- The table headers row -->
                <th align="center"><h4>Customer</h4></th>
                <th align="center"><h4>Address</h4></th>
                <th align="center"><h4>Category</h4></th>
            </tr>

            <%
                while (customers.next()) { // For each Customer record returned...
                    // Extract the attribute values for every row returned
                    final String cname = customers.getString("cname");
                    final String caddress = customers.getString("caddress");
                    final String category = customers.getString("category");

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

retrieve_customers.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Customers</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        h4 {
            margin: 0;
        }

        table {
            margin: auto;
            border-collapse: collapse;
            width: 50%;
            margin-top: 20px;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }
    </style>
</head>
<body>
    <%@page import="jsp_azure.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>

    <%
        // We instantiate the data handler here and get customers from the database
        final DataHandler handler = new DataHandler();

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

        if (lower_b == null || upper_b == null || lower_b.trim().isEmpty() || upper_b.trim().isEmpty()) {
            response.sendRedirect("retrieve_customer_form.jsp");
```

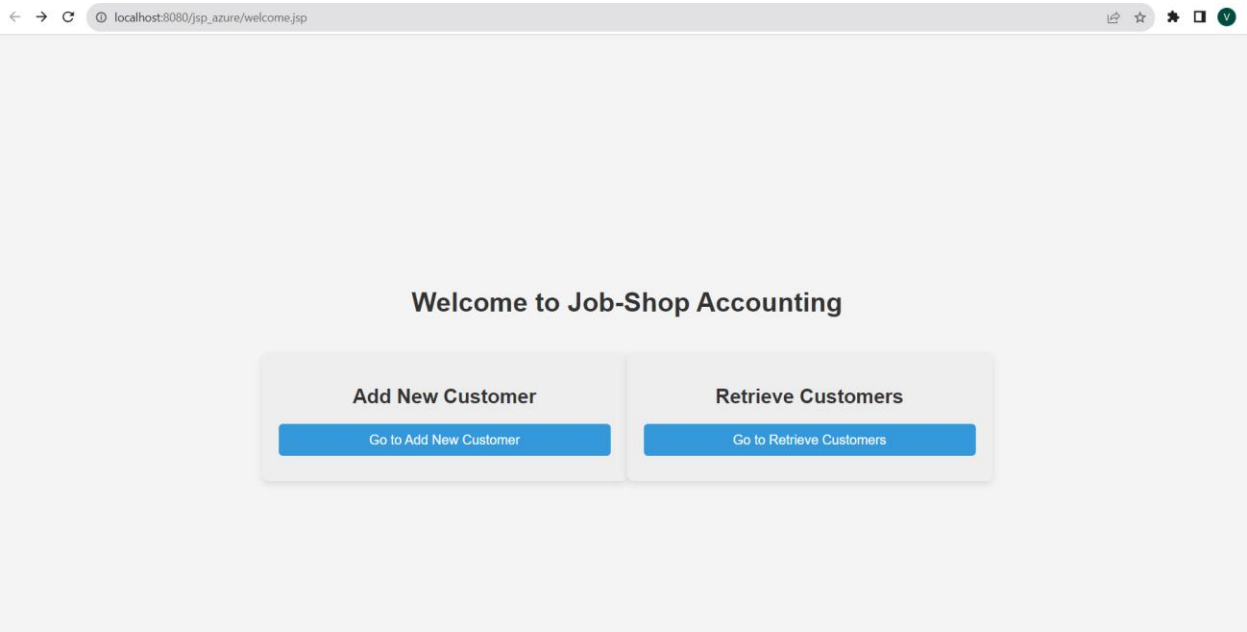
```jsp
            response.sendRedirect("retrieve_customer_form.jsp");
        } else {
            try {
                int lower_b1 = Integer.parseInt(lower_b);
                int upper_b1 = Integer.parseInt(upper_b);

                // Now perform the query with the data from the form.
                final ResultSet customers = handler.retrieveCustomers(lower_b1, upper_b1);
    %>

                <!-- The table for displaying all the Customer records -->
                <table>
                    <tr> <!-- The table headers row -->
                        <th align="center"><h4>Customer</h4></th>
                        <th align="center"><h4>Category</h4></th>
                    </tr>

                    <%
                        while (customers.next()) { // For each Customer record returned...
