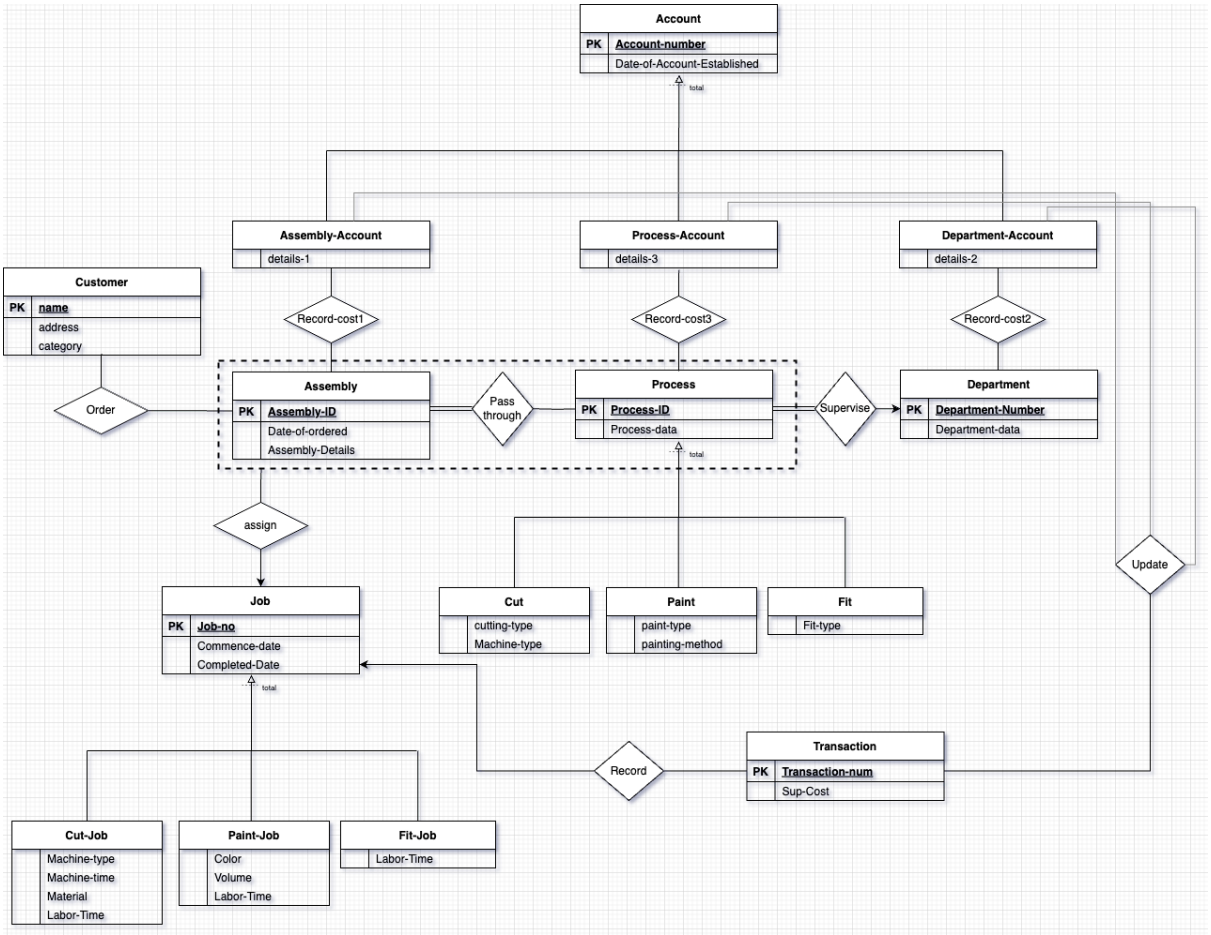
CS/DSA-4513-001-Fall 2023, SECTION: 001, Dr. Le Gruenwald
DATABASE MANAGEMENT

