COURSE : CS/DSA-4513 DATABASE MANAGEMENT

SECTION : 001

SEMESTER: FALL 2023

INSTRUCTOR: DR. LE GRUENWALD

AUTHOR'S NAME: TEDDY DIALLO

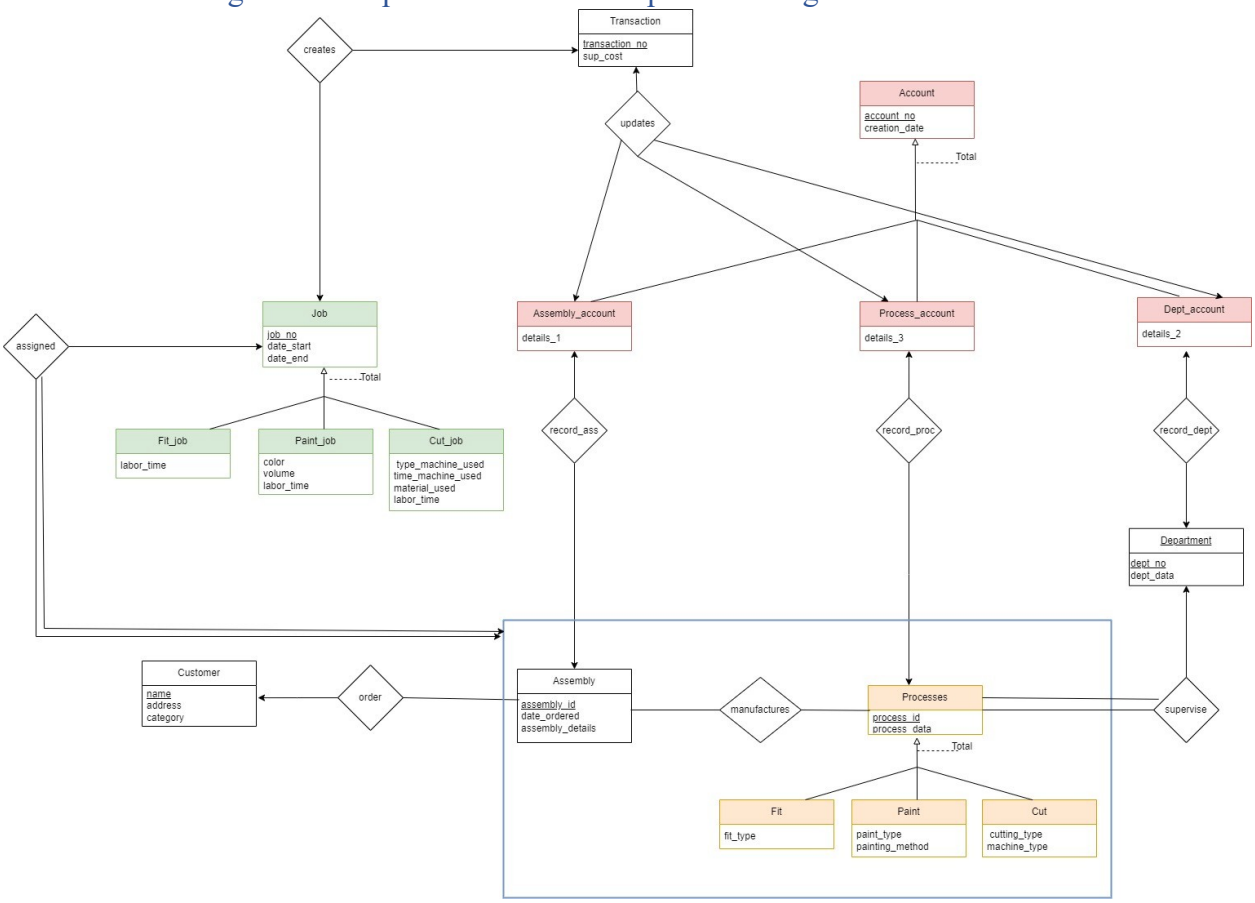
AUTHOR'S Id: 113522777

AUTHOR'S EMAIL: teddy.f.diallo-1@ou.edu

PROJECT'S TITLE: Job-Shop Accounting Database System

Contents

TASK 1 – ER diagram that represents the Job-Shop Accounting database



TASK 2 – Converting the ER diagram to a relational database

Assembly (assembly_id, date_ordered, assembly_details, Customer_name)

Customer (name, address, category)

manufacture (assembly_id, process_id, job_no)

- Fit_process (process_id, process_data, dept_no, fit_type)
- Paint_process (process_id, process_data, dept_no, paint_type, painting_method)
- Cut_process (process_id, process_data, dept_no, cutting_type, machine_type)

Department (dept_no, dept_data)

- Fit_job (job_no, date_start, date_end, labor_time)
- Paint_job (job_no, date_start, date_end, color, volume, labor_time)
- Cut_job (job_no, date_start, date_end, type_machine_used, time_machine_used, material_used, labor_time)

Account_assembly (account_no, creation_date, details_1)

Account_process (account_no, creation_date, details_3)

Account_dept (account_no, creation_date, details_2)

- record_assembly (account_no, assembly_id)
- record_proc (account_no, process_id)
- record_dept (account_no, dept_no)

Transaction (transaction_no, sup_cost)

updates (transaction_no, assembly_account_no, process_account_no, dept_account_no)

creates (job_no , transaction_no)

TASK 3 – File management discussion

3.1 Query analysis and discussion

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	1 – Insertion 12- Range search	NA category	30/day 100/day	B+ tree on category	This table has a lot of insertions, but it has even more range search operations. A B+ tree on category search key is best suited for the efficient range search which is the most frequent operation on that table.
Department	2 – Insertion	NA	Infrequent	Heap file	We are only inserting data therefore a heap file is optimal for quick insertion.
Manufacture	4 – Insertion 6 – Insertion 10 – Range search 11 – Range search	NA NA Date_end (applied to the job tables) Asmbly_id	40/day 50/day 20/day 100/day	B+ tree index on asmbly_id	Regarding that the frequency of range search 120/day and the one for insertion is 90/day (overall), I decided to focus on making the range search queries more efficient using a B+ tree index with the search key assembly_id. That would be slower on insertion but very efficient for the range search
Process_fit	3 – Insertion 11 – Range search	NA Assbly_id (from manufacturtable)	Infrequent 100/day	Heap file	The process is mostly impacted by only the insertion. The range search uses the search key on assembly_id from the assembly table. Therefore, we do not need to worry about a range search

					with the processes.
Process_paint	3 – Insertion 11 – Range search	NA Assbly_id (from manufacturtable)	Infrequent 100/day	Heap file	There is a range search that involves the process tables, but the search key does not involve them therefore, we only need to worry about the insertion that happens. We will use a heap file which is the best file type for efficient insertion. When running query 11, the index made on the manufacture table will handle the efficiency.
Process_cut	3 – Insertion 11 – Range search	NA Assbly_id (from manufacturtable)	Infrequent 100/day	Heap file	There is a range search that involves the process tables, but the search key does not involve them therefore, we only need to worry about the insertion that happens. We will use a heap file which is the best file type for efficient insertion. When running query 11, the index made on the manufacture table will handle the efficiency.
Assembly	4 – Insertion	NA	40/day	Heap file	This query consists of only an insertion on the assembly table therefore a heap file will be sufficient for efficient execution.
Account_assembly	5 – Insertion 8 – Insertion/ Random search 9 – Random Search	NA Acount_no Assembly_id (from record_ass)	10/day 50/day 200/day	Static Hashing on account_no	We have a high frequency random search with query 9 but that applies to the search key from another table. However, query 8 is a random search on account_no. it is low

					frequency but creating a hash function on the table for account_no would make for an efficient random search.
Account_process	5 – Insertion 8 – Update: Random search	NA Acount_no	10/day 50/day	Static Hashing on account_no	A hash file organization would allow an easier implementation of the update function from query 8.
Account_dept	5 – Insertion 8 – Update: Random search	NA Acount_no	10/day 50/day	Static Hashing on account_no	A hash file organization would allow an easier implementation of the update function from query 8.
Transaction	8 – Insertion 9 – Random search	NA Assembly_id (from record_ass table)	50/day 200/day	Heap file	We will use a heap file to allow efficient insertion. We have a range search that involves the transaction table, but the search key of that random search is in another table therefore query9 does not impact our file organization choice for table transaction.
Fit_job	6 – Insertion 7 – Insertion 10 – Range search	NA NA Date_end & Dept_no	50/day 50/day 20/day	Heap file	For job_fit, we have insertions and range search. There are significantly more insertions through the days compared to the range search operation therefore, we will prioritize an efficient insertion rather than an efficient range search and just create a heap file for that table instead of for instance two B+ tree on date_end and dept_no that would have made the range

					search more efficient but the insertions slower.
Cut_Job	6 – Insertion 7 – Insertion 10 – Range search 13 – Range search	NA NA Date_end & Dept_no Job_no	50/day 50/day 20/day 1/month	Heap file	We have two range search on job_cut table however, we have more insertion therefore I will prioritize making those insertions more efficient and use a heap file.
Paint_Job	6 – Insertion 7 – Insertion 10 – Range search 14 – Random Search	Na NA Date_end & Dept_no Job_no	50/day 50/day 20/day 1/week	Heap file	We also have two range search operation happening on this file but the insertions are more frequent therefore I will prioritize making that more efficient by using a heap file.
Record_assembly	5 – Insertion 9 – Random Search	NA Assembly_id	10/day 200/day	Dynamic Hashing	We have a high frequency random search on the search key assembly_id therefore a hashing function on the search key assembly_id from the record_ass table would make that query execute very efficiently.
Record_proc	5 – Insertion	NA	10/day	Heap file	Just insertions therefore we will use heap file for efficient insertion
Record_dept	5 - Insertion	NA	10/day	Heap file	Just insertions therefore we will use heap file for efficient insertion
Updates	8 – Insertion 9 – Random search	NA Assembly_id)	50/day 200/day	Heap file	The update table is involved in query 9 but the search key to execute that random search query affects another table. Therefore, we only care about the

					insertion when choosing the file organization structure therefore, a heap file is the best choice.
Creates	Just insertions	NA	NA	Heap file	We will just do heap file to collect the insertions

3.2 File storage analysis and discussion

In the above table, we have an overview of the different queries, the tables on which they applied and the best file structure to represent such a table. All the file structures that we picked were well translated in the sql code through the creation of B+ indexes mostly. However, for the queries that involved random searches on the tables, I could not implement a hash function which would have been the most optimal for an efficient random search. But azure does not have hashing abilities. For that reason, for the table that would have benefited best from the hashing function, I decided to use an index sequential file organization for a slight improvement on the random searches. The following table shows the updated file organization structure for these tables after I looked at the documentation and failed to implement my initial storage choice.

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Account_assembly	5 – Insertion 8 – Insertion/ Random search 9 – Random Search	NA Acount_no Assembly_id (from record_ass)	10/day 50/day 200/day	Sequential file	To be able to perform a more efficient random search, I will create a sequential file on the search key account number to be able to accommodate query 8. This is good because the random search occurs more than the insertion therefore it is helpful to optimize that query execution.
Account_process	5 – Insertion 8 – Update: Random search	NA Acount_no	10/day 50/day	Sequential file	To be able to perform a more efficient random search, I will create a sequential file on the search key account number to be able to accommodate query 8. This is good

					because the random search occurs more than the insertion therefore it is helpful to optimize that query execution.
Account_dept	5 – Insertion 8 – Update: Random search	NA Acount_no	10/day 50/day	Sequential file	To be able to perform a more efficient random search, I will create a sequential file on the search key account number to be able to accommodate query 8. This is good because the random search occurs more than the insertion therefore it is helpful to optimize that query execution.
Record_assembly	5 – Insertion 9 – Random Search	NA Assembly_id	10/day 200/day	Index-sequential file with secondary index on assembly id	I will create and indexed sequential file on account_no which is the primary key of the table and create a secondary index on assembly_id to optimize the execution of query 9.

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

Creating the drop tables to help with the creation of the database.

```
DROP TABLE IF EXISTS customer;
DROP TABLE IF EXISTS department;
DROP TABLE IF EXISTS manufacture;
DROP TABLE IF EXISTS assembly;
DROP TABLE IF EXISTS job_fit;
DROP TABLE IF EXISTS job_paint;
DROP TABLE IF EXISTS job_cut;
DROP TABLE IF EXISTS process_fit;
DROP TABLE IF EXISTS process_paint;
DROP TABLE IF EXISTS Process_cut;
DROP TABLE IF EXISTS account_assembly;
DROP TABLE IF EXISTS account_process;
DROP TABLE IF EXISTS account_department;
DROP TABLE IF EXISTS record_asssembly;
DROP TABLE IF EXISTS record_process;
DROP TABLE IF EXISTS record_department;
DROP TABLE IF EXISTS transactions;
DROP TABLE IF EXISTS updates;
DROP TABLE IF EXISTS creates;
```

Create the customer table with a B+ tree structure. I also included the constraint so that the customer category can only be either 1,2,3,4,or 5.

```
CREATE TABLE customer (
    name_customer VARCHAR(100)  PRIMARY KEY,
    address_customer VARCHAR(100),
    category INT,
    CONSTRAINT CHK_category_type CHECK (category IN (1,2,3,4,5))
);
CREATE NONCLUSTERED INDEX idx_customer_category ON customer(category);
```

Create the department table as a heap file which is the default when a clustered index for the table is not explicitly defined.

```
CREATE TABLE department (
    dept_no INT PRIMARY KEY,
    dept_data VARCHAR(100),
);
```

Create the manufacture table with a B+ structure.

```
CREATE TABLE manufacture (
    assembly_id INT,
    process_id INT,
    job_no INT,
    PRIMARY KEY (assembly_id, process_id),
);

CREATE NONCLUSTERED INDEX idx_manufacture_assembly_id ON manufacture (assembly_id);
```

Create the assembly table as a heap file.

```
CREATE TABLE assembly (
    assembly_id INT  PRIMARY KEY,
    date_ordered VARCHAR(100),
    assembly_details VARCHAR(100),
    customer_name VARCHAR(100),
);
```

Create the job tables as a heap files.

```
CREATE TABLE job_fit (
    job_no INT PRIMARY KEY,
    date_start VARCHAR(100),
    date_end VARCHAR(100),
    labor_time VARCHAR(100),
);

CREATE TABLE job_paint (
    job_no INT PRIMARY KEY,
    date_start VARCHAR(100),
    date_end VARCHAR(100),
    labor_time VARCHAR(100),
    color VARCHAR(100),
    volume INT,
);

CREATE TABLE job_cut (
    job_no INT PRIMARY KEY,
    date_start VARCHAR(100),
    date_end VARCHAR(100),
    labor_time VARCHAR(100),
    type_machine_used VARCHAR(100),
    time_machine_used VARCHAR(100),
    material_used VARCHAR(100),
);
```

Create the process tables as heap files.

```
CREATE TABLE process_fit(
    process_id INT PRIMARY KEY,
    process_data VARCHAR(100),
    dept_no INT,
    fit_type VARCHAR(100),
);

CREATE TABLE process_paint (
    process_id INT PRIMARY KEY,
    process_data VARCHAR(100),
    dept_no INT,
    paint_type VARCHAR(100),
    painting_method VARCHAR(100),
);

CREATE TABLE process_cut (
    process_id INT PRIMARY KEY,
    process_data VARCHAR(100),
    dept_no INT,
    cutting_type VARCHAR(100),
    machine_type VARCHAR(100),
);
```

Create the account tables as a sequential file.

```
CREATE TABLE account_assembly (
    account_no INT PRIMARY KEY,
    creation_date VARCHAR(100),
    details_1 VARCHAR(100),
);
CREATE CLUSTERED INDEX idx_account_assembly ON account_assembly(account_no);


CREATE TABLE account_process (
    account_no INT PRIMARY KEY,
    creation_date VARCHAR(100),
    details_3 VARCHAR(100),
);
CREATE CLUSTERED INDEX idx_account_process ON account_process(account_no);


 CREATE TABLE account_department (
     account_no INT PRIMARY KEY,
     creation_date VARCHAR(100),
     details_2 VARCHAR(100),
 );
 CREATE CLUSTERED INDEX idx_account_department ON account_department(account_no);
```

Create record_assembly table as an index sequential file.

```
CREATE TABLE record_assembly (
    account_no INT PRIMARY KEY,
    assembly_id INT,
);
CREATE NONCLUSTERED INDEX idx_record_assembly ON record_assembly(assembly_id);
```

Create the record_process and record_department tables as heap files.

```
CREATE TABLE record_process (
    account_no INT PRIMARY KEY,
    process_id INT,
);


CREATE TABLE record_department (
    account_no INT PRIMARY KEY,
    dept_no INT,
);
```

Create the tables: transactions, updates and creates as heap files.

```
CREATE TABLE transactions (
    transaction_no INT PRIMARY KEY,
    sup_cost INT,
);

CREATE TABLE updates (
    transaction_no INT PRIMARY KEY,
    account_no INT,
);

CREATE TABLE creates (
    job_no INT PRIMARY KEY,
    transaction_no INT,
);
```

Task 5 -

5.1 SQL statements (and Transact SQL stored procedures, if any) implementing all queries (1-15 and error checking)

In this section, I will show the execution for an example of the given query and the output that they produce. The queries will be presented in the order they are given.