                            // Extract the attribute values for every row returned
                            final String cname = customers.getString("name");
                            final String category = customers.getString("category");

                            out.println("<tr>"); // Start printing out the new table row
                            out.println( // Print each attribute value
                                    "<td align=\"center\">" + cname +
                                    "</td><td align=\"center\"> " + category + "</td>");
                            out.println("</tr>");
                        }
                    }
                 catch (NumberFormatException e) {
                    // Handle the case where the input is not a valid integer
    %>
                    <h2>Error: Invalid input for category range. Please enter valid integers.</h2>
                    <a href="retrieve_customer_form.jsp">Go back to the form</a>
    <%
                }
            }
    %>
    </table>
</body>
</html>
```

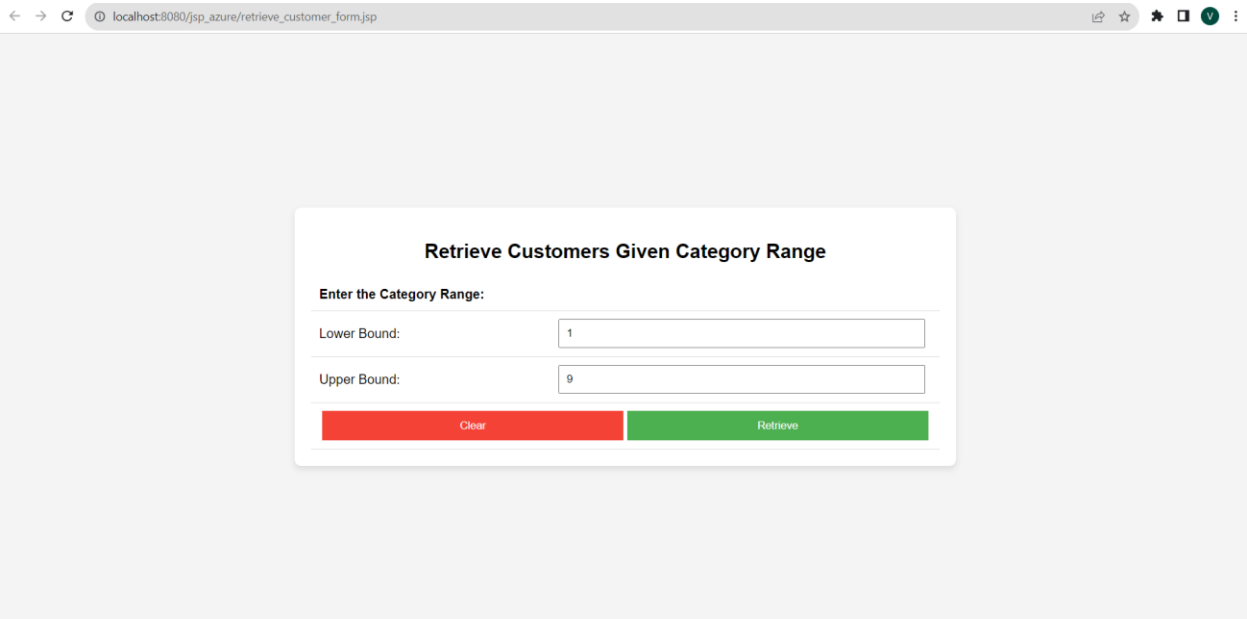
retrieve_customer_form.jsp

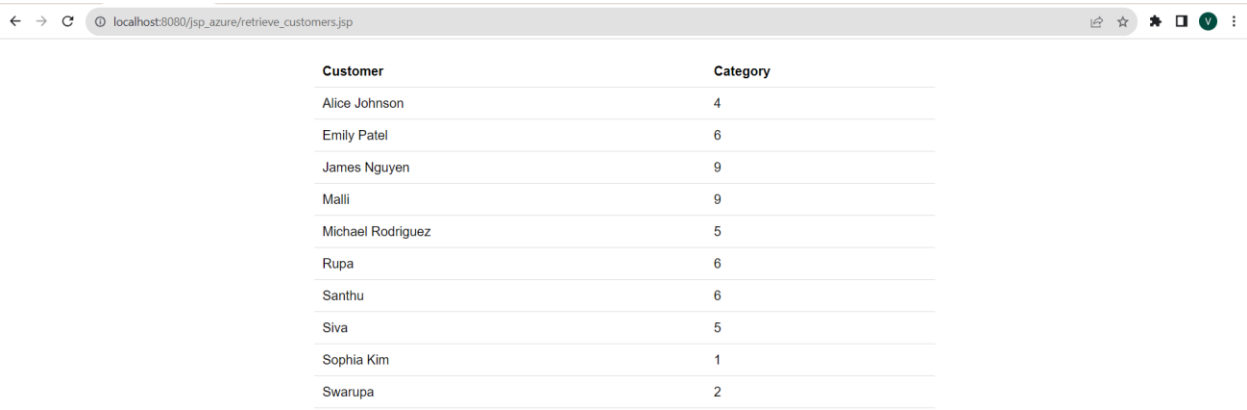
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Retrieve Customers Given Category Range</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f4f4f4;
        }

        .card {
            width: 50%;
            background-color: #fff;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
            padding: 20px;
        }

        h2 {
            text-align: center;
        }

        table {
            margin: auto;
            border-collapse: collapse;
            width: 100%;
            margin-top: 20px;
        }

        th, td {
            padding: 10px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }

        input[type="text"] {
            width: calc(100% - 16px);
            width: calc(100% - 16px);
            padding: 8px;
            box-sizing: border-box;
        }

        input[type="reset"], input[type="submit"] {
            padding: 10px;
            width: calc(50% - 6px);
            box-sizing: border-box;
        }

        input[type="reset"] {
            background-color: #f44336;
            color: white;
            border: none;
        }

        input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            border: none;
        }

        div {
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="card">
        <h2>Retrieve Customers Given Category Range</h2>

        <!-- Form for collecting user input for retrieving customer records -->
        <form action="retrieve_customers.jsp" method="post">
            <!-- The form organized in an HTML table for better clarity. -->
            <table>
                <tr>
                    <th colspan="2">Enter the Category Range:</th>
                </tr>
                <tr>
                    <td>Lower Bound:</td>
                    <td>
                        <div>
```

```
                    <td>Lower Bound:</td>
                    <td>
                        <div>
                            <input type="text" name="lower_b" required>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>Upper Bound:</td>
                    <td>
                        <div>
                            <input type="text" name="upper_b" required>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td colspan="2">
                        <div>
                            <input type="reset" value="Clear">
                            <input type="submit" value="Retrieve">
                        </div>
                    </td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

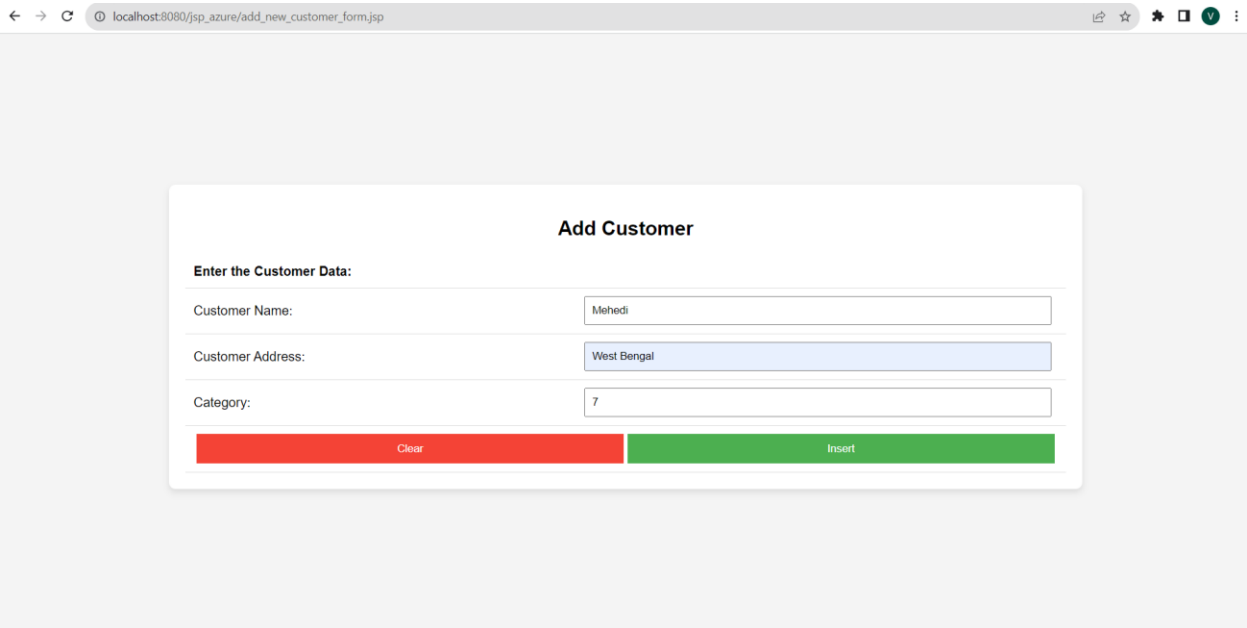
Outputs of the web application:

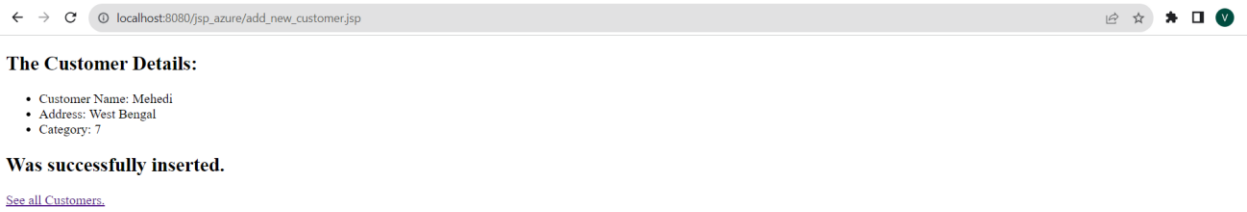


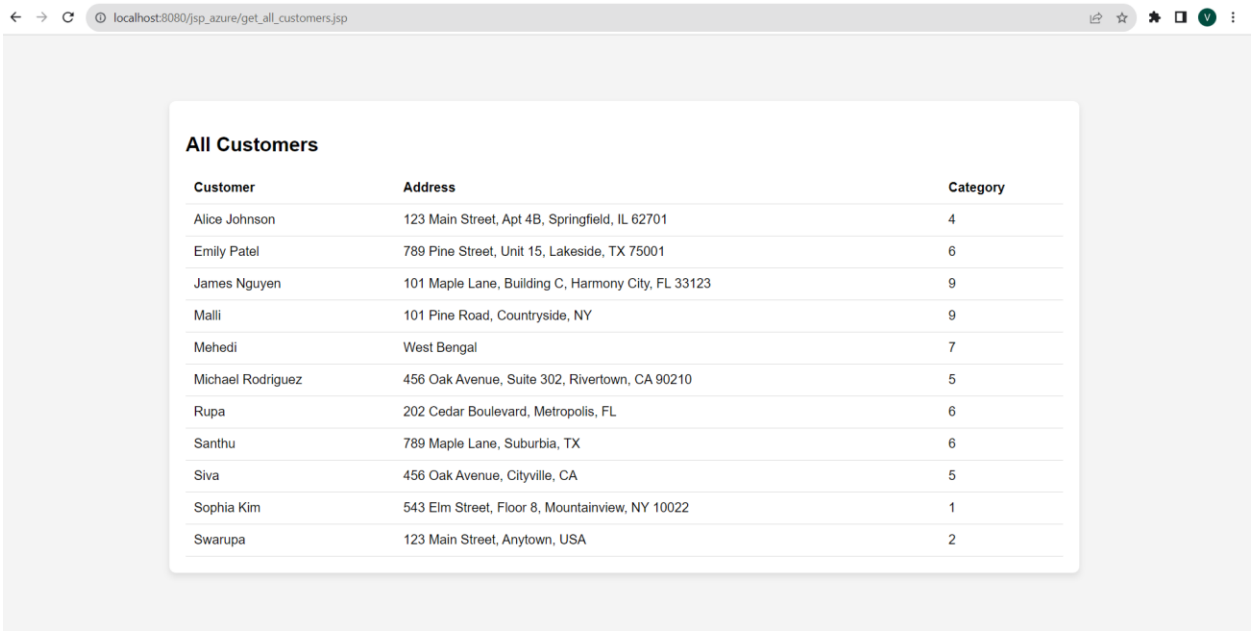


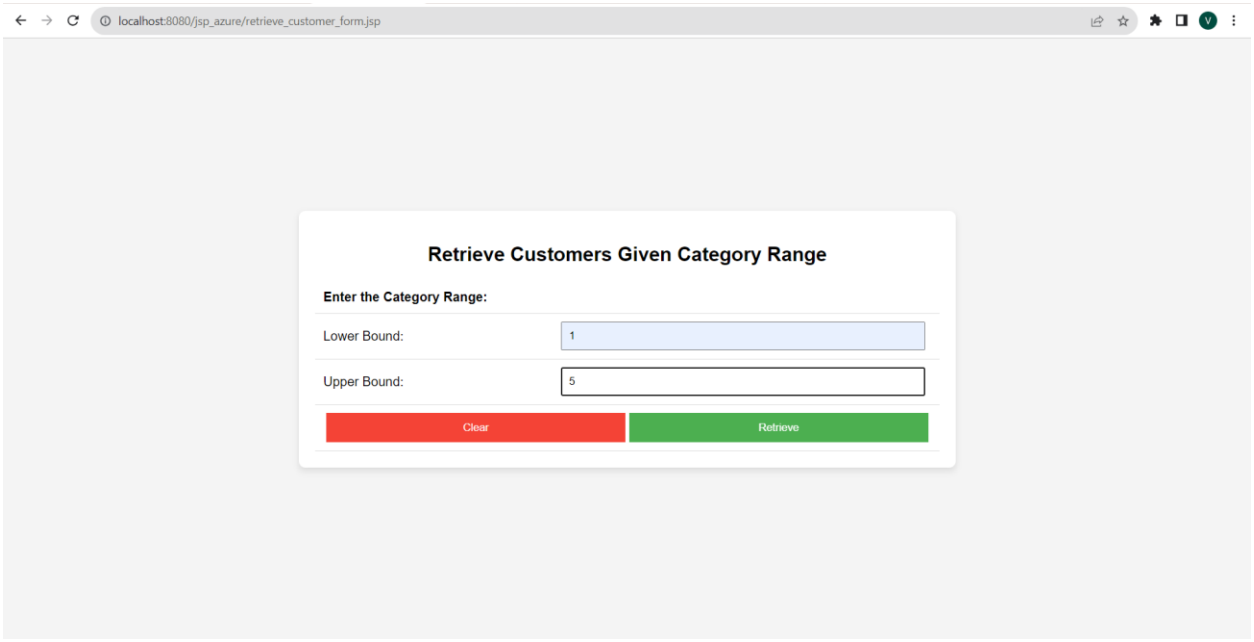


Customer	Category
Alice Johnson	4
Emily Patel	6
James Nguyen	9
Malli	9
Michael Rodriguez	5
Rupa	6
Santhu	6
Siva	5
Sophia Kim	1
Swarupa	2

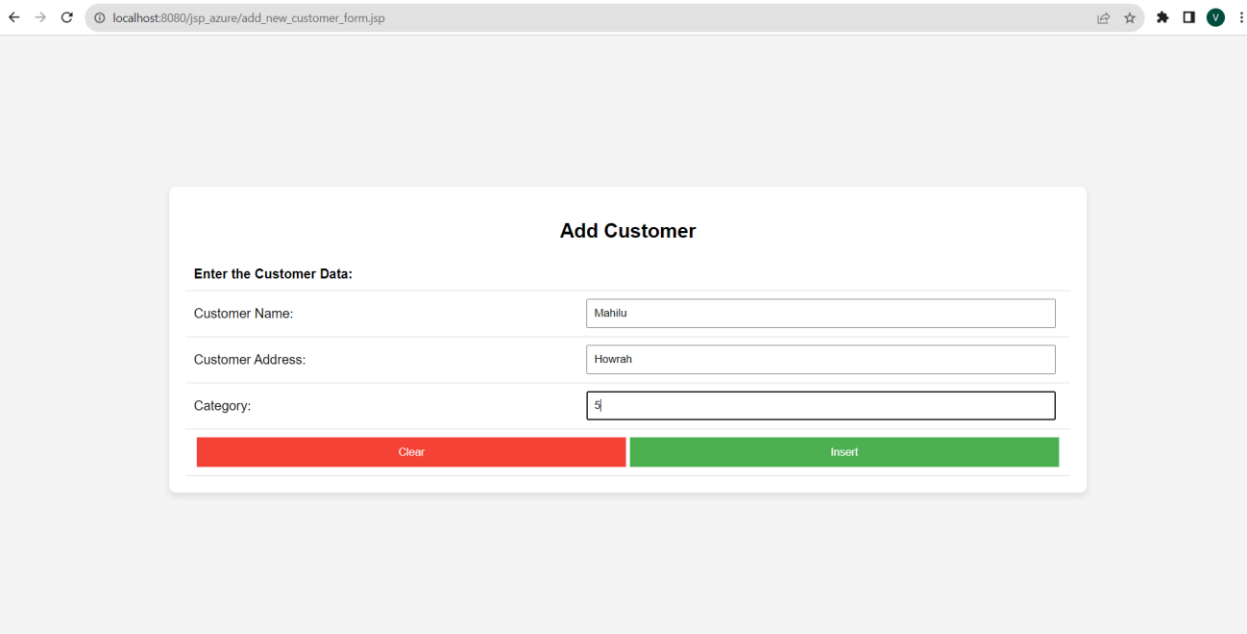

localhost:8080/jsp_azure/add_new_customer_form.jsp
Add Customer
Enter the Customer Data:
Customer Name:
Mehedi
Customer Address:
West Bengal
Category:
7
Clear
Insert


localhost:8080/jsp_azure/add_new_customer.jsp
The Customer Details:
Customer Name: Mehedi
Address: West Bengal
Category: 7
Was successfully inserted.
See all Customers.

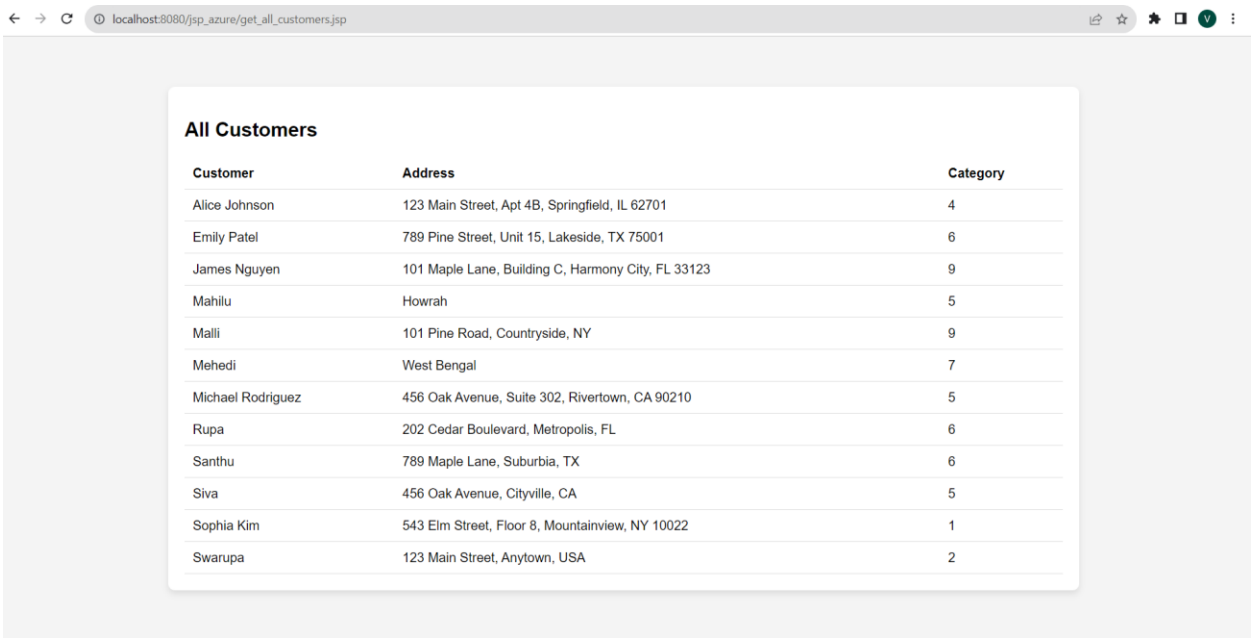

localhost:8080/jsp_azure/get_all_customers.jsp
All Customers
Customer