Individual Project: A JOB-SHOP ACCOUNTING SYSTEM

Mohammad Amin Basiri
113606709
ma.basiri@ou.edu

Task 1: ***ER Diagram***



Task 2: ***Relational Database***

- Assembly (Assembly-id, date-of-ordered, Assembly-details)
- Process (Process-Id, Process-data)
- Process-Fit (Process-Id, Fit-type)
- Process-Paint (Process-Id, paint-type, Painting-method)
- Process-Cut (process-Id, Cutting-type, Machine-type)
- Pass-through(Assembly-id, process-Id)
- Customer (name ,address ,category)
- Order (Assembly-id ,name)
- Department (Dep-num, Dep-data)
- Supervise (process-Id, Dep-number)
- job (Job-no, Commence-Date, Completed-Date)
- Cut-job (Job-no, type-of-machine, amount-of-time, material, labor-time)
- Paint-job (Job-no, color, volume, labor-time)
- Fit-job (Job-no, labor-time)
- Transaction (Transaction-num, sup-cost)
- Record (Transaction-num, job-number)
- Assign (Assembly-id, process-Id, job-number)
- Account (Account-number, Established_date)
- Assembly-account (Account-number, details_1)
- Department-account (Account-number, details_2)
- Process-account (Account-number, details_3)
- Update (transaction-num, Account-Number)
- Record-cost1 (Account-number, Assembly-id)
- Record-cost2 (Account-number, dep-number)
- Record-cost3 (Account-number, process-Id)

Task 3: ***Storage Structures***

3.1) Discussion of storage structures for tables

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Assembly	4) insert 9) random search	Assembly-ID	40/day 200/day	Static hashing	Frequent Random search
Process	3) insert		infrequent	Heap files	Insert only
Process-Fit	3) insert		infrequent	Heap files	Insert only
Process-Paint	3) insert		infrequent	Heap files	Insert only
Process-Cut	3) insert		infrequent	Heap files	Insert only
Pass-through	4) insert 11) random search	Assembly_ID	40/day 100/day	Sequential files	Inserting with assembly-ID order
Customer	1) insert 12) range search	category	30/day 100/day	Sequential files and an indexed sequential file on category	In name order, so sequential files And also having frequent range search on category
Order	4) insert		40/day	Heap files	Insert only
Department	2) insert		Infrequent	Heap files	Insert only
Supervise	3) insert		Infrequent	Heap files	Insert only
job	6) insert 7) update 10) random search	Job-no Completed-Date	50/day 20/day	Sequential files and an indexed sequential file on completed date	In Job-no order, so sequential files And also having frequent random search on Completed-Date
Cut-job	6) insert 7) update 10) random search	Job-no Completed-Date	50/day 20/day	Sequential files and an indexed sequential	In Job-no order, so sequential files And also having frequent random

	13)Delete range search	job-no	1/month	file on completed date	search on Completed-Date
Paint-job	6) insert 7) update 10) random search 14) update	 Job-no Completed-Date Job-no	 50/day 20/day 1/week	Sequential files and an indexed sequential file on completed date	In Job-no order, so sequential files And also having frequent random search on Completed-Date
Fit-job	6) insert 7) update 10) random search	 Job-no Completed-Date	 50/day 20/day	Sequential files and an indexed sequential file on completed date	In Job-no order, so sequential files And also having frequent random search on Completed-Date
Transaction	8) insert		50/day	Heap files	Insert only
Record	8) insert		50/day	Heap files	Insert only
Assign	6) insert		50/day	Heap files	Insert only
Account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Assembly-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Department-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Process-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Update	8) insert		50/day	Heap files	Insert only
Record-cost1	5) insert		10/day	Heap files	Insert only
Record-cost2	5) insert		10/day	Heap files	Insert only
Record-cost3	5) insert		10/day	Heap files	Insert only