Query 1

```
CREATE TABLE customer (
    name_customer VARCHAR(100)  PRIMARY KEY,
    address_customer VARCHAR(100),
    category INT,
    CONSTRAINT CHK_category_type CHECK (category IN (1,2,3,4,5))
);
CREATE NONCLUSTERED INDEX idx_customer_category ON customer(category);

EXEC insertCustomer
    @name_customer = 'Teddy Diallo',
    @address_customer = 'Traditions East',
    @category = 5;
Select * From customer
```

Results | Messages

	name_customer	address_customer	category
1	Teddy Diallo	Traditions East	5

Query 2

```
-- 2. Enter a new department (infrequent).
DROP PROCEDURE IF EXISTS insertDepartment;
GO
CREATE PROCEDURE insertDepartment
    @dept_no INT,
    @dept_data VARCHAR(100)
AS
BEGIN
    INSERT INTO department (dept_no, dept_data)
    VALUES (@dept_no, @dept_data);
END

EXEC insertDepartment
    @dept_no = 1,
    @dept_data = 'Research and Development';
Select * From department
```

Results Messages

	dept_no	dept_data
1	1	Research and Development

Query 3

```
-- 3. FIT: Enter a new process-ID and its department information (infrequent).
DROP PROCEDURE IF EXISTS insertFitProcess;
GO
CREATE PROCEDURE insertFitProcess
    @process_id INT,
    @dept_no INT,
    @process_data VARCHAR(100),
    @fit_type VARCHAR(50)
AS
BEGIN
    INSERT INTO process_fit (process_id, dept_no, process_data, fit_type)
    VALUES (@process_id, @dept_no, @process_data, @fit_type);
END

EXEC insertFitProcess
    @process_id = 101,
    @dept_no = 1,
    @process_data = 'Fitting data',
    @fit_type = 'Type A';
SELECT * FROM process_fit
```

Results Messages

	process_id	process_data	dept_no	fit_type
1	101	Fitting data	1	Type A

```sql
-- 3. PAINT: Enter a new process-ID and its department information (infrequent).
DROP PROCEDURE IF EXISTS insertPaintProcess;
GO
CREATE PROCEDURE insertPaintProcess
    @process_id INT,
    @process_data VARCHAR(100),
    @dept_no INT,
    @paint_type VARCHAR(50),
    @painting_method VARCHAR(50)
AS
BEGIN
    INSERT INTO process_paint (process_id, process_data, dept_no, paint_type, painting_method)
    VALUES (@process_id, @process_data, @dept_no, @paint_type, @painting_method);
END

EXEC insertPaintProcess
    @process_id = 201,
    @process_data = 'Painting data',
    @dept_no = 2,
    @paint_type = 'Glossy',
    @painting_method = 'Spray';
SELECT * FROM process_paint
```

Results Messages

	process_id	process_data	dept_no	paint_type	painting_method
1	201	Painting data	2	Glossy	Spray

```sql
-- 3. CUT: Enter a new process-ID and its department information (infrequent).
DROP PROCEDURE IF EXISTS insertCutProcess;
GO
CREATE PROCEDURE insertCutProcess
    @process_id INT,
    @process_data VARCHAR(100),
    @dept_no INT,
    @cutting_type VARCHAR(50),
    @machine_type VARCHAR(50)
AS
BEGIN
    INSERT INTO process_cut (process_id, process_data, dept_no, cutting_type, machine_type)
    VALUES (@process_id, @process_data, @dept_no, @cutting_type, @machine_type);
END

EXEC insertCutProcess
    @process_id = 301,
    @process_data = 'Cutting data',
    @dept_no = 3,
    @cutting_type = 'Laser',
    @machine_type = 'Laser Cutter Model X';
SELECT * FROM process_cut
```

Results Messages

	process_id	process_data	dept_no	cutting_type	machine_type
1	301	Cutting data	3	Laser	Laser Cutter Model X

Query 4

```
-- 4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and date ordered and associate it with one or more processes
DROP PROCEDURE IF EXISTS insertAssembly;
GO
CREATE PROCEDURE insertAssembly
    @assembly_id INT,
    @date_ordered DATE,
    @assembly_details VARCHAR(100),
    @customer_name VARCHAR(100),
    @process_ids VARCHAR(MAX)  -- Comma-separated string of process IDs
AS
BEGIN
    -- Insert data into Assembly table
    INSERT INTO Assembly (assembly_id, date_ordered, assembly_details, Customer_name)
    VALUES (@assembly_id, @date_ordered, @assembly_details, @customer_name);

    -- Split the comma-separated process_ids string into a table
    DECLARE @process_id_table TABLE (process_id INT);
    INSERT INTO @process_id_table (process_id)
    SELECT value
    FROM STRING_SPLIT(@process_ids, ',');

    -- Associate with processes
    INSERT INTO manufacture (assembly_id, process_id)
    SELECT @assembly_id, process_id
    FROM @process_id_table;
END
GO

-- Example: Insert a new assembly associated existing with process IDs
EXEC insertAssembly
    @assembly_id = 1,
    @date_ordered = '2023-11-08',
    @assembly_details = 'Test Assembly',
    @customer_name = 'Customer A',
    @process_ids = '101,201,301';

-- Verify the assembly insertion
SELECT * FROM Assembly;

-- Verify the associations in the manufacture table
SELECT * FROM manufacture;
```

Results Messages

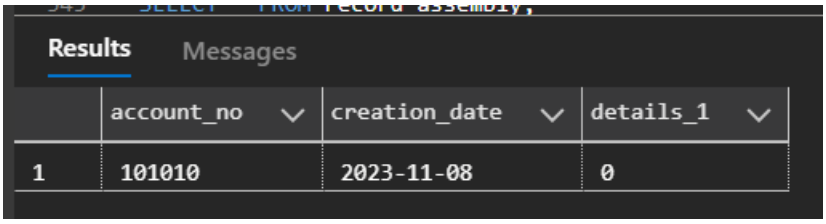
	assembly_id	date_ordered	assembly_details	customer_name
1	1	2023-11-08	Test Assembly	Customer A

	assembly_id	process_id	job_no
1	1	101	NULL
2	1	201	NULL
3	1	301	NULL

Query 5

```
-- 5. ASSEMBLY ACCOUNT: Create a new account and associate it with the process, assembly, or department to which it is applicable.
-- Create a new assembly account and associate it with an assembly
DROP PROCEDURE IF EXISTS createAssemblyAccount;
GO
CREATE PROCEDURE createAssemblyAccount
    @account_no INT,
    @creation_date DATE,
    @details_1 INT,
    @assembly_id INT
AS
BEGIN
    -- Insert data into account_assembly
    INSERT INTO account_assembly (account_no, creation_date, details_1)
    VALUES (@account_no, @creation_date, @details_1);

    -- Insert into record_assembly to associate account with assembly
    INSERT INTO record_assembly (account_no, assembly_id)
    VALUES (@account_no, @assembly_id);
END
GO

-- Example: Create an assembly account and associate it with assembly ID 1
EXEC createAssemblyAccount
    @account_no = 101010,
    @creation_date = '2023-11-08',
    @details_1 = 0,
    @assembly_id = 1;

-- Verify the assembly account insertion
SELECT * FROM account_assembly;

-- Verify the association in the record_assembly table
SELECT * FROM record_assembly;
```

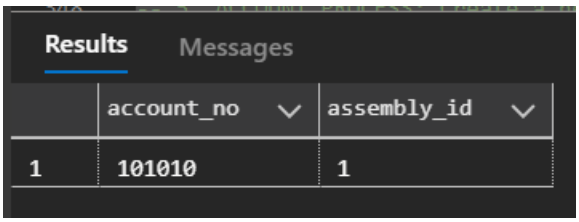
Account_assembly output



Results | Messages

	account_no	creation_date	details_1
1	101010	2023-11-08	0

Record_assembly output

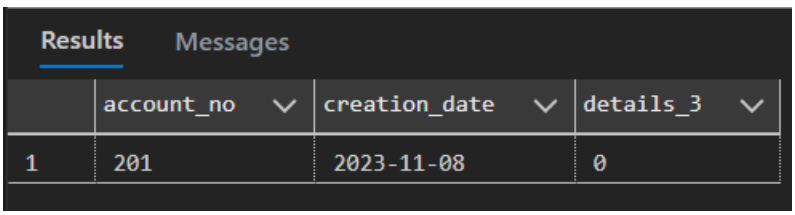


Results | Messages

	account_no	assembly_id
1	101010	1

```sql
-- 5. ACCOUNT PROCESS: Create a new account and associate it with the process, assembly, or department to which it is applicable.
DROP PROCEDURE IF EXISTS createProcessAccount;
GO
CREATE PROCEDURE createProcessAccount
    @account_no INT,
    @creation_date DATE,
    @details_3 VARCHAR(100),
    @process_id INT
AS
BEGIN
    -- Insert data into account_process
    INSERT INTO account_process (account_no, creation_date, details_3)
    VALUES (@account_no, @creation_date, @details_3);

    -- Insert into record_proc to associate account with process
    INSERT INTO record_process (account_no, process_id)
    VALUES (@account_no, @process_id);
END
GO

-- Example: Create a process account and associate it with process ID 101
EXEC createProcessAccount
    @account_no = 201,
    @creation_date = '2023-11-08',
    @details_3 = 0,
    @process_id = 101;

-- Verify the process account insertion
SELECT * FROM account_process;

-- Verify the association in the record_process table
SELECT * FROM record_process;
```

Account_process output



Results | Messages

	account_no	creation_date	details_3
1	201	2023-11-08	0

Record_process output

Results Messages

	account_no	process_id
1	201	101

```sql
-- 5. ACCOUNT DEPT: Create a new account and associate it with the process, assembly, or department to which it is applicable (10/day).
DROP PROCEDURE IF EXISTS createDepartmentAccount;
GO
CREATE PROCEDURE createDepartmentAccount
    @account_no INT,
    @creation_date DATE,
    @details_2 VARCHAR(100),
    @dept_no INT
AS
BEGIN
    -- Insert data into account_dept
    INSERT INTO account_department (account_no, creation_date, details_2)
    VALUES (@account_no, @creation_date, @details_2);

    -- Insert into record_dept to associate account with department
    INSERT INTO record_department (account_no, dept_no)
    VALUES (@account_no, @dept_no);
END
GO
-- Example: Create a department account and associate it with department number 1
EXEC createDepartmentAccount
    @account_no = 301,
    @creation_date = '2023-11-08',
    @details_2 = 0,
    @dept_no = 1;

-- Verify the department account insertion
SELECT * FROM account_department;

-- Verify the association in the record_department table
SELECT * FROM record_department;
```

Account_department

Results Messages

	account_no	creation_date	details_2
1	301	2023-11-08	0

Record department output

Results Messages

	account_no	dept_no
1	301	1

Query 6

Fit Job

```
-- 6. FIT: Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day)
DROP PROCEDURE IF EXISTS insertJobFit;
GO
CREATE PROCEDURE insertJobFit
    @job_no INT,
    @assembly_id INT,
    @process_id INT,
    @date_start VARCHAR(100)
AS
BEGIN
    -- Insert data into job_fit
    INSERT INTO job_fit (job_no, date_start)
    VALUES (@job_no, @date_start);

    -- Insert or update the job number in the manufacture table
    MERGE manufacture AS target
    USING (SELECT @assembly_id AS assembly_id, @process_id AS process_id) AS source
    ON target.assembly_id = source.assembly_id AND target.process_id = source.process_id
    WHEN MATCHED THEN
        UPDATE SET target.job_no = @job_no
    WHEN NOT MATCHED THEN
        INSERT (assembly_id, process_id, job_no)
        VALUES (@assembly_id, @process_id, @job_no);
END

-- Example for insertJobFit:
EXEC insertJobFit
    @job_no = 3,
    @assembly_id = 1,
    @process_id = 101,
    @date_start = '10 Dec'
SELECT * FROM job_fit
SELECT * From manufacture
```

Results Messages

	job_no	date_start	date_end	labor_time
1	3	10 Dec	NULL	NULL

Results Messages

	assembly_id	process_id	job_no
1	1	101	3

Paint Job

```sql
-- 6. PAINT: Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day).
DROP PROCEDURE IF EXISTS insertJobPaint;
GO
CREATE PROCEDURE insertJobPaint
    @job_no INT,
    @assembly_id INT,
    @process_id INT,
    @date_start VARCHAR(100)
AS
BEGIN
    -- Insert data into job_paint
    INSERT INTO job_paint (job_no, date_start)
    VALUES (@job_no, @date_start);

    -- Insert or update the job number in the manufacture table
    MERGE manufacture AS target
    USING (SELECT @assembly_id AS assembly_id, @process_id AS process_id) AS source
    ON target.assembly_id = source.assembly_id AND target.process_id = source.process_id
    WHEN MATCHED THEN
        UPDATE SET target.job_no = @job_no
    WHEN NOT MATCHED THEN
        INSERT (assembly_id, process_id, job_no)
        VALUES (@assembly_id, @process_id, @job_no);
END

-- Example for insertJobPaint:
EXEC insertJobPaint
    @job_no = 2,
    @assembly_id = 2,
    @process_id = 201,
    @date_start = '1 May'
SELECT * FROM job_paint
SELECT * FROM manufacture
```

Results Messages

	job_no	date_start	date_end	labor_time	color	volume
1	2	1 May	NULL	NULL	NULL	NULL

Results Messages

	assembly_id	process_id	job_no
1	1	101	3
2	2	201	2

Cut Job

```sql
-- 6. CUT: Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced (50/day).
DROP PROCEDURE IF EXISTS insertJobCut;
GO
CREATE PROCEDURE insertJobCut
    @job_no INT,
    @assembly_id INT,
    @process_id INT,
    @date_start DATE
AS
BEGIN
    -- Insert data into job_cut
    INSERT INTO job_cut (job_no, date_start)
    VALUES (@job_no, @date_start);

    -- Insert or update the job number in the manufacture table
    MERGE manufacture AS target
    USING (SELECT @assembly_id AS assembly_id, @process_id AS process_id) AS source
    ON target.assembly_id = source.assembly_id AND target.process_id = source.process_id
    WHEN MATCHED THEN
        UPDATE SET target.job_no = @job_no
    WHEN NOT MATCHED THEN
        INSERT (assembly_id, process_id, job_no)
        VALUES (@assembly_id, @process_id, @job_no);
END

-- Example for insertJobCut:
EXEC insertJobCut
    @job_no = 3,
    @assembly_id = 1,
    @process_id = 301,
    @date_start = '2023-11-08'
SELECT * FROM job_cut
SELECT * FROM manufacture
```

Results Messages

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	3	2023-11-08	NULL	NULL	NULL	NULL	NULL

Results Messages

	assembly_id	process_id	job_no
1	1	101	3
2	1	301	3
3	2	201	2

Query 7 – Completing with the end date and the missing details

Job fit

```
-- 7. FIT: At the completion of a job, enter the date its completed and the information relevant to the type (50/day)
-- Update a Fit_job upon completion
DROP PROCEDURE IF EXISTS completeFitJob;
GO
CREATE PROCEDURE completeFitJob
    @job_no INT,
    @date_end DATE,
    @labor_time DECIMAL(10,2)
AS
BEGIN
    UPDATE job_fit
    SET date_end = @date_end, labor_time = @labor_time
    WHERE job_no = @job_no;
END;
GO

-- Example for completeFitJob
EXEC completeFitJob
    @job_no = 3,
    @date_end = '2023-11-10',
    @labor_time = 12.5;
SELECT * FROM job_fit
```

Results | Messages

	job_no	date_start	date_end	labor_time
1	3	10 Dec	2023-11-10	12.50

Job paint

```sql
-- 7. PAINT: At the completion of a job, enter the date its completed and the information relevant to the type (50/day)
-- Update a Paint_job upon completion
DROP PROCEDURE IF EXISTS completePaintJob;
GO
CREATE PROCEDURE completePaintJob
    @job_no INT,
    @date_end DATE,
    @color VARCHAR(100),
    @volume INT,
    @labor_time DECIMAL(10,2)
AS
BEGIN
    UPDATE job_paint
    SET date_end = @date_end, color = @color, volume = @volume, labor_time = @labor_time
    WHERE job_no = @job_no;
END;
GO

-- Example for completePaintJob
EXEC completePaintJob
    @job_no = 2,
    @date_end = '2023-11-15',
    @color = 'Blue',
    @volume = 100,
    @labor_time = 10.5;
SELECT * FROM job_paint
```

Results Messages

	job_no	date_start	date_end	labor_time	color	volume
1	2	1 May	2023-11-15	10.50	Blue	100