Address
Category
Alice Johnson
123 Main Street, Apt 4B, Springfield, IL 62701
4
Emily Patel
789 Pine Street, Unit 15, Lakeside, TX 75001
6
James Nguyen
101 Maple Lane, Building C, Harmony City, FL 33123
9
Malli
101 Pine Road, Countryside, NY
9
Mehedi
West Bengal
7
Michael Rodriguez
456 Oak Avenue, Suite 302, Rivertown, CA 90210
5
Rupa
202 Cedar Boulevard, Metropolis, FL
6
Santhu
789 Maple Lane, Suburbia, TX
6
Siva
456 Oak Avenue, Cityville, CA
5
Sophia Kim
543 Elm Street, Floor 8, Mountainview, NY 10022
1
Swarupa
123 Main Street, Anytown, USA
2

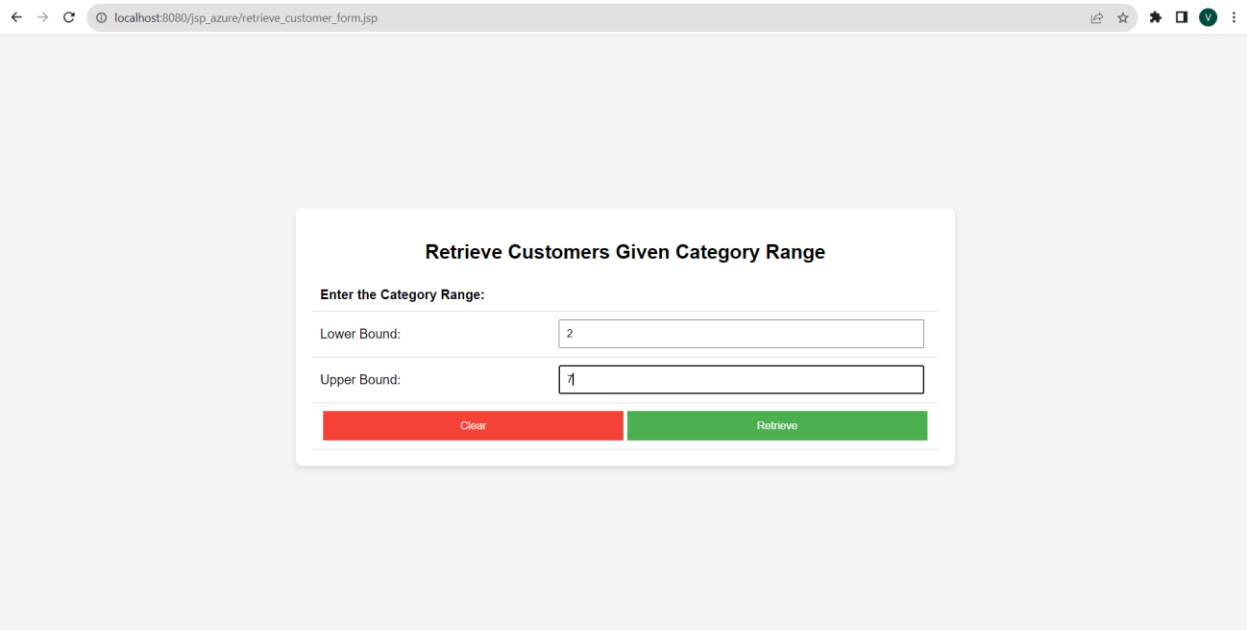

localhost:8080/jsp_azure/retrieve_customer_form.jsp
Retrieve Customers Given Category Range
Enter the Category Range:
Lower Bound:
1
Upper Bound:
5
Clear
Retrieve

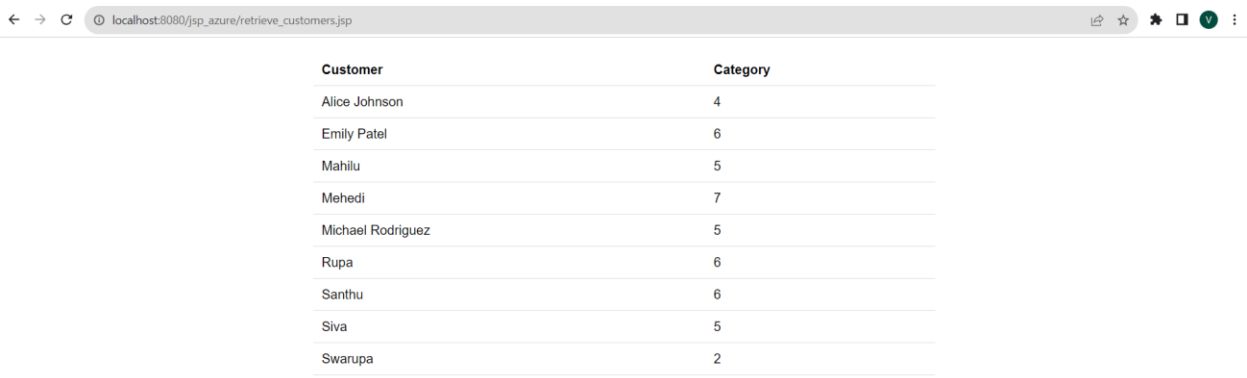
Customer	Category
Alice Johnson	4
Michael Rodriguez	5
Siva	5
Sophia Kim	1
Swarupa	2