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

Answer

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Assembly	4) insert 9) random search	Assembly-ID	40/day 200/day	Static hashing	Frequent Random search
Process-Fit	3) insert		infrequent	Heap files	Insert only
Process-Paint	3) insert		infrequent	Heap files	Insert only
Process-Cut	3) insert		infrequent	Heap files	Insert only
Pass-through	4) insert 11) random search	Assembly_ID	40/day 100/day	Sequential files	Inserting with assembly-ID order
Customer	1) insert 12) range search	category	30/day 100/day	Sequential files and an indexed sequential file on category	In name order, so sequential files And also having frequent range search on category
Order	4) insert		40/day	Heap files	Insert only
Department	2) insert		Infrequent	Heap files	Insert only
Supervise	3) insert		Infrequent	Heap files	Insert only
Cut-job	6) insert 7) update 10) range search 13)Delete range search	Job-no Completed-Date job-no	50/day 20/day 1/month	Sequential files and an indexed sequential file on completed date	In name order, so sequential files And also having frequent range search on Completed-Date
Paint-job	6) insert 7) update 10) range search 14) update	Job-no Completed-Date Job-no	50/day 20/day 1/week	Sequential files and an indexed sequential file on completed date	In name order, so sequential files And also having frequent range search on Completed-Date

Fit-job	6) insert 7) update 10) range search	Job-no Completed-Date	50/day 20/day	Sequential files and an indexed sequential file on completed date	In name order, so sequential files And also having frequent range search on Completed-Date
Transaction	8) insert		50/day	Heap files	Insert only
Record	8) insert		50/day	Heap files	Insert only
Assign	6) insert		50/day	Heap files	Insert only
Assembly-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Department-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Process-account	5) insert 8) update	Account-Number	10/day 50/day	Sequential files	Inserting with Account-Number order
Update	8) insert		50/day	Heap files	Insert only
Record-cost1	5) insert		10/day	Heap files	Insert only
Record-cost2	5) insert		10/day	Heap files	Insert only
Record-cost3	5) insert		10/day	Heap files	Insert only

Selecting appropriate storage structures for relational tables is crucial for optimal performance in Azure SQL Database. The key factors to consider are:

Table Size and Growth:

For small tables (less than 100 MB), a clustered index on the primary key columns allows efficient keyed lookups and range scans. The leaf level of the clustered index contains the actual data rows. This works well for fast indexing on smaller tables.

Larger tables benefit from heap storage, which is an unordered structure. Without clustered indexes, heaps avoid overhead of index maintenance as data grows. Appending new data is fast with heaps. However, heaps require secondary nonclustered indexes to allow fast lookups and searches on columns.

If the table is expected to grow significantly over time, a heap storage with nonclustered indexes can offer better scalability. Clustered indexes can get fragmented with large data loads, so heaps are preferred for high volume inserts and updates.

Usage Patterns:

For read-heavy workloads like data warehouses, columnstore indexes provide massive performance gains. Columnstore stores data by column rather than row, allowing fast batch-oriented access, high data compression ratios and efficient aggregation. This makes it ideal for analytics queries. Rowstore storage is better for transactional workloads requiring frequent inserts, updates and deletes.

Indexing Strategies:

Effective indexing is crucial for fast data access. Smaller tables can rely solely on the clustered index on primary keys and avoid over-indexing. Larger tables require careful indexing based on query patterns. Seek keys should be added on columns used for filtering, sorting, and join operations. Analyze and tune indexes over time based on usage.

The optimal index types like B-tree, columnstore or hash indexes depend on the usage - whether point lookups, range scans or aggregations are more common.

Partitioning:

Partitioning divides larger tables across multiple physical storage units transparently. It allows queries to focus only on relevant partitions. Partitioning keys should be chosen based on filter predicates in query patterns. Partition pruning dramatically improves query performance.

Right-sized partitions should be created to optimize management overhead. Too many partitions can impact performance. Partitions can be merged, split or moved between storage as needed.