Job cut

```sql
-- 7. CUT: At the completion of a job, enter the date its completed and the information relevant to the type (50/day)
-- Update a Cut_job upon completion
DROP PROCEDURE IF EXISTS completeCutJob;
GO
CREATE PROCEDURE completeCutJob
    @job_no INT,
    @date_end DATE,
    @type_machine_used VARCHAR(100),
    @time_machine_used DECIMAL(10,2),
    @material_used VARCHAR(100),
    @labor_time DECIMAL(10,2)
AS
BEGIN
    UPDATE job_cut
    SET date_end = @date_end, type_machine_used = @type_machine_used,
        time_machine_used = @time_machine_used, material_used = @material_used,
        labor_time = @labor_time
    WHERE job_no = @job_no;
END;
GO
-- Example for completeCutJob
EXEC completeCutJob
    @job_no = 3,
    @date_end = '2023-11-20',
    @type_machine_used = 'Saw',
    @time_machine_used = 5.0,
    @material_used = 'Wood',
    @labor_time = 15.0;
SELECT * FROM job_cut
```

Results Messages

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	3	2023-11-08	2023-11-20	15.00	Saw	5.00	Wood

Query 8

```sql
-- 8. Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup
DROP PROCEDURE IF EXISTS enterTransactionAndUpdateAccounts;
GO
CREATE PROCEDURE enterTransactionAndUpdateAccounts
    @transaction_no INT,
    @sup_cost DECIMAL(10, 2),
    @assembly_account_no INT = NULL,  -- Optional parameters, NULL if not applicable
    @process_account_no INT = NULL,
    @dept_account_no INT = NULL
AS
BEGIN
    -- Insert data into transactions table
    INSERT INTO transactions (transaction_no, sup_cost)
    VALUES (@transaction_no, @sup_cost);

    -- Check if the assembly account number is provided
    IF @assembly_account_no IS NOT NULL
    BEGIN
        -- Update the details_1 in account_assembly
        UPDATE account_assembly
        SET details_1 = details_1 + @sup_cost
        WHERE account_no = @assembly_account_no;

        --Also fill up the update account (comes in handy for query 9)
        INSERT INTO updates(transaction_no, account_no_assembly)
        VALUES (@transaction_no, @assembly_account_no)
    END

    -- Check if the process account number is provided
    IF @process_account_no IS NOT NULL
    BEGIN
        -- Update the details_3 in account_process
        UPDATE account_process
        SET details_3 = details_3 + @sup_cost
        WHERE account_no = @process_account_no;
    END

    -- Check if the department account number is provided
    IF @dept_account_no IS NOT NULL
    BEGIN
        -- Update the details_2 in account_dept
        UPDATE account_department
        SET details_2 = details_2 + @sup_cost
        WHERE account_no = @dept_account_no;
    END
END
```

```sql
-- Example for enterTransactionAndUpdateAccounts
EXEC enterTransactionAndUpdateAccounts
    @transaction_no = 1,
    @sup_cost = 100.00,
    @assembly_account_no = 101010,
    @process_account_no = NULL,
    @dept_account_no = NULL;

EXEC enterTransactionAndUpdateAccounts
    @transaction_no = 2,
    @sup_cost = 100.00,
    @assembly_account_no = NULL,
    @process_account_no = 201,
    @dept_account_no = NULL;

EXEC enterTransactionAndUpdateAccounts
    @transaction_no = 3,
    @sup_cost = 100.00,
    @assembly_account_no = NULL,
    @process_account_no = NULL,
    @dept_account_no = 301;

SELECT * FROM transactions
SELECT * FROM account_assembly
SELECT * FROM account_process
SELECT * FROM account_department
SELECT * FROM updates
```

Transaction table

Results Messages

	transaction_no	sup_cost
1	1	100
2	2	100
3	3	100

Assembly_account updated as a result of query 8.

Results Messages

	account_no	creation_date	details_1
1	101010	2023-11-08	100

Process _account updated because of query 8.

	account_no	creation_date	details_3
1	201	2023-11-08	100

Department_account updated because of query 8.

	account_no	creation_date	details_2
1	301	2023-11-08	100

Query 9

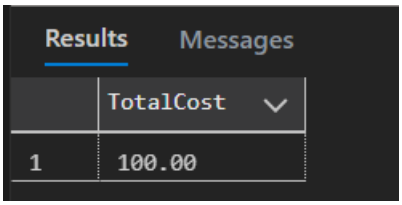
```
-- 9. Retrieve the total cost incurred on an assembly-id (200/day).
DROP PROCEDURE IF EXISTS calculateTotalCostForAssembly;
GO
CREATE PROCEDURE calculateTotalCostForAssembly
    @assembly_id INT
AS
BEGIN
    DECLARE @totalCost DECIMAL(10, 2);

    SELECT @totalCost = SUM(t.sup_cost)
    FROM transactions t
    INNER JOIN updates u ON t.transaction_no = u.transaction_no
    INNER JOIN record_assembly ra ON u.account_no_assembly = ra.account_no
    WHERE ra.assembly_id = @assembly_id;

    -- Return the total cost
    SELECT @totalCost AS TotalCost;
END
GO
-- Declare a variable to hold the result
DECLARE @result DECIMAL(10, 2);
-- Execute the procedure with the desired @assembly_id
EXEC calculateTotalCostForAssembly @assembly_id = 1; -- Replace with a valid assembly_id
-- Retrieve the result
SELECT @result AS TotalCost;
```

Content of update account after running the query 8 with an assembly.

Results Messages

	transaction_no	account_no_assembly	account_no_process	account_no_department
1	1	101010	NULL	NULL

The total cost of the assembly being recorded (comes from the record_assembly table connect through the update table).



Query 10

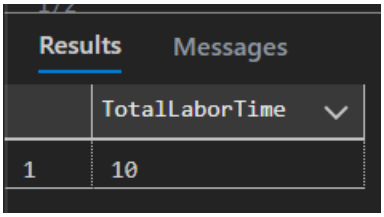
The stored procedure for query 10

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

DROP PROCEDURE IF EXISTS TotalLaborTimeInDepartment_WithIndex;
GO

CREATE PROCEDURE TotalLaborTimeInDepartment_WithIndex
    @dept_no INT,
    @given_date VARCHAR(100),
    @total_labor_time INT OUTPUT
AS
BEGIN
    SELECT @total_labor_time = SUM(labor_time)
    FROM (
        SELECT labor_time FROM job_fit
        WHERE job_no IN (SELECT job_no FROM manufacture WHERE process_id IN (SELECT process_id FROM process_fit WHERE dept_no = @dept_no))
          AND date_end = @given_date
        UNION ALL
        SELECT labor_time FROM job_cut
        WHERE job_no IN (SELECT job_no FROM manufacture WHERE process_id IN (SELECT process_id FROM process_cut WHERE dept_no = @dept_no))
          AND date_end = @given_date
        UNION ALL
        SELECT labor_time FROM job_paint
        WHERE job_no IN (SELECT job_no FROM manufacture WHERE process_id IN (SELECT process_id FROM process_paint WHERE dept_no = @dept_no))
          AND date_end = @given_date
    ) AS Jobs;
END;
```

Note that to test query 10, I manually populated the tables involved in the stored procedure to test that the stored procedure worked correctly. That is because that stored procedure was more involved, and I could have an easier time testing by declaring new variables in the tables myself to be able to test. Here are the values I inserted in the tables involved in the stored procedure.

```sql
-- Insert data for department
INSERT INTO department (dept_no, dept_data) VALUES (1, 'Assembly');

-- Insert process data
INSERT INTO process_fit (process_id, process_data, dept_no, fit_type) VALUES (101, 'Process Data', 1, 'Type 1');
INSERT INTO process_paint (process_id, process_data, dept_no, paint_type, painting_method) VALUES (201, 'Process Data', 1, 'Type A', 'Method A');
INSERT INTO process_cut (process_id, process_data, dept_no, cutting_type, machine_type) VALUES (301, 'Process Data', 1, 'Laser Cut', 'Laser Cutter');

-- Insert manufacture data to associate processes with assemblies
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 101, 1);
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 201, 2);
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 301, 3);

-- Insert job data
INSERT INTO job_fit (job_no, date_start, date_end, labor_time) VALUES (1, '2023-11-08', '2023-11-10', 5);
INSERT INTO job_paint (job_no, date_start, date_end, labor_time) VALUES (2, '2023-11-08', '2023-11-10', 3);
INSERT INTO job_cut (job_no, date_start, date_end, labor_time) VALUES (3, '2023-11-08', '2023-11-10', 2);

-- Now, execute the stored procedure for the department number 1 and the date '2023-11-10'
DECLARE @TotalLabor INT;
EXEC TotalLaborTimeInDepartment_WithIndex
    @dept_no = 1, -- Department number
    @given_date = '2023-11-10', -- Date of interest
    @total_labor_time = @TotalLabor OUTPUT;

-- Retrieve the result
SELECT @TotalLabor AS TotalLaborTime;
```

And here is the total time based on that input which shows correct logic.



Query 11

```sql
-- 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).
DROP PROCEDURE IF EXISTS ProcessesForAssembly_WithIndex;
GO

CREATE PROCEDURE ProcessesForAssembly_WithIndex
    @givenAssemblyId INT
AS
BEGIN
    SELECT
        m.assembly_id,
        m.process_id,
        COALESCE(pf.process_data, pp.process_data, pc.process_data) AS process_data,
        COALESCE(pf.dept_no, pp.dept_no, pc.dept_no) AS dept_no,
        d.dept_data AS department_responsible
    FROM
        manufacture m
        LEFT JOIN process_fit pf ON m.process_id = pf.process_id
        LEFT JOIN process_paint pp ON m.process_id = pp.process_id
        LEFT JOIN process_cut pc ON m.process_id = pc.process_id
        LEFT JOIN department d ON d.dept_no = COALESCE(pf.dept_no, pp.dept_no, pc.dept_no)
    WHERE
        m.assembly_id = @givenAssemblyId
    ORDER BY
        COALESCE(pf.process_data, pp.process_data, pc.process_data); -- Assuming process_data has date info
END;
GO
```

I also manually populated the tables involved in query 11 to have a better control of the testing.

```sql
-- Insert into department
INSERT INTO department (dept_no, dept_data) VALUES (1, 'Assembly');
INSERT INTO department (dept_no, dept_data) VALUES (2, 'Painting');
INSERT INTO department (dept_no, dept_data) VALUES (3, 'Cutting');

-- Insert into jobs
INSERT INTO job_fit (job_no, date_start, date_end, labor_time) VALUES (1, '2023-01-01', '2023-01-02', 8);
INSERT INTO job_paint (job_no, date_start, date_end, labor_time, color, volume) VALUES (2, '2023-01-03', '2023-01-04', 10, 'Red', 5);
INSERT INTO job_cut (job_no, date_start, date_end, labor_time, type_machine_used, time_machine_used, material_used)
VALUES (3, '2023-01-05', '2023-01-06', 6, 'Laser Cutter', '1.5', 'Steel');

-- Insert into manufacture
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 101, 1);
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 201, 2);
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES (1, 301, 3);

-- Insert into processes
INSERT INTO process_fit (process_id, process_data, dept_no, fit_type) VALUES (101, 'Fitting Data', 1, 'Type A');
INSERT INTO process_paint (process_id, process_data, dept_no, paint_type, painting_method) VALUES (201, 'Painting Data', 2, 'Glossy', 'Spray');
INSERT INTO process_cut (process_id, process_data, dept_no, cutting_type, machine_type) VALUES (301, 'Cutting Data', 3, 'Laser', 'Model X');

-- Execute the procedure for assembly_id = 1
EXEC ProcessesForAssembly_WithIndex @givenAssemblyId = 1;
```

Results | Messages

	assembly_id	process_id	process_data	dept_no	department_responsible
1	1	301	Cutting Data	3	Cutting
2	1	101	Fitting Data	1	Assembly
3	1	201	Painting Data	2	Painting

Query 12

Note that the type of category in customer is an int to facilitate operations.

```sql
-- 12. Retrieve the customers (in name order) whose category is in a given range (100/day).
DROP PROCEDURE IF EXISTS retrieveCustomerByRange;
GO
CREATE PROCEDURE retrieveCustomerByRange
    @category_start VARCHAR(100),
    @category_end VARCHAR(100)
AS
BEGIN
    SELECT name_customer, address_customer, category
    FROM customer
    WHERE category BETWEEN @category_start AND @category_end
    ORDER BY name_customer;
END

-- Insert sample data into the 'customer' table
INSERT INTO customer (name_customer, address_customer, category)
VALUES
    ('Ana', 'Address 2', 5),
    ('Kate', 'Address 3', 1),
    ('Laury', 'Address 4', 3),
    ('Lili', 'Address 5', 4);
-- Call the stored procedure to retrieve customers in a given category range
EXEC retrieveCustomerByRange @category_start = 3, @category_end = 5;
```

Results Messages

	name_customer	address_customer	category
1	Ana	Address 2	5
2	Laury	Address 4	3
3	Lili	Address 5	4

Query 13

```sql
-- 13. Delete all cut-jobs whose job-no is in a given range (1/month).
DROP PROCEDURE IF EXISTS deleteCutJobsInRange;
GO
CREATE PROCEDURE deleteCutJobsInRange
    @start_job_no INT,
    @end_job_no INT
AS
BEGIN
    DELETE FROM job_cut
    WHERE job_no BETWEEN @start_job_no AND @end_job_no;
END;
GO

-- Insert sample data into the 'job_cut' table
INSERT INTO job_cut (job_no, date_start, date_end, labor_time, type_machine_used, time_machine_used, material_used)
VALUES
(1, '2023-01-01', '2023-01-02', 5, 'Cutter A', '2 hours', 'Steel'),
(2, '2023-01-05', '2023-01-06', 4, 'Cutter B', '3 hours', 'Aluminum'),
(3, '2023-01-10', '2023-01-11', 6, 'Cutter C', '1.5 hours', 'Copper'),
(4, '2023-01-15', '2023-01-16', 8, 'Cutter D', '2.5 hours', 'Titanium');
-- Check the data before deletion
SELECT * FROM job_cut;
-- Call the stored procedure to delete jobs with job_no between 2 and 3
EXEC deleteCutJobsInRange @start_job_no = 2, @end_job_no = 3;
-- Check the data after deletion
SELECT * FROM job_cut;
```

Results Messages

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	1	2023-01-01	2023-01-02	5	Cutter A	2 hours	Steel
2	2	2023-01-05	2023-01-06	4	Cutter B	3 hours	Aluminum
3	3	2023-01-10	2023-01-11	6	Cutter C	1.5 hours	Copper
4	4	2023-01-15	2023-01-16	8	Cutter D	2.5 hours	Titanium

Results Messages

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	1	2023-01-01	2023-01-02	5	Cutter A	2 hours	Steel
2	4	2023-01-15	2023-01-16	8	Cutter D	2.5 hours	Titanium

Query 14

```sql
-- 14. Change the color of a given paint job (1/week).
DROP PROCEDURE IF EXISTS updatePaintJobColor;
GO
CREATE PROCEDURE updatePaintJobColor
    @job_no INT,
    @new_color VARCHAR(100)
AS
BEGIN
    UPDATE job_paint
    SET color = @new_color
    WHERE job_no = @job_no;
END;
GO

-- Insert sample data into the 'job_paint' table
INSERT INTO job_paint (job_no, date_start, date_end, labor_time, color, volume)
VALUES
(1, '2023-01-01', '2023-01-02', 8, 'Red', 10),
(2, '2023-01-05', '2023-01-06', 7, 'Blue', 20),
(3, '2023-01-10', '2023-01-11', 9, 'Green', 15);
-- Check the data before the update
SELECT * FROM job_paint;
-- Call the stored procedure to change the color of job_no 2 to 'Yellow'
EXEC updatePaintJobColor @job_no = 2, @new_color = 'Yellow';
-- Check the data after the update
SELECT * FROM job_paint;
```

Results Messages

	job_no	date_start	date_end	labor_time	color	volume
1	1	2023-01-01	2023-01-02	8	Red	10
2	2	2023-01-05	2023-01-06	7	Blue	20
3	3	2023-01-10	2023-01-11	9	Green	15

Results Messages

	job_no	date_start	date_end	labor_time	color	volume
1	1	2023-01-01	2023-01-02	8	Red	10
2	2	2023-01-05	2023-01-06	7	Yellow	20
3	3	2023-01-10	2023-01-11	9	Green	15

5.2 The Java source program and screenshots showing its successful compilation.

Connexion and menu code creation