localhost:8080/jsp_azure/add_new_customer.jsp
The Customer Details:
Customer Name: Mahilu
Address: Howrah
Category: 5
Was successfully inserted.
See all Customers.


localhost:8080/jsp_azure/get_all_customers.jsp
All Customers
Customer
Address
Category
Alice Johnson
123 Main Street, Apt 4B, Springfield, IL 62701
4
Emily Patel
789 Pine Street, Unit 15, Lakeside, TX 75001
6
James Nguyen
101 Maple Lane, Building C, Harmony City, FL 33123
9
Mahilu
Howrah
5
Malli
101 Pine Road, Countryside, NY
9
Mehedi
West Bengal
7
Michael Rodriguez
456 Oak Avenue, Suite 302, Rivertown, CA 90210
5
Rupa
202 Cedar Boulevard, Metropolis, FL
6
Santhu
789 Maple Lane, Suburbia, TX
6
Siva
456 Oak Avenue, Cityville, CA
5
Sophia Kim
543 Elm Street, Floor 8, Mountainview, NY 10022
1
Swarupa
123 Main Street, Anytown, USA
2





Customer	Category
Alice Johnson	4
Emily Patel	6
Mahilu	5
Mehedi	7
Michael Rodriguez	5
Rupa	6
Santhu	6
Siva	5
Swarupa	2