Compression:

Data compression like row, page or columnstore compression can significantly reduce storage requirements and lower I/O costs. However, it increases CPU usage for compression/decompression. Highly analytical workloads gain greatly from columnstore compression. Transactional workloads may prefer limited or no compression.

Testing different storage structures to determine the optimal balance is key. Continuously analyzing usage patterns and fine-tuning the structures maximizes the performance benefits.

Reference

1. Microsoft Docs. "Table data storage in Azure SQL Database - Azure SQL Database | Microsoft Docs." https://docs.microsoft.com/en-us/azure/azure-sql/database/table-storage
2. Solomon Rutzky, "A SQL DBA's Guide to Azure SQL Database Storage", Brent Ozar Unlimited, August 15, 2022.

https://www.brentozar.com/archive/2022/08/a-sql-dbas-guide-to-azure-sql-database-storage/

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

The following is all the SQL statements:

```
-- Drop tables if they exist

DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Supervise;
DROP TABLE IF EXISTS Record;
DROP TABLE IF EXISTS Assign;
DROP TABLE IF EXISTS Updates;
DROP TABLE IF EXISTS Record_cost1;
DROP TABLE IF EXISTS Record_cost2;
DROP TABLE IF EXISTS Record_cost3;
DROP TABLE IF EXISTS Pass_through;
DROP TABLE IF EXISTS Cut_job;
DROP TABLE IF EXISTS Paint_job;
DROP TABLE IF EXISTS Fit_job;
DROP TABLE IF EXISTS job;
DROP TABLE IF EXISTS Process_Fit;
DROP TABLE IF EXISTS Process_Paint;
DROP TABLE IF EXISTS Process_Cut;
DROP TABLE IF EXISTS Process;
DROP TABLE IF EXISTS Transactions;
DROP TABLE IF EXISTS Assemblys;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Department;
DROP TABLE IF EXISTS Assembly_account;
DROP TABLE IF EXISTS Department_account;
DROP TABLE IF EXISTS Process_account;
DROP TABLE IF EXISTS account;


-- Assembly table
CREATE TABLE Assemblys (
   Assembly_id INT PRIMARY KEY,
   Date_of_ordered VARCHAR(100),
   Assembly_details VARCHAR(255)
);
-- Process table
CREATE TABLE Process (
   Process_Id INT PRIMARY KEY,
```

```
    Process_data VARCHAR(255)
);

-- Process-Fit table
CREATE TABLE Process_Fit (
    Process_Id INT PRIMARY KEY,
    Fit_type VARCHAR(50),
    FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id)
);

-- Process-Paint table
CREATE TABLE Process_Paint (
    Process_Id INT PRIMARY KEY,
    Paint_type VARCHAR(50),
    Painting_method VARCHAR(50),
    FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id)
);

-- Process-Cut table
CREATE TABLE Process_Cut (
    Process_Id INT PRIMARY KEY,
    Cutting_type VARCHAR(50),
    Machine_type VARCHAR(50),
    FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id)
);

-- Pass-through table
CREATE TABLE Pass_through (
    Assembly_id INT,
    Process_Id INT,
    PRIMARY KEY (Assembly_id, Process_Id),
    FOREIGN KEY (Assembly_id) REFERENCES Assemblys(Assembly_id),
    FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id)
);

-- Customer table
CREATE TABLE Customer (
    Names VARCHAR(20) PRIMARY KEY,
    Addresses VARCHAR(20),
    Category INT
    CONSTRAINT CHK_Category CHECK (Category BETWEEN 1 AND 10),
);

-- Creating index for category in Customer table
CREATE INDEX idx_Customer_Category ON Customer(Category);
```

```sql
-- Order table
CREATE TABLE Orders (