```
import java.io.FileReader;
public class JobShopAccounting {

    public static void main(String[] args) {
        // JDBC connection setup
        String url = "jdbc:sqlserver://dial0007.database.windows.net:1433;database=cs-dsa-4513-sql-db;user=dial0007@dial0007;password=Candidatkey99;encrypt=true;
        String user = "dial0007";
        String password = "Candidatkey99";

        try (Connection connection = DriverManager.getConnection(url, user, password)) {
            Scanner scanner = new Scanner(System.in);

            while (true) {
                // Display menu options
                System.out.println("Options:");
                System.out.println("1. Enter a new customer");
                System.out.println("2. Enter a new department");
                System.out.println("3. Enter a new process-id and its department together with its type and information relevant to the type");
                System.out.println("4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-ordered and associate it with one or
                System.out.println("5. Create a new account and associate it with the process, assembly, or department to which it is applicable");
                System.out.println("6. Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced");
                System.out.println("7. At the completion of a job, enter the date it completed and the information relevant to the type of job");
                System.out.println("8. Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to
                System.out.println("9. Retrieve the total cost incurred on an assembly-id");
                System.out.println("10. Retrieve the total labor time within a department for jobs completed in the department during a given date");
                System.out.println("11. Retrieve the processes through which a given assembly-id has passed so far (in date-commenced order) and the department r
                System.out.println("12. Retrieve the customers (in name order) whose category is in a given range");
                System.out.println("13. Delete all cut-jobs whose job-no is in a given range");
                System.out.println("14. Change the color of a given paint job");
                System.out.println("15. Import option: enter new customers from a data file");
                System.out.println("16. Export option: output customers in a data file");
                System.out.println("17. Exit the program");
                System.out.println("Enter your choice (1-17): ");
                int choice = scanner.nextInt();
```

Cases code

```
switch (choice) {
    case 1:
        executeQueryForOption1(connection, scanner);
        break;
    case 2:
        executeQueryForOption2(connection, scanner);
        break;
    case 3:
        executeQueryForOption3(connection, scanner);
        break;
    case 17:
        // Option 4: Quit
        System.out.println("Exiting the program.");
        System.exit(0);
        break;
    case 4:
        executeQueryForOption4(connection, scanner);
        break;
    case 5:
        executeQueryForOption5(connection, scanner);
        break;
    case 6:
        executeQueryForOption6(connection, scanner);
        break;
    case 7:
        // Execute the query for option 7
        executeQueryForOption7(connection, scanner);
        break;
    case 8:
        // Execute the query for option 8
        executeQueryForOption8(connection, scanner);
        break;
    case 9:
        // Execute the query for option 9
        executeQueryForOption9(connection, scanner);
        break;
    case 10:
        // Execute the query for option 10
        executeQueryForOption10(connection, scanner);
        break;
```

```
                    break;
                case 11:
                    // Execute the query for option 11
                    executeQueryForOption11(connection, scanner);
                    break;
                case 12:
                    // Execute the query for option 12
                    executeQueryForOption12(connection, scanner);
                    break;
                case 13:
                    // Execute the query for option 13
                    executeQueryForOption13(connection, scanner);
                    break;
                case 14:
                    // Execute the query for option 14
                    executeQueryForOption14(connection, scanner);
                    break;
                case 15:
                    // Execute the query for option 15
                    executeQueryForOption15(connection, scanner);
                    break;
                case 16:
                    // Execute the query for option 16
                    executeQueryForOption16(connection, scanner);
                    break;
                default:
                    System.out.println("Invalid choice. Please enter a number between 1 and 17.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Case 1 code

```
    //Input taking for query 1
    private static void executeQueryForOption1(Connection connection, Scanner scanner) throws SQLException {
        // Get user input for customer details
        System.out.println("Enter Customer Name:");
        String name = scanner.next();

        System.out.println("Enter Customer Address:");
        String address = scanner.next();

        System.out.println("Enter Customer Category:");
        int category = scanner.nextInt();

     // Call the stored procedure to insert the new customer
        insertCustomer(connection, name, address, category); //Put user input in table

        System.out.println("New customer inserted successfully!");
    }
    //Connection to the DBMS for query 1
    public static void insertCustomer(Connection connection, String nameCustomer, String addressCustomer, int category) throws SQLException {
        String sql = "{call insertCustomer(?, ?, ?)}";
        try (CallableStatement statement = connection.prepareCall(sql)) {
            statement.setString(1, nameCustomer);
            statement.setString(2, addressCustomer);
            statement.setInt(3, category);
            statement.execute();
        }
    }
```

Case 2 code

```java
private static void executeQueryForOption2(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for department details
    System.out.println("Enter Department Number:");
    int deptNo = scanner.nextInt();

    System.out.println("Enter Department Data:");
    String deptData = scanner.next();

 // Call the stored procedure to insert the new department
    insertDepartment(connection, deptNo, deptData); //Put user input in table

    System.out.println("New department inserted successfully!");
}

public static void insertDepartment(Connection connection, int deptNo, String deptData) throws SQLException {
    String sql = "{call insertDepartment(?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, deptNo);
        statement.setString(2, deptData);
        statement.execute();
    }
}
```

Case 3 code

```
private static void executeQueryForOption3(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for process details
    System.out.println("Enter Process ID:");
    int processId = scanner.nextInt();

    System.out.println("Enter Department Number:");
    int deptNo = scanner.nextInt();

    System.out.println("Enter Process Data:");
    String processData = scanner.next();

    System.out.println("Select your process type: "
            + "1 = Fit process "
            + "2 = Paint process "
            + "3 = Cut process");
    int processType = scanner.nextInt();

    if (processType == 1) {
        System.out.println("Enter Fit Type:");
        String fitType = scanner.next();

        insertFitProcess(connection, processId, deptNo, processData, fitType);

        System.out.println("New fit type process inserted successfully!");
    } else if (processType == 2) {
        System.out.println("Enter Paint type:");
        String paintType = scanner.next();

        System.out.println("Enter painting method:");
        String paintMethod = scanner.next();

        insertPaintProcess(connection, processId, deptNo, processData, paintType, paintMethod);

        System.out.println("New paint type process inserted successfully!");
    } else if (processType == 3) {
        System.out.println("Enter cutting type:");
        String cutType = scanner.next();

        System.out.println("Enter machine type:");
        String machineType = scanner.next();

        insertCutProcess(connection, processId, deptNo, processData, cutType, machineType);

        System.out.println("New cut type process inserted successfully!");
    } else {
        System.out.println("Invalid process type");
    }
}
```

```
public static void insertFitProcess(Connection connection, int processId, int deptNo, String processData, String fitType) throws SQLException {
    String sql = "{call insertFitProcess(?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, processId);
        statement.setInt(2, deptNo);
        statement.setString(3, processData);
        statement.setString(4, fitType);
        statement.execute();
    }
}

public static void insertPaintProcess(Connection connection, int processId, int deptNo, String processData, String paintType, String paintMethod) throws SQLException {
    String sql = "{call insertPaintProcess(?, ?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, processId);
        statement.setInt(3, deptNo);
        statement.setString(2, processData);
        statement.setString(4, paintType);
        statement.setString(5, paintMethod);
        statement.execute();
    }
}

public static void insertCutProcess(Connection connection, int processId, int deptNo, String processData, String cutType, String machineType) throws SQLException {
    String sql = "{call insertCutProcess(?, ?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, processId);
        statement.setInt(3, deptNo);
        statement.setString(2, processData);
        statement.setString(4, cutType);
        statement.setString(5, machineType);
        statement.execute();
    }
}
```

Case 4 code

```
private static void executeQueryForOption4(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for assembly details
    System.out.println("Enter Assembly ID:");
    int assemblyId = scanner.nextInt();

    System.out.println("Enter Date Ordered:");
    String dateOrdered = scanner.next();

    System.out.println("Enter Assembly Details:");
    String assemblyDetails = scanner.next();

    System.out.println("Enter Customer Name:");
    String customerName = scanner.next();

    System.out.println("Enter the processes associated with the "
            + "assembly and separate them with a comma if there are several.");
    String processIds = scanner.next();

 // Call the stored procedure to insert the new performer (Option 1)
    insertAssembly(connection, assemblyId, dateOrdered, assemblyDetails, customerName, processIds); //Put user input in table

    System.out.println("New assembly inserted successfully!");
}

public static void insertAssembly(Connection connection, int assemblyId, String dateOrdered, String assemblyDetails, String customerName, String processIds) throws SQLException
    String sql = "{call InsertAssembly(?, ?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, assemblyId);
        statement.setString(2, dateOrdered);
        statement.setString(3, assemblyDetails);
        statement.setString(4, customerName);
        statement.setString(5, processIds);
        statement.execute();
    }
}
```

Case 5 code

```java
private static void executeQueryForOption5(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for account assembly details
    System.out.println("Enter Account Number:");
    int accountNo = scanner.nextInt();

    System.out.println("Enter Creation Date:");
    String creationDate = scanner.next();

    System.out.println("Select your account type: "
            + " 1 = Assembly account"
            + " 2 = Department account"
            + " 3 = Process account");
    int accountType = scanner.nextInt();

    if (accountType == 1) {
        System.out.println("Enter assembly account details:");
        int details1 = scanner.nextInt();
        System.out.println("Enter the associated assemly id: ");
        int assemblyId = scanner.nextInt();
        createAssemblyAccount(connection, accountNo, creationDate, details1, assemblyId); //Put user input in table
        System.out.println("New assembly account inserted successfully!");
    }

    else if (accountType == 2) {
        System.out.println("Enter department account details:");
        int details2 = scanner.nextInt();
        System.out.println("Enter the associated department number: ");
        int deptNo = scanner.nextInt();
        createDepartmentAccount(connection, accountNo, creationDate, details2, deptNo); //Put user input in table
        System.out.println("New department account inserted successfully!");
    }
    else if (accountType == 3) {
        System.out.println("Enter process account details:");
        int details3 = scanner.nextInt();
        System.out.println("Enter the associated process id: ");
        int processId = scanner.nextInt();
        createProcessAccount(connection, accountNo, creationDate, details3, processId); //Put user input in table
        System.out.println("New process account inserted successfully!");
    }

}


public static void createAssemblyAccount(Connection connection, int accountNo, String creationDate, int details1, int assemblyId) throws SQLException {
    String sql = "{call createAssemblyAccount(?, ?, ?, ?)}";
    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, accountNo);
        statement.setString(2, creationDate);
        statement.setInt(3, details1);
        statement.setInt(4, assemblyId);
        statement.executeUpdate();
    }
}

public static void createDepartmentAccount(Connection connection, int accountNo, String creationDate, int details2, int deptNo) throws SQLException {
    String sql = "{call createDepartmentAccount(?, ?, ?, ?)}";
    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, accountNo);
        statement.setString(2, creationDate);
        statement.setInt(3, details2);
        statement.setInt(4, deptNo);
        statement.executeUpdate();
    }
}

public static void createProcessAccount(Connection connection, int accountNo, String creationDate, int details3, int processId) throws SQLException {
    String sql = "{call createProcessAccount(?, ?, ?, ?)}";
    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, accountNo);
        statement.setString(2, creationDate);
        statement.setInt(3, details3);
        statement.setInt(4, processId);
        statement.executeUpdate();
    }
}
```

Case 6 code

```
private static void executeQueryForOption6(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for job details
    System.out.println("Enter Job Number:");
    int jobNo = scanner.nextInt();

    System.out.println("Enter Assembly ID:");
    int assemblyId = scanner.nextInt();

    System.out.println("Enter Process ID:");
    int processId = scanner.nextInt();

    System.out.println("Enter date the job commenced (YYYY-MM-DD):");
    String dateStart = scanner.next();

    System.out.println("Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)");
    int jobType = scanner.nextInt();

    switch (jobType) {
        case 1:
            insertJobFit(connection, jobNo, assemblyId, processId, dateStart);
            System.out.println("New FIT job inserted successfully!");
            break;
        case 2:
            insertJobPaint(connection, jobNo, assemblyId, processId, dateStart);
            System.out.println("New PAINT job inserted successfully!");
            break;
        case 3:
            insertJobCut(connection, jobNo, assemblyId, processId, dateStart);
            System.out.println("New CUT job inserted successfully!");
            break;
        default:
            System.out.println("Invalid job type");
    }
}
```

```
public static void insertJobFit(Connection connection, int jobNo, int assemblyId, int processId, String dateStart) throws SQLException {
    String sql = "{call insertJobFit(?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setInt(2, assemblyId);
        statement.setInt(3, processId);
        statement.setString(4, dateStart);
        statement.execute();
    }
}

public static void insertJobPaint(Connection connection, int jobNo, int assemblyId, int processId, String dateStart) throws SQLException {
    String sql = "{call insertJobPaint(?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setInt(2, assemblyId);
        statement.setInt(3, processId);
        statement.setString(4, dateStart);
        statement.execute();
    }
}

public static void insertJobCut(Connection connection, int jobNo, int assemblyId, int processId, String dateStart) throws SQLException {
    String sql = "{call insertJobCut(?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setInt(2, assemblyId);
        statement.setInt(3, processId);
        statement.setString(4, dateStart);
        statement.execute();
    }
}
```

Case 7 code

```java
private static void executeQueryForOption7(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for job details
    System.out.println("Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)");
    int jobType = scanner.nextInt();

    switch (jobType) {
        case 1:
            System.out.println("Enter Fit Job Number:");
            int fitJobNo = scanner.nextInt();

            System.out.println("Enter Completion Date (YYYY-MM-DD):");
            String completionDateFit = scanner.next();

            System.out.println("Enter Labor Time for Fit Job:");
            double laborTimeFit = scanner.nextDouble();

            // Call the stored procedure
            completeFitJob(connection, fitJobNo, completionDateFit, laborTimeFit);

            System.out.println("Fit Job completed successfully!");
            break;

        case 2:
            System.out.println("Enter Paint Job Number:");
            int paintJobNo = scanner.nextInt();

            System.out.println("Enter Completion Date (YYYY-MM-DD):");
            String completionDatePaint = scanner.next();

            System.out.println("Enter Paint Color:");
            String color = scanner.next();

            System.out.println("Enter Paint Volume:");
            int volume = scanner.nextInt();

            System.out.println("Enter Labor Time for Paint Job:");
            double laborTimePaint = scanner.nextDouble();

            // Call the stored procedure
            completePaintJob(connection, paintJobNo, completionDatePaint, color, volume, laborTimePaint);

            System.out.println("Paint Job completed successfully!");
            break;
```

```
        case 3:
            System.out.println("Enter Cut Job Number:");
            int cutJobNo = scanner.nextInt();

            System.out.println("Enter Completion Date (YYYY-MM-DD):");
            String completionDateCut = scanner.next();

            System.out.println("Enter Machine Type Used for Cut Job:");
            String machineType = scanner.next();

            System.out.println("Enter Time Machine Used for Cut Job:");
            double timeMachineUsed = scanner.nextDouble();

            System.out.println("Enter Material Used for Cut Job:");
            String materialUsed = scanner.next();

            System.out.println("Enter Labor Time for Cut Job:");
            double laborTimeCut = scanner.nextDouble();

            // Call the stored procedure
            completeCutJob(connection, cutJobNo, completionDateCut, machineType, timeMachineUsed, materialUsed, laborTimeCut);

            System.out.println("Cut Job completed successfully!");
            System.out.println("New CUT job inserted successfully!");
            break;
        default:
            System.out.println("Invalid job type");
    }
}
private static void completeFitJob(Connection connection, int jobNo, String dateEnd, double laborTime) throws SQLException {
    String sql = "{call completeFitJob(?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setString(2, dateEnd);
        statement.setDouble(3, laborTime);
        statement.execute();
    }
}
```

```
private static void completePaintJob(Connection connection, int jobNo, String dateEnd, String color, int volume, double laborTime) throws SQLException {
    String sql = "{call completePaintJob(?, ?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setString(2, dateEnd);
        statement.setString(3, color);
        statement.setInt(4, volume);
        statement.setDouble(5, laborTime);
        statement.execute();
    }
}

private static void completeCutJob(Connection connection, int jobNo, String dateEnd, String machineType, double timeMachineUsed, String materialUsed, double
    String sql = "{call completeCutJob(?, ?, ?, ?, ?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, jobNo);
        statement.setString(2, dateEnd);
        statement.setString(3, machineType);
        statement.setDouble(4, timeMachineUsed);
        statement.setString(5, materialUsed);
        statement.setDouble(6, laborTime);
        statement.execute();
    }
}
```

Case 8 code

```
    private static void executeQueryForOption8(Connection connection, Scanner scanner) throws SQLException {
    // Prompt the user to enter input for the stored procedure
    System.out.println("Enter Transaction Number:");
    int transactionNo = scanner.nextInt();

    System.out.println("Enter Supplier Cost:");
    double supCost = scanner.nextDouble();

    System.out.println("Enter Assembly Account Number (or enter 0 if not applicable):");
    int assemblyAccountNo = scanner.nextInt();

    System.out.println("Enter Process Account Number (or enter 0 if not applicable):");
    int processAccountNo = scanner.nextInt();

    System.out.println("Enter Department Account Number (or enter 0 if not applicable):");
    int deptAccountNo = scanner.nextInt();

    // Call the stored procedure
    executeEnterTransactionAndUpdateAccounts(connection, transactionNo, supCost, assemblyAccountNo, processAccountNo, deptAccountNo);

    System.out.println("Transaction successfully entered and accounts updated!");


private static void executeEnterTransactionAndUpdateAccounts(Connection connection, int transactionNo, double supCost, int assemblyAccountNo, int processAccou
    String sql = "{call enterTransactionAndUpdateAccounts(?, ?, ?, ?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, transactionNo);
        statement.setDouble(2, supCost);

        if (assemblyAccountNo == 0) {
            statement.setNull(3, java.sql.Types.INTEGER);
        } else {
            statement.setInt(3, assemblyAccountNo);
        }

        if (processAccountNo == 0) {
            statement.setNull(4, java.sql.Types.INTEGER);
        } else {
            statement.setInt(4, processAccountNo);
        }

        if (deptAccountNo == 0) {
            statement.setNull(5, java.sql.Types.INTEGER);
        } else {
            statement.setInt(5, deptAccountNo);
        }

        statement.execute();
    }
}
```

Case 9 code

```
    private static void executeQueryForOption9(Connection connection, Scanner scanner) throws SQLException {
        // Get user input for assembly ID
        System.out.println("Enter Assembly ID:");
        int assemblyId = scanner.nextInt();

        // Call the stored procedure
        double totalCost = executeCalculateTotalCostForAssembly(connection, assemblyId);

        // Display the result
        System.out.println("Total Cost for Assembly ID " + assemblyId + ": " + totalCost);
    }

    private static double executeCalculateTotalCostForAssembly(Connection connection, int assemblyId) throws SQLException {
        String sql = "{call calculateTotalCostForAssembly(?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameter
            statement.setInt(1, assemblyId);

            // Execute the stored procedure
            statement.execute();

            // Retrieve the total cost
            double totalCost;
            try (ResultSet rs = statement.getResultSet()) {
                if (rs.next()) {
                    totalCost = rs.getDouble(1); // Assuming the total cost is the first column
                } else {
                    totalCost = 0; // Handle the case where no value is returned
                }
            }

            return totalCost;
        }
    }
```

Case 10 code

```
private static void executeQueryForOption10(Connection connection, Scanner scanner) throws SQLException {
    // Get user input for department number and date
    System.out.println("Enter Department Number:");
    int deptNo = scanner.nextInt(); // Read department number
    scanner.nextLine(); // Consume the newline left-over

    System.out.println("Enter the date (YYYY-MM-DD):");
    String givenDate = scanner.nextLine(); // Read the given date

    // Call the stored procedure
    int totalLaborTime = executeTotalLaborTimeInDepartment(connection, deptNo, givenDate);

    // Display the result
    System.out.println("Total Labor Time for Department " + deptNo + " on " + givenDate + ": " + totalLaborTime);
}

private static int executeTotalLaborTimeInDepartment(Connection connection, int deptNo, String givenDate) throws SQLException {
    String sql = "{call TotalLaborTimeInDepartment_WithIndex(?, ?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, deptNo);
        statement.setString(2, givenDate);

        // Register the output parameter
        statement.registerOutParameter(3, Types.INTEGER);

        // Execute the stored procedure
        statement.execute();

        // Retrieve the total labor time from the output parameter
        int totalLaborTime = statement.getInt(3);

        return totalLaborTime;
    }
}
```

Case 11 code

```
private static void executeQueryForOption11(Connection connection, Scanner scanner) throws SQLException {
    System.out.println("Enter the assembly ID:");
    int assemblyId = scanner.nextInt(); // Read assembly ID

    retrieveProcessesForAssembly(connection, assemblyId);
}

private static void retrieveProcessesForAssembly(Connection connection, int givenAssemblyId) throws SQLException {
    String sql = "{call ProcessesForAssembly_WithIndex(?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, givenAssemblyId);

        boolean hasResults = statement.execute();

        if (hasResults) {
            try (ResultSet rs = statement.getResultSet()) {
                while (rs.next()) {
                    int assemblyId = rs.getInt("assembly_id");
                    int processId = rs.getInt("process_id");
                    String processData = rs.getString("process_data");
                    int deptNo = rs.getInt("dept_no");
                    String departmentResponsible = rs.getString("department_responsible");

                    System.out.println("Assembly ID: " + assemblyId + ", Process ID: " + processId +
                                       ", Process Data: " + processData + ", Department Number: " + deptNo +
                                       ", Department Responsible: " + departmentResponsible);
                }
            }
        } else {
            System.out.println("No processes found for assembly ID " + givenAssemblyId);
        }
    }
}
```

Case 12 code

```
private static void executeQueryForOption12(Connection connection, Scanner scanner) throws SQLException {
    System.out.println("Enter the start category range:");
    int startCategory = scanner.nextInt(); // Read start category
    System.out.println("Enter the end category range:");
    int endCategory = scanner.nextInt(); // Read end category

    retrieveCustomerByRange(connection, startCategory, endCategory);
}

private static void retrieveCustomerByRange(Connection connection, int categoryStart, int categoryEnd) throws SQLException {
    String sql = "{call retrieveCustomerByRange(?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setInt(1, categoryStart);
        statement.setInt(2, categoryEnd);

        boolean hasResults = statement.execute();

        if (hasResults) {
            try (ResultSet rs = statement.getResultSet()) {
                while (rs.next()) {
                    String nameCustomer = rs.getString("name_customer");
                    String addressCustomer = rs.getString("address_customer");
                    int category = rs.getInt("category");

                    System.out.println("Customer Name: " + nameCustomer + ", Address: " + addressCustomer +
                                       ", Category: " + category);
                }
            }
        } else {
            System.out.println("No customers found in the given category range " + categoryStart + " to " + categoryEnd);
        }
    }
}
```

Case 13 code

```
private static void executeQueryForOption13(Connection connection, Scanner scanner) throws SQLException {
    System.out.println("Enter the start job number:");
    int startJobNo = scanner.nextInt(); // Read start job number
    System.out.println("Enter the end job number:");
    int endJobNo = scanner.nextInt(); // Read end job number

    deleteCutJobsInRange(connection, startJobNo, endJobNo);

    System.out.println("Cut jobs in range " + startJobNo + " to " + endJobNo + " have been deleted.");
}

private static void deleteCutJobsInRange(Connection connection, int startJobNo, int endJobNo) throws SQLException {
    String sql = "{call deleteCutJobsInRange(?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, startJobNo);
        statement.setInt(2, endJobNo);

        // Execute the stored procedure
        statement.execute();
    }
}
```

Case 14 code

```
private static void executeQueryForOption14(Connection connection, Scanner scanner) throws SQLException {
    System.out.println("Enter the job number for the paint job:");
    int jobNo = scanner.nextInt(); // Read job number
    scanner.nextLine(); // Consume the newline left-over

    System.out.println("Enter the new color for the paint job:");
    String newColor = scanner.nextLine(); // Read the new color

    updatePaintJobColor(connection, jobNo, newColor);

    System.out.println("The color of paint job " + jobNo + " has been changed to " + newColor + ".");
}

private static void updatePaintJobColor(Connection connection, int jobNo, String newColor) throws SQLException {
    String sql = "{call updatePaintJobColor(?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, jobNo);
        statement.setString(2, newColor);

        // Execute the stored procedure
        statement.execute();
    }
}
```

Case 15 code

```
private static void executeQueryForOption15(Connection connection, Scanner scanner) {
    System.out.println("Enter the input file name:");
    String fileName = scanner.next(); // Read the file name

    try {
        // Open the file
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                // Assuming the file has comma-separated values (CSV) like name,address,category
                String[] customerData = line.split(",");
                if (customerData.length == 3) {
                    insertExportedCustomer(connection, customerData[0], customerData[1], Integer.parseInt(customerData[2]));
                }
            }
        }
        System.out.println("All customers from the file have been inserted.");
    } catch (FileNotFoundException e) {
        System.out.println("File not found: " + fileName);
    } catch (IOException e) {
        System.out.println("An error occurred while reading the file.");
    } catch (SQLException e) {
        System.out.println("An error occurred while inserting the customer data into the database.");
        e.printStackTrace();
    } catch (NumberFormatException e) {
        System.out.println("Category must be an integer.");
    }
}

private static void insertExportedCustomer(Connection connection, String name, String address, int category) throws SQLException {
    String sql = "{call insertCustomer(?, ?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        statement.setString(1, name.trim());
        statement.setString(2, address.trim());
        statement.setInt(3, category); // No need to convert to integer as it is already an integer

        statement.executeUpdate();
    }
}
```

Case 16 code

```
private static void executeQueryForOption16(Connection connection, Scanner scanner) throws SQLException {
    System.out.println("Enter the start category range:");
    int startCategory = scanner.nextInt(); // Read start category
    System.out.println("Enter the end category range:");
    int endCategory = scanner.nextInt(); // Read end category
    scanner.nextLine(); // Consume the newline left-over

    System.out.println("Enter the output file name:");
    String outputFileName = scanner.nextLine(); // Read the output file name

    exportCustomersToDataFileUsingStoredProc(connection, startCategory, endCategory, outputFileName);
}

private static void exportCustomersToDataFileUsingStoredProc(Connection connection, int startCategory, int endCategory, String outputFileName) throws SQLExcep
    String sql = "{call retrieveCustomerByRange(?, ?)}";
    try (CallableStatement statement = connection.prepareCall(sql);
         BufferedWriter writer = new BufferedWriter(new FileWriter(outputFileName))) {

        statement.setInt(1, startCategory);
        statement.setInt(2, endCategory);
        ResultSet rs = statement.executeQuery();

        while (rs.next()) {
            String nameCustomer = rs.getString("name_customer");
            String addressCustomer = rs.getString("address_customer");
            int category = rs.getInt("category");
            String customerRecord = nameCustomer + "," + addressCustomer + "," + category;

            writer.write(customerRecord);
            writer.newLine(); // To move to the next line for the next record
        }

        System.out.println("Customer data has been written to " + outputFileName);
    } catch (IOException e) {
        System.out.println("An error occurred while writing to the file: " + e.getMessage());
    }
}
```

Case 17 code

```
case 17:
    // Option 4: Quit
    System.out.println("Exiting the program.");
    System.exit(0);
    break;
```

Task 6. Java program Execution

6.1 Screenshots showing the testing of query 1.

Test 1

```
17. Exit the program
Enter your choice (1-17):
1
Enter Customer Name:
Kate
Enter Customer Address:
Oklahoma
Enter Customer Category:
1
```

Results | Messages

	name_customer	address_customer	category
1	Kate	Oklahoma	1

Test 2

```
Enter your choice (1-17):
1
Enter Customer Name:
Lili
Enter Customer Address:
Edmond
Enter Customer Category:
1
New customer inserted successfully!
```

Results | Messages

	name_customer	address_customer	category
1	Kate	Oklahoma	1
2	Lili	Edmond	1

Test 3

```
Enter your choice (1-17):
1
Enter Customer Name:
Bob
Enter Customer Address:
Italy
Enter Customer Category:
4
```

	name_customer	address_customer	category
1	Bob	Italy	4
2	Kate	Oklahoma	1
3	Lili	Edmond	1

Test 4

```
Enter your choice (1-17):
1
Enter Customer Name:
Anna
Enter Customer Address:
NYC
Enter Customer Category:
3
```

	name_customer	address_customer	category
1	Anna	NYC	3
2	Bob	Italy	4
3	Kate	Oklahoma	1
4	Lili	Edmond	1

Test 5

```
Enter your choice (1-17):
1
Enter Customer Name:
Wung
Enter Customer Address:
Nepal
Enter Customer Category:
5
New customer inserted successfully!
```

	name_customer	address_customer	category
1	Anna	NYC	3
2	Bob	Italy	4
3	Kate	Oklahoma	1
4	Lili	Edmond	1
5	Wung	Nepal	5

6.2 Screenshots showing the testing of query 2.

Test 1

```
Enter your choice (1-17):
2
Enter Department Number:
1
Enter Department Data:
Assembly
New department inserted successfully!
```

	dept_no	dept_data
1	1	Assembly

Test 2

```
Enter your choice (1-17):
2
Enter Department Number:
2
Enter Department Data:
Process
New department inserted successfully!
```

Results Messages

	dept_no	dept_data
1	1	Assembly
2	2	Process

Test 3

```
Enter your choice (1-17):
2
Enter Department Number:
3
Enter Department Data:
PaintType
New department inserted successfully!
```

	dept_no	dept_data
1	1	Assembly
2	2	Process
3	3	PaintType

Test 4

```
Enter your choice (1-17):
2
Enter Department Number:
4
Enter Department Data:
FitType
New department inserted successfully!
```

	dept_no	dept_data
1	1	Assembly
2	2	Process
3	3	PaintType
4	4	FitType

Test 5

```
Enter your choice (1-17):
2
Enter Department Number:
5
Enter Department Data:
favType
New department inserted successfully!
```

	dept_no	dept_data
1	1	Assembly
2	2	Process
3	3	PaintType
4	4	FitType
5	5	favType

6.3 Screenshots showing the testing of query 3

Test 1

```
Enter your choice (1-17):
3
Enter Process ID:
101
Enter Department Number:
4
Enter Process Data:
fittingProcess
Select your process type: 1 = Fit process2 = Paint process3 = Cut process
1
Enter Fit Type:
TypeA
New fit type process inserted successfully!
```

	process_id	process_data	dept_no	fit_type
1	101	fittingProcess	4	TypeA

Test 2

```
Enter your choice (1-17):
3
Enter Process ID:
201
Enter Department Number:
3
Enter Process Data:
PaintProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
2
Enter Paint type:
Acrylic
Enter painting method:
Brush
New paint type process inserted successfully!
```

	process_id	process_data	dept_no	paint_type	painting_method
1	201	PaintProcess	3	Acrylic	Brush

Test 3

```
Enter your choice (1-17):
3
Enter Process ID:
301
Enter Department Number:
5
Enter Process Data:
CutProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
3
Enter cutting type:
Angular
Enter machine type:
Sharp
New cut type process inserted successfully!
```

	process_id	process_data	dept_no	cutting_type	machine_type
1	301	CutProcess	5	Angular	Sharp

Test 4

```
17. Exit the program
Enter your choice (1-17):
3
Enter Process ID:
302
Enter Department Number:
5
Enter Process Data:
CutProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
3
Enter cutting type:
Circles
Enter machine type:
DS156
New cut type process inserted successfully!
```

	process_id	process_data	dept_no	cutting_type	machine_type
1	301	CutProcess	5	Angular	Sharp
2	302	CutProcess	5	Circles	DS156

Test 5

```
17. Exit the program
Enter your choice (1-17):
3
Enter Process ID:
202
Enter Department Number:
3
Enter Process Data:
PaintProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
2
Enter Paint type:
Oil
Enter painting method:
Hands
New paint type process inserted successfully!
```

	process_id	process_data	dept_no	paint_type	painting_method
1	201	PaintProcess	3	Acrylic	Brush
2	202	PaintProcess	3	Oil	Hands

Test 6

```
17. Exit the program
Enter your choice (1-17):
3
Enter Process ID:
102
Enter Department Number:
4
Enter Process Data:
FitType
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut proce
1
Enter Fit Type:
TypeB
New fit type process inserted successfully!
```

	process_id	process_data	dept_no	fit_type
1	101	fittingProcess	4	TypeA
2	102	FitType	4	TypeB

Test 7

```
Enter your choice (1-17):
3
Enter Process ID:
103
Enter Department Number:
4
Enter Process Data:
FitProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
1
Enter Fit Type:
TypeC
New fit type process inserted successfully!
```

Results Messages

	process_id	process_data	dept_no	fit_type
1	101	fittingProcess	4	TypeA
2	102	FitType	4	TypeB
3	103	FitProcess	4	TypeC

Test 8

```
Enter your choice (1-17):
3
Enter Process ID:
203
Enter Department Number:
10
Enter Process Data:
AProcess
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
2
Enter Paint type:
blublu
Enter painting method:
brush
New paint type process inserted successfully!
```

	process_id	process_data	dept_no	paint_type	painting_method
1	201	PaintProcess	3	Acrylic	Brush
2	202	PaintProcess	3	Oil	Hands
3	203	AProcess	10	blublu	brush

Test 9

```
Enter your choice (1-17):
3
Enter Process ID:
303
Enter Department Number:
5
Enter Process Data:
CutP
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
3
Enter cutting type:
round
Enter machine type:
K120
New cut type process inserted successfully!
```

	process_id	process_data	dept_no	cutting_type	machine_type
1	301	CutProcess	5	Angular	Sharp
2	302	CutProcess	5	Circles	DS156
3	303	CutP	5	round	K120

Test 10

```
Enter your choice (1-17):
3
Enter Process ID:
304
Enter Department Number:
5
Enter Process Data:
CutP
Select your process type: 1 = Fit process 2 = Paint process 3 = Cut process
3
Enter cutting type:
Straight
Enter machine type:
laser
New cut type process inserted successfully!
```

	process_id	process_data	dept_no	cutting_type	machine_type
1	301	CutProcess	5	Angular	Sharp
2	302	CutProcess	5	Circles	DS156
3	303	CutP	5	round	K120
4	304	CutP	5	Straight	laser

6.4 Screenshots showing the testing of query 4

Test 1

```
Enter your choice (1-17):
4
Enter Assembly ID:
11
Enter Date Ordered:
23-11-01
Enter Assembly Details:
Something1
Enter Customer Name:
Anna
Enter the processes associated with the assembly and separate them with a comma if there are several.
301
New assembly inserted successfully!
```

Assembly table

	assembly_id	date_ordered	assembly_details	customer_name
1	11	23-11-01	Something1	Anna

Impacted Manufacture table

	assembly_id	process_id	job_no
1	11	301	NULL

Test 2

```
17. Exit the program
Enter your choice (1-17):
4
Enter Assembly ID:
20
Enter Date Ordered:
2023-11-01
Enter Assembly Details:
Something2
Enter Customer Name:
Yasmina
Enter the processes associated with the assembly and separate them with a comma if there ar
101
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembl…	customer_n…
1	20	2023-11-01	Something2	Yasmina

	assembly_id	process_id	job_no
1	20	101	NULL

Test 3

```
17. Exit the program
Enter your choice (1-17):
4
Enter Assembly ID:
21
Enter Date Ordered:
2023-11-02
Enter Assembly Details:
Something3
Enter Customer Name:
Kayla
Enter the processes associated with the assembly and separate them with a comma if there ar
102,103,104
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL

Test 4

```
Enter your choice (1-17):
4
Enter Assembly ID:
22
Enter Date Ordered:
2023-11-03
Enter Assembly Details:
Something4
Enter Customer Name:
Mira
Enter the processes associated with the assembly and separate them with a comma if there ar
101
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL

Test 5

```
Enter your choice (1-17):
4
Enter Assembly ID:
23
Enter Date Ordered:
2024-11-09
Enter Assembly Details:
Something5
Enter Customer Name:
Lili
Enter the processes associated with the assembly and separate them with a comma if there ar
000
New assembly inserted successfully!
```

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili

Test 6

```
Enter your choice (1-17):
4
Enter Assembly ID:
24
Enter Date Ordered:
2024-12-9
Enter Assembly Details:
Something6
Enter Customer Name:
Jack
Enter the processes associated with the assembly and separate them with a comma if there ar
105
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili
5	24	2024-12-9	Something6	Jack

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL
7	24	105	NULL

Test 7

```
Enter your choice (1-17):
4
Enter Assembly ID:
25
Enter Date Ordered:
2024-3-8
Enter Assembly Details:
Something7
Enter Customer Name:
Mina
Enter the processes associated with the assembly and separate them with a comma if there ar
106
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili
5	24	2024-12-9	Something6	Jack
6	25	2024-3-8	Something7	Mina

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL
7	24	105	NULL
8	25	106	NULL

Test 8

```
17. Exit the program
Enter your choice (1-17):
4
Enter Assembly ID:
26
Enter Date Ordered:
2023-11-27
Enter Assembly Details:
29
Enter Customer Name:
Bob
Enter the processes associated with the assembly and separate them with a comma if there ar
108
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili
5	24	2024-12-9	Something6	Jack
6	25	2024-3-8	Something7	Mina
7	26	2023-11-27	29	Bob

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL
7	24	105	NULL
8	25	106	NULL
9	26	108	NULL

Test 9

```
Enter your choice (1-17):
4
Enter Assembly ID:
30
Enter Date Ordered:
2021-1-1
Enter Assembly Details:
109
Enter Customer Name:
Ethan
Enter the processes associated with the assembly and separate them with a comma if there ar
99
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili
5	24	2024-12-9	Something6	Jack
6	25	2024-3-8	Something7	Mina
7	26	2023-11-27	29	Bob
8	30	2021-1-1	109	Ethan

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL
7	24	105	NULL
8	25	106	NULL
9	26	108	NULL
10	30	99	NULL

Test 10

```
Enter your choice (1-17):
4
Enter Assembly ID:
31
Enter Date Ordered:
2021-09-11
Enter Assembly Details:
Details
Enter Customer Name:
Luck
Enter the processes associated with the assembly and separate them with a comma if there ar
199
New assembly inserted successfully!
```

	assembly_id	date_ordered	assembly_details	customer_name
1	20	2023-11-01	Something2	Yasmina
2	21	2023-11-02	Something3	Kayla
3	22	2023-11-03	Something4	Mira
4	23	2024-11-09	Something5	Lili
5	24	2024-12-9	Something6	Jack
6	25	2024-3-8	Something7	Mina
7	26	2023-11-27	29	Bob
8	30	2021-1-1	109	Ethan
9	31	2021-09-11	Details	Luck

	assembly_id	process_id	job_no
1	20	101	NULL
2	21	102	NULL
3	21	103	NULL
4	21	104	NULL
5	22	101	NULL
6	23	0	NULL
7	24	105	NULL
8	25	106	NULL
9	26	108	NULL
10	30	99	NULL
11	31	199	NULL

6.5 Screenshots showing the testing of query 5

Test 1

```
Enter your choice (1-17):
5
Enter Account Number:
101010
Enter Creation Date:
23-11-02
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
1
Enter assembly account details:
0
Enter the associated assemly id:
11
New assembly account inserted successfully!
```

Assembly account

	account_no	creation_date	details_1
1	101010	23-11-02	0

Associated record_assembly table

	account_no	assembly_id
1	101010	1

Test 2

```
Enter your choice (1-17):
5
Enter Account Number:
101010
Enter Creation Date:
219299
Select your account type:  1 = Assembly account 2 = Department account 3 = Process accour
2
Enter department account details:
0
Enter the associated department number:
1
New department account inserted successfully!
```

	account_no	creation_date	details_2
1	101010	219299	0

(account dept)

	account_no	dept_no
1	101010	1

(record_dept)

Test 3

```
Enter your choice (1-17):
5
Enter Account Number:
303030
Enter Creation Date:
223344
Select your account type:  1 = Assembly account 2 = Department account 3 = Process accour
3
Enter process account details:
0
Enter the associated process id:
201
New process account inserted successfully!
```

	account_no	creation_date	details_3
1	303030	223344	0

(account_process)

	account_no	process_id
1	303030	201

(record_process)

Test 4

```
Enter your choice (1-17):
5
Enter Account Number:
1
Enter Creation Date:
23-11-2
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
1
Enter assembly account details:
0
Enter the associated assemly id:
202
New assembly account inserted successfully!
```

	account_no	creation_date	details…
1	1	23-11-2	0

	account_no	assembly_id
1	1	202

Test 5

```
Enter your choice (1-17):
5
Enter Account Number:
2
Enter Creation Date:
2023-11-99
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
2
Enter department account details:
0
Enter the associated department number:
301
New department account inserted successfully!
```

	account_no	creation_date	details_2
1	2	2023-11-99	0

	account_no	dept_no
1	2	301

Test 6

```
Enter your choice (1-17):
5
Enter Account Number:
3
Enter Creation Date:
24-0-9
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
3
Enter process account details:
0
Enter the associated process id:
101
New process account inserted successfully!
```

	account_no	creation_date	details_3
1	3	24-0-9	0

	account_no	process_id
1	3	101

Test 7

```
17. Exit the program
Enter your choice (1-17):
5
Enter Account Number:
4
Enter Creation Date:
23-9-6
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
3
Enter process account details:
100
Enter the associated process id:
102
New process account inserted successfully!
```

	account_no	creation_date	details_3
1	3	24-0-9	0
2	4	23-9-6	100

	account_no	process_id
1	3	101
2	4	102

Test 8

```
Enter your choice (1-17):
5
Enter Account Number:
5
Enter Creation Date:
2024-09-8
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
3
Enter process account details:
89
Enter the associated process id:
103
New process account inserted successfully!
```

	account_no	creation_date	details_3
1	3	24-0-9	0
2	4	23-9-6	100
3	5	2024-09-8	89

	account_no	process_id
1	3	101
2	4	102
3	5	103

Test 9

```
Enter your choice (1-17):
5
Enter Account Number:
6
Enter Creation Date:
2024-77-8
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
1
Enter assembly account details:
9
Enter the associated assemly id:
2023
New assembly account inserted successfully!
```

	account_no	creation_date	details_1
1	6	2024-77-8	9

	account_no	assembly_id
1	1	202
2	6	2023

Test 10

```
Enter your choice (1-17):
5
Enter Account Number:
7
Enter Creation Date:
2024-01-01
Select your account type:  1 = Assembly account 2 = Department account 3 = Process account
2
Enter department account details:
0
Enter the associated department number:
302
New department account inserted successfully!
```

	account_no	creation_date	details_2
1	2	2023-11-99	0
2	7	2024-01-01	0

	account_no	dept_no
1	2	301
2	7	302

6.6 Screenshots showing the testing of query 6

Test 1

```
Enter your choice (1-17):
6
Enter Job Number:
1
Enter Assembly ID:
11
Enter Process ID:
101
Enter date the job commenced (YYYY-MM-DD):
2023-08-09
Enter the job type
1
New FIT job inserted successfully!
```

	job_no	date_start	date_end	labor_time
1	1	2023-08-09	NULL	NULL

(job fit table)

	assembly_id	process_id	job_no
1	11	101	1

(impacted manufacture table)

Test 2

```
Enter your choice (1-17):
6
Enter Job Number:
12
Enter Assembly ID:
102
Enter Process ID:
111
Enter date the job commenced (YYYY-MM-DD):
2023-09-10
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut
2
New PAINT job inserted successfully!
```

job_no	date_start	date_end	labor_time	color	volume
12	2023-09-10	NULL	NULL	NULL	NULL

(job paint table)

	assembly_id	process_id	job_no
1	11	101	1
2	102	111	12

(affected manufacture table)

Test 3

```
Enter your choice (1-17):
6
Enter Job Number:
13
Enter Assembly ID:
44
Enter Process ID:
113
Enter date the job commenced (YYYY-MM-DD):
2023-00-00
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
New CUT job inserted successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	13	2	NULL	NULL	NULL	NULL	NULL

(job cut table)

	assembly_id	process_id	job_no
1	11	101	1
2	44	113	13
3	102	111	12

(impacted manufacture table)

Test 4

```
Enter your choice (1-17):
6
Enter Job Number:
1
Enter Assembly ID:
11
Enter Process ID:
111
Enter date the job commenced (YYYY-MM-DD):
20-00-9
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
New FIT job inserted successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1

Test 5

```
Enter your choice (1-17):
6
Enter Job Number:
2
Enter Assembly ID:
22
Enter Process ID:
222
Enter date the job commenced (YYYY-MM-DD):
11-11-11
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
New FIT job inserted successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	*NULL*	*NULL*
2	2	11-11-11	*NULL*	*NULL*

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2

Test 6

```
Enter your choice (1-17):
6
Enter Job Number:
3
Enter Assembly ID:
33
Enter Process ID:
333
Enter date the job commenced (YYYY-MM-DD):
22-22-22
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
New FIT job inserted successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	NULL	NULL
2	2	11-11-11	NULL	NULL
3	3	22-22-22	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2
3	33	333	3

Test 7

```
Enter your choice (1-17):
6
Enter Job Number:
4
Enter Assembly ID:
44
Enter Process ID:
444
Enter date the job commenced (YYYY-MM-DD):
44-44-44
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
2
New PAINT job inserted successfully!
```

	job_no	date_start	date_end	labor_time	color	volume
1	4	44-44-44	NULL	NULL	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2
3	33	333	3
4	44	444	4

Test 8

```
Enter your choice (1-17):
6
Enter Job Number:
5
Enter Assembly ID:
55
Enter Process ID:
555
Enter date the job commenced (YYYY-MM-DD):
55-55-55
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
2
New PAINT job inserted successfully!
```

	job_no	date_start	date_end	labor_time	color	volume
1	4	44-44-44	NULL	NULL	NULL	NULL
2	5	55-55-55	NULL	NULL	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2
3	33	333	3
4	44	444	4
5	55	555	5

Test 9

```
17. Exit the program
Enter your choice (1-17):
6
Enter Job Number:
6
Enter Assembly ID:
66
Enter Process ID:
666
Enter date the job commenced (YYYY-MM-DD):
22-22-22
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
New CUT job inserted successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	6	2	NULL	NULL	NULL	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2
3	33	333	3
4	44	444	4
5	55	555	5
6	66	666	6

Test 10

```
Enter your choice (1-17):
6
Enter Job Number:
7
Enter Assembly ID:
77
Enter Process ID:
777
Enter date the job commenced (YYYY-MM-DD):
77-77-77
Enter the job type (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
New CUT job inserted successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	6	2	NULL	NULL	NULL	NULL	NULL
2	7	7	NULL	NULL	NULL	NULL	NULL

	assembly_id	process_id	job_no
1	11	111	1
2	22	222	2
3	33	333	3
4	44	444	4
5	55	555	5
6	66	666	6
7	77	777	7

6.7 Screenshots showing the testing of query 7

Test 1

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
Enter Fit Job Number:
1
Enter Completion Date (YYYY-MM-DD):
2024-08-09
Enter Labor Time for Fit Job:
12
Fit Job completed successfully!
```

	job_no	date_start	date_end	labor_time
1	1	2023-08-09	2024-08-09	12

Test 2

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
2
Enter Paint Job Number:
12
Enter Completion Date (YYYY-MM-DD):
2024-09-10
Enter Paint Color:
yellow
Enter Paint Volume:
5
Enter Labor Time for Paint Job:
10
Paint Job completed successfully!
```

	job_no	date_start	date_end	labor_time	color	volume
1	12	2023-09-10	2024-09-10	10	yellow	5

Test 3

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
Enter Cut Job Number:
13
Enter Completion Date (YYYY-MM-DD):
2024-09-8
Enter Machine Type Used for Cut Job:
laser
Enter Time Machine Used for Cut Job:
89
Enter Material Used for Cut Job:
hands
Enter Labor Time for Cut Job:
10
Cut Job completed successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	13	2	2024-09-8	10	laser	89.00	hands

Test 4

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
Enter Fit Job Number:
1
Enter Completion Date (YYYY-MM-DD):
25-8-8
Enter Labor Time for Fit Job:
60
Fit Job completed successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	25-8-8	60
2	2	11-11-11	*NULL*	*NULL*
3	3	22-22-22	*NULL*	*NULL*

Test 5

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
1
Enter Fit Job Number:
2
Enter Completion Date (YYYY-MM-DD):
26-8-8
Enter Labor Time for Fit Job:
89
Fit Job completed successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	25-8-8	60
2	2	11-11-11	26-8-8	89
3	3	22-22-22	*NULL*	*NULL*

Test 6

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut
1
Enter Fit Job Number:
3
Enter Completion Date (YYYY-MM-DD):
27-8-8
Enter Labor Time for Fit Job:
123
Fit Job completed successfully!
```

	job_no	date_start	date_end	labor_time
1	1	20-00-9	25-8-8	60
2	2	11-11-11	26-8-8	89
3	3	22-22-22	27-8-8	123

Test 7

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
2
Enter Paint Job Number:
4
Enter Completion Date (YYYY-MM-DD):
26-9-9
Enter Paint Color:
Bleue
Enter Paint Volume:
5
Enter Labor Time for Paint Job:
290
Paint Job completed successfully!
```

	job_no	date_start	date_end	labor_time	color	volume
1	4	44-44-44	26-9-9	290	Bleue	5
2	5	55-55-55	*NULL*	*NULL*	*NULL*	*NULL*

Test 8

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
2
Enter Paint Job Number:
5
Enter Completion Date (YYYY-MM-DD):
2023-9-9
Enter Paint Color:
Yellow
Enter Paint Volume:
23
Enter Labor Time for Paint Job:
100
Paint Job completed successfully!
```

	job_no	date_start	date_end	labor_time	color	volume
1	4	44-44-44	26-9-9	290	Bleue	5
2	5	55-55-55	2023-9-9	100	Yellow	23

Test 9

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
Enter Cut Job Number:
6
Enter Completion Date (YYYY-MM-DD):
20-9-8
Enter Machine Type Used for Cut Job:
Laser
Enter Time Machine Used for Cut Job:
99
Enter Material Used for Cut Job:
Cissors
Enter Labor Time for Cut Job:
190
Cut Job completed successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	6	2	20-9-8	190	Laser	99.00	Cissors
2	7	7	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Test 10

```
Enter your choice (1-17):
7
Enter the job type you want to update (1 = Job Fit, 2 = Job Paint, 3 = Job Cut)
3
Enter Cut Job Number:
7
Enter Completion Date (YYYY-MM-DD):
2024-11-10
Enter Machine Type Used for Cut Job:
screwdriver
Enter Time Machine Used for Cut Job:
88
Enter Material Used for Cut Job:
hammer
Enter Labor Time for Cut Job:
179
Cut Job completed successfully!
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	6	2	20-9-8	190	Laser	99.00	Cissors
2	7	7	2024-11-10	179	screwdriver	88.00	hammer

6.8 Screenshots showing the testing of query 8

Test 1

```
Enter your choice (1-17):
8
Enter Transaction Number:
1
Enter Supplier Cost:
100
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	transaction_no	sup_cost
1	1	100

(transaction table)

Original content of assembly account

	account_no	creation_date	details_1
1	1010	2023-09-11	0

Then this java code is ran

```
Enter your choice (1-17):
8
Enter Transaction Number:
2
Enter Supplier Cost:
200
Enter Assembly Account Number (or enter 0 if not applicable):
1010
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	transaction_no	sup_cost
1	1	100
2	2	200

Updated assembly account

	account_no	creation_date	details_1
1	1010	2023-09-11	200

Test 2

```
Enter your choice (1-17):
8
Enter Transaction Number:
3
Enter Supplier Cost:
400
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
303030
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	account_no	creation_date	details_3
1	303030	223344	0

(original account process)

	account_no	creation_date	details_3
1	303030	223344	400

(updated account process)

	transaction_no	sup_cost
1	1	100
2	2	200
3	3	400

Test 3

```
Enter your choice (1-17):
8
Enter Transaction Number:
4
Enter Supplier Cost:
250
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
2020
Transaction successfully entered and accounts updated!
```

	account_no	creation_date	details_2
1	2020	2026-09-11	0

(original account department)

	account_no	creation_date	details_2
1	2020	2026-09-11	250

(updated department account)

	transaction_no	sup_cost
1	1	100
2	2	200
3	3	400
4	4	250

Original accounts for the following tests:

	account_no	creation_date	details_1
1	6	2024-77-8	9

(assembly account)

	account_no	creation_date	details_3
1	3	24-0-9	0
2	4	23-9-6	100
3	5	2024-09-8	89

(process account)

	account_no	creation_date	details_2
1	2	2023-11-99	0
2	7	2024-01-01	0

(department account)

Test 4

```
Enter your choice (1-17):
8
Enter Transaction Number:
1
Enter Supplier Cost:
100
Enter Assembly Account Number (or enter 0 if not applicable):
6
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	account_no	creation_date	details_1
1	6	2024-77-8	109

	transaction_no	sup_cost
1	1	100
2	2	200

Test 5

```
17. Exit the program
Enter your choice (1-17):
8
Enter Transaction Number:
2
Enter Supplier Cost:
200
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
3
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	account_no	creation_date	details_3
1	3	24-0-9	200
2	4	23-9-6	100
3	5	2024-09-8	89

(Updated assembly account)

	transaction_no	sup_cost
1	1	100
2	2	200

Test 6 & Test 7

```
Enter your choice (1-17):

8
Enter Transaction Number:
3
Enter Supplier Cost:
100
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
4
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

```
Enter your choice (1-17):
8
Enter Transaction Number:
4
Enter Supplier Cost:
100
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
5
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

Output test 6 and test 7

	account_no	creation_date	details_3
1	3	24-0-9	200
2	4	23-9-6	200
3	5	2024-09-8	189

(updated process accounts)

	transaction_no	sup_cost
1	1	100
2	2	200
3	3	100
4	4	100

Test 8 & Test 9

```
Enter your choice (1-17):
8
Enter Transaction Number:
5
Enter Supplier Cost:
244
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
2
Transaction successfully entered and accounts updated!
```

```
Enter your choice (1-17):
8
Enter Transaction Number:
6
Enter Supplier Cost:
200
Enter Assembly Account Number (or enter 0 if not applicable):
0
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
7
Transaction successfully entered and accounts updated!
```

Output test 8 & test 9

	account_no	creation_date	details_2
1	2	2023-11-99	244
2	7	2024-01-01	200

(updated department accounts)

	transaction_no	sup_cost
1	1	100
2	2	200
3	3	100
4	4	100
5	5	244
6	6	200
7	7	50

Test 10

```
Enter your choice (1-17):
8
Enter Transaction Number:
7
Enter Supplier Cost:
50
Enter Assembly Account Number (or enter 0 if not applicable):
1
Enter Process Account Number (or enter 0 if not applicable):
0
Enter Department Account Number (or enter 0 if not applicable):
0
Transaction successfully entered and accounts updated!
```

	account_no	creation_date	details_1
1	6	2024-77-8	109

(no update on assembly account, I did not yet have an account number 1).

	transaction_no	sup_cost
1	1	100
2	2	200
3	3	100
4	4	100
5	5	244
6	6	200
7	7	50

6.9 Screenshots showing the testing of query 9

Note that to be able to execute this query via java and follow it easily, I manually populated the necessary tables so I could verify that my logic worked when I get my output. The way I populated the tables, will also be attached.

Population of the table

```
-- Insert sample data into the assembly table
INSERT INTO assembly (assembly_id, date_ordered, assembly_details, customer_name)
VALUES
(1, '2023-01-01', 'Details for Assembly 1', 'Customer A'),
(2, '2023-01-02', 'Details for Assembly 2', 'Customer B'),
(3, '2023-01-03', 'Details for Assembly 3', 'Customer C'),
(4, '2023-01-04', 'Details for Assembly 4', 'Customer D'),
(5, '2023-01-05', 'Details for Assembly 5', 'Customer E'),
(6, '2023-01-06', 'Details for Assembly 6', 'Customer F'),
(7, '2023-01-07', 'Details for Assembly 7', 'Customer G'),
(8, '2023-01-08', 'Details for Assembly 8', 'Customer H'),
(9, '2023-01-09', 'Details for Assembly 9', 'Customer I'),
(10, '2023-01-10', 'Details for Assembly 10', 'Customer J');

-- Insert sample data into the account_assembly table with arbitrary details
INSERT INTO account_assembly (account_no, creation_date, details_1)
VALUES
(1, '2023-01-11', 1000),
(2, '2023-01-12', 2000),
(3, '2023-01-13', 3000),
(4, '2023-01-14', 4000),
(5, '2023-01-15', 5000),
(6, '2023-01-16', 6000),
(7, '2023-01-17', 7000),
(8, '2023-01-18', 8000),
(9, '2023-01-19', 9000),
(10, '2023-01-20', 10000);
```

```sql
-- Insert sample data into the record_assembly table
-- Assuming each account_no in account_assembly corresponds to an assembly_id
INSERT INTO record_assembly (account_no, assembly_id)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10);

-- Insert sample data into the transactions table with increasing supply costs
INSERT INTO transactions (transaction_no, sup_cost)
VALUES
(1, 100),
(2, 200),
(3, 300),
(4, 400),
(5, 500),
(6, 600),
(7, 700),
(8, 800),
(9, 900),
(10, 1000);
```

```sql
-- Insert sample data into the updates table linking transactions to account_assembly
-- Assuming that each transaction is linked to a corresponding account_assembly
INSERT INTO updates (transaction_no, account_no_assembly, account_no_process, account_no_department)
VALUES
(1, 1, NULL, NULL),
(2, 2, NULL, NULL),
(3, 3, NULL, NULL),
(4, 4, NULL, NULL),
(5, 5, NULL, NULL),
(6, 6, NULL, NULL),
(7, 7, NULL, NULL),
(8, 8, NULL, NULL),
(9, 9, NULL, NULL),
(10, 10, NULL, NULL);
```

Test 1

```
17. Exit the program
Enter your choice (1-17):
9
Enter Assembly ID:
1
Total Cost for Assembly ID 1: 100.0
```

Test 2

```
Enter your choice (1-17):
9
Enter Assembly ID:
4
Total Cost for Assembly ID 4: 400.0
```

Test 3

```
Enter your choice (1-17):
9
Enter Assembly ID:
10
Total Cost for Assembly ID 10: 1000.0
```

6.10 Screenshots showing the testing of query 10

Find attached the way I populated the tables involved in the stored procedure to be able to keep track of how everything works. Note that I used the same end date in the job for simplicity when doing the 10 test.

Test 1

```
Enter your choice (1-17):
10
Enter Department Number:
1
Enter the date (YYYY-MM-DD):
2023-01-10
Total Labor Time for Department 1 on 2023-01-10: 21
```

Test 2

```
Enter Department Number:
2
Enter the date (YYYY-MM-DD):
2023-01-11
Total Labor Time for Department 2 on 2023-01-11: 0
```

(we get zero because there is no such date)

Test 3

```
Enter your choice (1-17):
10
Enter Department Number:
2
Enter the date (YYYY-MM-DD):
2023-01-10
Total Labor Time for Department 2 on 2023-01-10: 24
```

6.11 Screenshots showing the testing of query 11

How I populated the relevant tables to allow easy testing

```sql
-- Insert sample data into department table
INSERT INTO department (dept_no, dept_data) VALUES
(1, 'Cutting'),
(2, 'Fitting'),
(3, 'Painting');

-- Insert sample data into process_fit table
INSERT INTO process_fit (process_id, process_data, dept_no, fit_type) VALUES
(101, 'Fitting A1', 2, 'Type F1'),
(102, 'Fitting A2', 2, 'Type F2'),
(103, 'Fitting A3', 2, 'Type F3');

-- Insert sample data into process_paint table
INSERT INTO process_paint (process_id, process_data, dept_no, paint_type, painting_method) VALUES
(201, 'Painting A1', 3, 'Type P1', 'Method P1'),
(202, 'Painting A2', 3, 'Type P2', 'Method P2'),
(203, 'Painting A3', 3, 'Type P3', 'Method P3');

-- Insert sample data into process_cut table
INSERT INTO process_cut (process_id, process_data, dept_no, cutting_type, machine_type) VALUES
(301, 'Cutting A1', 1, 'Type C1', 'Machine C1'),
(302, 'Cutting A2', 1, 'Type C2', 'Machine C2'),
(303, 'Cutting A3', 1, 'Type C3', 'Machine C3');

-- Insert sample data into manufacture table
INSERT INTO manufacture (assembly_id, process_id, job_no) VALUES
(1, 101, 1001),
(1, 201, 1002),
(1, 301, 1003),
(2, 102, 1004),
(2, 202, 1005),
(2, 302, 1006),
(3, 103, 1007),
(3, 203, 1008),
(3, 303, 1009);
```

Test 1

```
Enter your choice (1-17):
11
Enter the assembly ID:
1
Assembly ID: 1, Process ID: 301, Process Data: Cutting A1, Department Number: 1, Department Responsible: Cutting
Assembly ID: 1, Process ID: 101, Process Data: Fitting A1, Department Number: 2, Department Responsible: Fitting
Assembly ID: 1, Process ID: 201, Process Data: Painting A1, Department Number: 3, Department Responsible: Painting
```

Test 2

```
Enter your choice (1-17):
11
Enter the assembly ID:
2
Assembly ID: 2, Process ID: 302, Process Data: Cutting A2, Department Number: 1, Department Responsible: Cutting
Assembly ID: 2, Process ID: 102, Process Data: Fitting A2, Department Number: 2, Department Responsible: Fitting
Assembly ID: 2, Process ID: 202, Process Data: Painting A2, Department Number: 3, Department Responsible: Painting
Options:
```

Test 3

```
Enter your choice (1-17):
11
Enter the assembly ID:
3
Assembly ID: 3, Process ID: 303, Process Data: Cutting A3, Department Number: 1, Department Responsible: Cutting
Assembly ID: 3, Process ID: 103, Process Data: Fitting A3, Department Number: 2, Department Responsible: Fitting
Assembly ID: 3, Process ID: 203, Process Data: Painting A3, Department Number: 3, Department Responsible: Painting
```

6.12 Screenshots showing the testing of query 12

Initial content of my customer table while testing

	name_customer	address_customer	category
1	Alice Smith	123 Main St	1
2	Bob Johnson	456 Oak St	2
3	Charlie Brown	789 Pine St	3
4	Diana Prince	321 Maple St	2
5	Edward Elric	654 Palm St	1
6	Fiona Gallagher	987 Cedar St	3

Test 1

```
Enter your choice (1-17):
12
Enter the start category range:
1
Enter the end category range:
2
Customer Name: Alice Smith, Address: 123 Main St, Category: 1
Customer Name: Bob Johnson, Address: 456 Oak St, Category: 2
Customer Name: Diana Prince, Address: 321 Maple St, Category: 2
Customer Name: Edward Elric, Address: 654 Palm St, Category: 1
Options:
```

Test 2

```
Enter your choice (1-17):
12
Enter the start category range:
2
Enter the end category range:
3
Customer Name: Bob Johnson, Address: 456 Oak St, Category: 2
Customer Name: Charlie Brown, Address: 789 Pine St, Category: 3
Customer Name: Diana Prince, Address: 321 Maple St, Category: 2
Customer Name: Fiona Gallagher, Address: 987 Cedar St, Category: 3
Customer Name: Laury, Address: Address 4, Category: 3
```

Test 3

```
Enter your choice (1-17):
12
Enter the start category range:
1
Enter the end category range:
3
Customer Name: Alice Smith, Address: 123 Main St, Category: 1
Customer Name: Bob Johnson, Address: 456 Oak St, Category: 2
Customer Name: Charlie Brown, Address: 789 Pine St, Category: 3
Customer Name: Diana Prince, Address: 321 Maple St, Category: 2
Customer Name: Edward Elric, Address: 654 Palm St, Category: 1
Customer Name: Fiona Gallagher, Address: 987 Cedar St, Category: 3
Customer Name: Kate, Address: Address 3, Category: 1
Customer Name: Laury, Address: Address 4, Category: 3
Options:
```

6.13 Screenshots showing the testing of query 13

Initial job cut table to test the query

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	1	2023-01-01	2023-01-02	5	Cutter A	1 hour	Steel
2	2	2023-01-02	2023-01-03	6	Cutter B	1.5 hours	Aluminum
3	3	2023-01-03	2023-01-04	4	Cutter C	2 hours	Copper
4	4	2023-01-04	2023-01-05	8	Cutter D	2.5 hours	Titanium
5	5	2023-01-05	2023-01-06	3	Cutter E	3 hours	Carbon Steel
6	6	2023-01-06	2023-01-07	7	Cutter F	3.5 hours	Stainless Steel
7	7	2023-01-07	2023-01-08	9	Cutter G	4 hours	Brass
8	8	2023-01-08	2023-01-09	5	Cutter H	4.5 hours	Bronze
9	9	2023-01-09	2023-01-10	6	Cutter I	5 hours	Iron
10	10	2023-01-10	2023-01-11	8	Cutter J	5.5 hours	Nickel

Test 1

```
Enter your choice (1-17):
13
Enter the start job number:
8
Enter the end job number:
10
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	1	2023-01-01	2023-01-02	5	Cutter A	1 hour	Steel
2	2	2023-01-02	2023-01-03	6	Cutter B	1.5 hours	Aluminum
3	3	2023-01-03	2023-01-04	4	Cutter C	2 hours	Copper
4	4	2023-01-04	2023-01-05	8	Cutter D	2.5 hours	Titanium
5	5	2023-01-05	2023-01-06	3	Cutter E	3 hours	Carbon Steel
6	6	2023-01-06	2023-01-07	7	Cutter F	3.5 hours	Stainless Steel
7	7	2023-01-07	2023-01-08	9	Cutter G	4 hours	Brass

Test 2

```
17. Exit the program
Enter your choice (1-17):
13
Enter the start job number:
1
Enter the end job number:
3
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	4	2023-01-04	2023-01-05	8	Cutter D	2.5 hours	Titanium
2	5	2023-01-05	2023-01-06	3	Cutter E	3 hours	Carbon Steel
3	6	2023-01-06	2023-01-07	7	Cutter F	3.5 hours	Stainless Steel
4	7	2023-01-07	2023-01-08	9	Cutter G	4 hours	Brass

Test 3

```
Enter your choice (1-17):
13
Enter the start job number:
5
Enter the end job number:
6
Cut jobs in range 5 to 6 have been deleted.
```

	job_no	date_start	date_end	labor_time	type_machine_used	time_machine_used	material_used
1	4	2023-01-04	2023-01-05	8	Cutter D	2.5 hours	Titanium
2	7	2023-01-07	2023-01-08	9	Cutter G	4 hours	Brass

6.14 Screenshots showing the testing of query 14

Initial job paint table populated.

	job_no	date_start	date_end	labor_time	color	volume
1	1	2023-01-12	2023-01-13	9	Red	100
2	2	2023-01-13	2023-01-14	11	Blue	120
3	3	2023-01-14	2023-01-15	7	Green	140
4	4	2023-01-15	2023-01-16	5	Yellow	160
5	5	2023-01-16	2023-01-17	8	Black	180
6	6	2023-01-17	2023-01-18	10	White	200
7	7	2023-01-18	2023-01-19	6	Orange	220
8	8	2023-01-19	2023-01-20	12	Purple	240
9	9	2023-01-20	2023-01-21	4	Pink	260
10	10	2023-01-21	2023-01-22	13	Gray	280

Test 1

```
Enter your choice (1-17):
14
Enter the job number for the paint job:
1
Enter the new color for the paint job:
Green
The color of paint job 1 has been changed to Green.
```

	job_no	color
1	1	Green

Test 2

```
17. Exit the program
Enter your choice (1-17):
14
Enter the job number for the paint job:
3
Enter the new color for the paint job:
darkBrown
The color of paint job 3 has been changed to darkBrown.
```

	job_no	color
1	3	darkBrown

Test 3

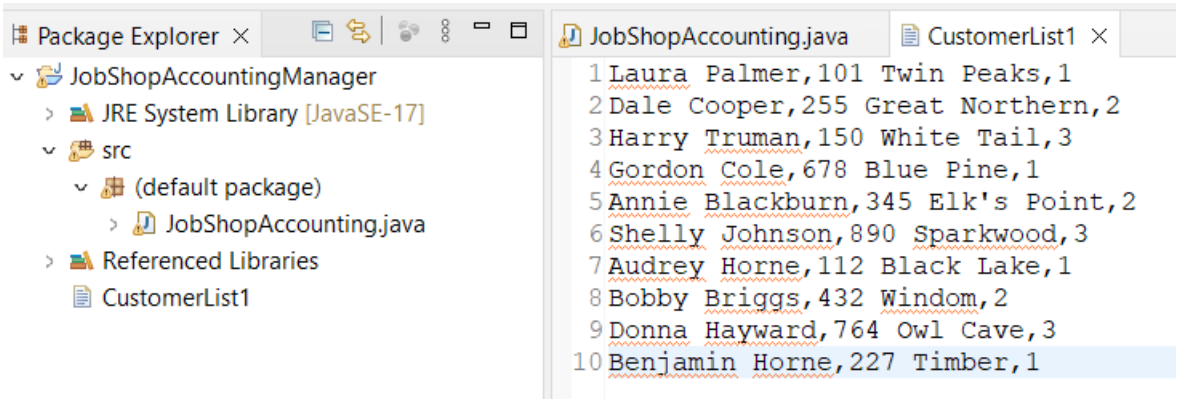
```
Enter your choice (1-17):
14
Enter the job number for the paint job:
4
Enter the new color for the paint job:
DarkPink
The color of paint job 4 has been changed to DarkPink.
```

	job_no	color
1	4	DarkPink

6.15 Screenshots showing the testing of the import and export options

Test 1 – Import

Overview of the text file



Calling query 15

```
Enter your choice (1-17):
15
Enter the input file name:
CustomerList1
All customers from the file have been inserted.
```

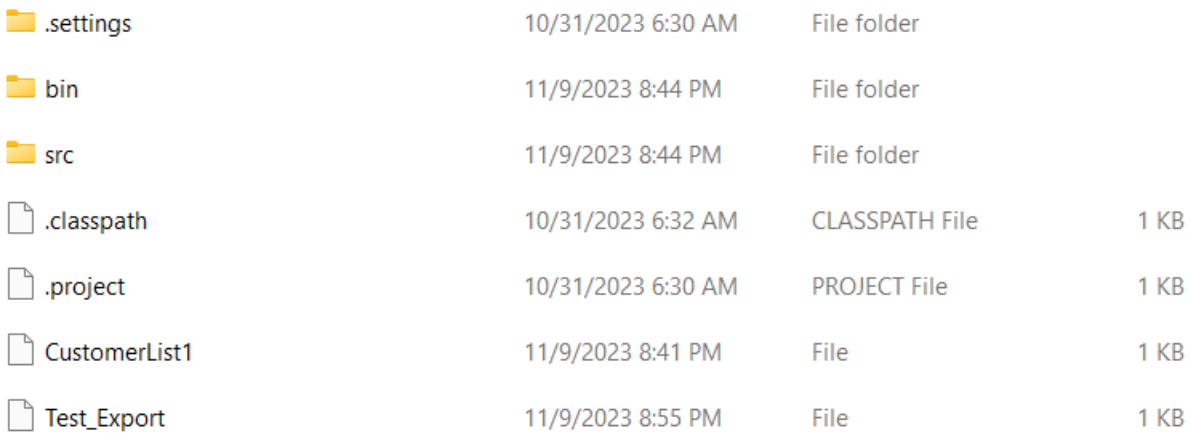
Resulting table

	name_customer	address_customer	category
1	Annie Blackburn	345 Elk's Point	2
2	Audrey Horne	112 Black Lake	1
3	Benjamin Horne	227 Timber	1
4	Bobby Briggs	432 Windom	2
5	Dale Cooper	255 Great Northern	2
6	Donna Hayward	764 Owl Cave	3
7	Gordon Cole	678 Blue Pine	1
8	Harry Truman	150 White Tail	3
9	Laura Palmer	101 Twin Peaks	1
10	Shelly Johnson	890 Sparkwood	3

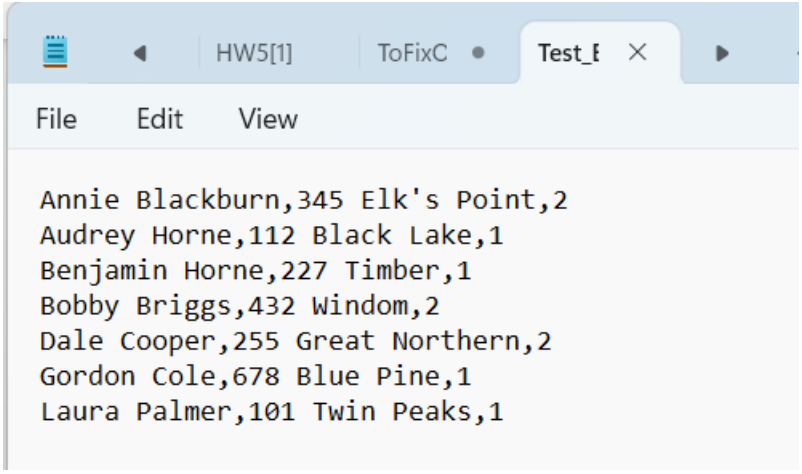
Test 2 – Export

```
Enter your choice (1-17):
16
Enter the start category range:
1
Enter the end category range:
2
Enter the output file name:
Test_Export
Customer data has been written to Test_Export
```

Exported text in my files



Content of the exported file



6.16 Screenshots showing the testing of three types of errors

Test 1

Trying to insert a wrong data type in my customer table

```
Enter your choice (1-17):
1
Enter Customer Name:
Lili
Enter Customer Address:
12Main
Enter Customer Category:
Gold
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at JobShopAccounting.executeQueryForOption1(JobShopAccounting.java:126)
        at JobShopAccounting.main(JobShopAccounting.java:43)
```

Test 2

Trying to insert a department that already exist in my database table

```
17. Exit the program
Enter your choice (1-17):
2
Enter Department Number:
2
Enter Department Data:
Error
com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__departme__DCA63FA6D7AB4E97'. Cannot insert duplicate key in obj
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLServerException.java:259)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerStatement.java:1695)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(SQLServerPreparec
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.doExecute(SQLServerPreparedS
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7675)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerConnection.java:4137)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerStatement.java:272)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServerStatement.java:246)
        at com.microsoft.sqlserver.jdbc@12.4.1.jre11/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.execute(SQLServerPreparedStatement.java:544)
        at JobShopAccounting.insertDepartment(JobShopAccounting.java:163)
        at JobShopAccounting.executeQueryForOption2(JobShopAccounting.java:153)
        at JobShopAccounting.main(JobShopAccounting.java:46)
```

Test 3

Putting the wrong data type in a range search

```
Enter your choice (1-17):
12
Enter the start category range:
T
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at JobShopAccounting.executeQueryForOption12(JobShopAccounting.java:749)
        at JobShopAccounting.main(JobShopAccounting.java:87)
```

6.17 Screenshots showing the testing of the quit option

```
17. Exit the program
Enter your choice (1-17):
17
Exiting the program.
```


```
Enter your choice (1-17):
17
Exiting the program.
```


Task 7 - Web database application and its execution

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

Data handler

```java
package handlers;

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

public class DataHandler {

    private Connection conn;
    private String url = "jdbc:sqlserver://dial0007.database.windows.net:1433;database=cs-dsa-4513-sql-db;user=

    public DataHandler() {
        // Removed the try-catch block so we can let the caller handle any connection issues
        getDBConnection();
    }

    private void getDBConnection() {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver"); // Explicitly load the JDBC driver
            if (conn == null || conn.isClosed()) {
                conn = DriverManager.getConnection(url);
            }
        } catch (ClassNotFoundException | SQLException e) {
            // For the ClassNotFoundException
            System.err.println("JDBC driver not found: " + e.getMessage());
            throw new RuntimeException(e);
        }
    }


    public boolean insertCustomer(String nameCustomer, String addressCustomer, int category) {
        try {
            getDBConnection(); // Ensure the connection is open
            final String sqlQuery = "{call insertCustomer(?, ?, ?)}";
            try (CallableStatement statement = conn.prepareCall(sqlQuery)) {
                statement.setString(1, nameCustomer);
                statement.setString(2, addressCustomer);
                statement.setInt(3, category);

                int rowsAffected = statement.executeUpdate();
                return rowsAffected == 1;
            }
```

```
            }
        } catch (SQLException e) {
            // Log and handle the exception
            System.err.println("SQL exception in insertCustomer: " + e.getMessage());
            return false;
        }
    }

    public ResultSet retrieveCustomerByRange(int categoryStart, int categoryEnd) {
        try {
            getDBConnection(); // Ensure the connection is open
            final String sqlQuery = "{call retrieveCustomerByRange(?, ?)}";
            CallableStatement statement = conn.prepareCall(sqlQuery);
            statement.setInt(1, categoryStart);
            statement.setInt(2, categoryEnd);

            return statement.executeQuery();
        } catch (SQLException e) {
            // Log and rethrow the exception
            System.err.println("SQL exception in retrieveCustomerByRange: " + e.getMessage());
            throw new RuntimeException(e);
        }
    }

    // Make sure to close the connection when done
    public void closeConnection() {
        try {
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        } catch (SQLException e) {
            System.err.println("Error closing database connection: " + e.getMessage());
        }
    }
}
```

JSP file to insert customer

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="handlers.DataHandler"%>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert New Customer</title>
</head>
<body>
    <h2>Add New Customer</h2>
    <form method="post">
        Name: <input type="text" name="name"><br>
        Address: <input type="text" name="address"><br>
        Category: <input type="number" name="category"><br>
        <input type="submit" value="Insert">
    </form>
    <%
        String name = request.getParameter("name");
        String address = request.getParameter("address");
        String categoryStr = request.getParameter("category");

        if (name != null && address != null && categoryStr != null) {
            int category = Integer.parseInt(categoryStr);

            try {
                DataHandler handler = new DataHandler();

                if (handler != null) {
                    boolean success = handler.insertCustomer(name, address, category);
                    handler.closeConnection();

                    if (success) {
                        out.println("<p>Customer inserted successfully!</p>");
                    } else {
                        out.println("<p>Insertion failed.</p>");
                    }
                } else {
                    out.println("<p>DataHandler is null. Unable to establish a connection.</p>");
                }
            } catch (Exception e) {
                out.println("<p>Exception occurred: " + e.getMessage() + "</p>");
            }
```

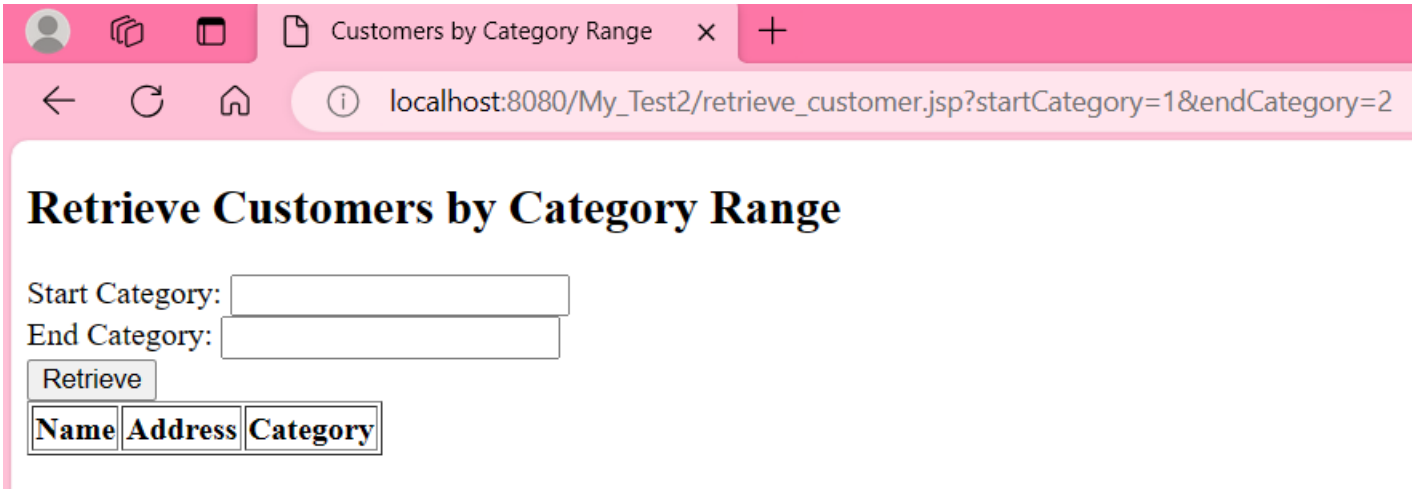
JSP file to retrieve customer

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="handlers.DataHandler"%>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Customers by Category Range</title>
</head>
<body>
    <h2>Retrieve Customers by Category Range</h2>
    <form method="get">
        Start Category: <input type="number" name="startCategory"><br>
        End Category: <input type="number" name="endCategory"><br>
        <input type="submit" value="Retrieve">
    </form>
    <%
        String startCategoryStr = request.getParameter("startCategory");
        String endCategoryStr = request.getParameter("endCategory");

        if(startCategoryStr != null && endCategoryStr != null) {
            int startCategory = Integer.parseInt(startCategoryStr);
            int endCategory = Integer.parseInt(endCategoryStr);
            DataHandler handler = new DataHandler();
            ResultSet rs = handler.retrieveCustomerByRange(startCategory, endCategory);

            out.println("<table border='1'>");
            out.println("<tr><th>Name</th><th>Address</th><th>Category</th></tr>");
            while(rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getString("name_customer") + "</td>");
                out.println("<td>" + rs.getString("address_customer") + "</td>");
                out.println("<td>" + rs.getInt("category") + "</td>");
                out.println("</tr>");
            }
            out.println("</table>");

            rs.close();
            handler.closeConnection();
        }
    %>
</body>
</html>
```

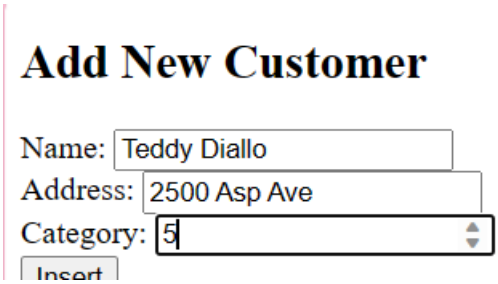
7.2 Screenshots showing the testing of the Web database application

Query 12 – Test 1

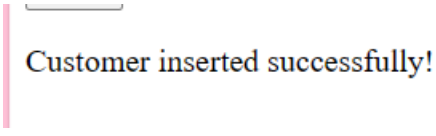
Query 12 – Output 1



Query 1 – Test 1



Query 1 - Output 1



Query 12 – Test 2

Retrieve Customers by Category Range

Start Category: 4
End Category: 5
Retrieve

Query 12 – Output 2

Retrieve

Name	Address	Category
Teddy Diallo	2500 Asp Ave	5

More Tests

Add New Customer

Name: Teddy
Address: okc
Category: 1
Insert

Add New Customer

Name:
Address:
Category:
Insert

Customer inserted successfully!

Using Java and JDBC for Azure S | Setting up Eclipse JSP Web Proje | Insert New Customer

localhost:8080/My_Test2/insert_customer.jsp

Add New Customer

Name: Bob
Address: Austin
Category: 3
Insert

Customer inserted successfully!

Add New Customer

Name: Lili
Address: NYC
Category: 2
Insert

Customer inserted successfully!

Add New Customer

Name: Jack
Address: Dallas
Category: 1
Insert

Customer inserted successfully!

localhost:8080/My_Test2/retrieve_customer.jsp

Retrieve Customers by Category Range

Start Category: 1
End Category: 2
Retrieve

localhost:8080/My_Test2/retrieve_customer.jsp?startCategory=1&endCategory=2

Retrieve Customers by Category Range

Start Category:
End Category:
Retrieve

Name	Address	Category
Jack	Dallas	1
Lili	NYC	2
Teddy	okc	1

Retrieve Customers by Category Range

Start Category: 2
End Category: 3
Retrieve

Name	Address	Category
Bob	Austin	3
Lili	NYC	2

Retrieve Customers by Category Range

Start Category: 1
End Category: 3
Retrieve

Retrieve Customers by Category Range

Start Category:
End Category:
Retrieve

Name	Address	Category
Bob	Austin	3
Jack	Dallas	1
Lili	NYC	2
Teddy	okc	1