   Assembly_id INT,
   Names VARCHAR(20),
   PRIMARY KEY (Assembly_id, Names),
   FOREIGN KEY (Assembly_id) REFERENCES Assemblys(Assembly_id),
   FOREIGN KEY (Names) REFERENCES Customer(Names)
);



-- Department table
CREATE TABLE Department (
   Dep_num INT PRIMARY KEY,
   Dep_data VARCHAR(255)
);

-- Supervise table
CREATE TABLE Supervise (
   Process_Id INT,
   Dep_number INT,
   PRIMARY KEY (Process_Id),
   FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id),
   FOREIGN KEY (Dep_number) REFERENCES Department(Dep_num)
);



-- job table
CREATE TABLE job (
   Job_no INT PRIMARY KEY,
   Commence_Date DATE,
   Completed_Date DATE
);

-- Creating index for completed date in job table
CREATE INDEX idx_Job_CompletedDate ON job(Completed_Date);

-- Cut-job table
CREATE TABLE Cut_job (
   Job_no INT PRIMARY KEY,
   type_of_machine VARCHAR(50),
   amount_of_time INT,
```

```
    material VARCHAR(100),
    labor_time INT,
    FOREIGN KEY (Job_no) REFERENCES job(Job_no)
);

-- Paint-job table
CREATE TABLE Paint_job (
    Job_no INT PRIMARY KEY,
    color VARCHAR(50),
    volume INT,
    labor_time INT,
    FOREIGN KEY (Job_no) REFERENCES job(Job_no)
);

-- Fit-job table
CREATE TABLE Fit_job (
    Job_no INT PRIMARY KEY,
    labor_time INT,
    FOREIGN KEY (Job_no) REFERENCES job(Job_no)
);




-- Transaction table
CREATE TABLE Transactions (
    Transaction_num INT PRIMARY KEY,
    Sup_cost DECIMAL(10, 2)
);

-- Record table
CREATE TABLE Record (
    Transaction_num INT,
    job_number INT,
    PRIMARY KEY (Transaction_num, job_number),
    FOREIGN KEY (Transaction_num) REFERENCES Transactions(Transaction_num),
    FOREIGN KEY (job_number) REFERENCES job(Job_no)
);

-- Assign table
CREATE TABLE Assign (
    Assembly_id INT,
    Process_Id INT,
    job_number INT,
    PRIMARY KEY (Assembly_id, Process_Id, job_number),
```

```
    FOREIGN KEY (Assembly_id, Process_Id) REFERENCES Pass_through(Assembly_id,
Process_Id),
    FOREIGN KEY (job_number) REFERENCES job(Job_no)
);


-- Account table
CREATE TABLE Account (
    Account_number INT PRIMARY KEY,
    Established_date DATE
);

-- Assembly-account table
CREATE TABLE Assembly_account (
    Account_number INT,
    details_1 DECIMAL(10, 2),
    PRIMARY KEY (Account_number),
    FOREIGN KEY (Account_number) REFERENCES Account(Account_number)
);

-- Department-account table
CREATE TABLE Department_account (
    Account_number INT,
    details_2 DECIMAL(10, 2),
    PRIMARY KEY (Account_number),
    FOREIGN KEY (Account_number) REFERENCES Account(Account_number)
);

-- Process-account table
CREATE TABLE Process_account (
    Account_number INT,
    details_3 DECIMAL(10, 2),
    PRIMARY KEY (Account_number),
    FOREIGN KEY (Account_number) REFERENCES Account(Account_number)
);

-- Update table
CREATE TABLE Updates (
    Transaction_num INT,
    Account_Number INT,
    PRIMARY KEY (Transaction_num, Account_Number),
    FOREIGN KEY (Transaction_num) REFERENCES Transactions(Transaction_num),
    FOREIGN KEY (Account_Number) REFERENCES Account(Account_number)
);
```

```sql
-- Record-cost1 table
CREATE TABLE Record_cost1 (
   Account_number INT,
   Assembly_id INT,
   PRIMARY KEY (Account_number, Assembly_id),
   FOREIGN KEY (Account_number) REFERENCES Assembly_account(Account_number),
   FOREIGN KEY (Assembly_id) REFERENCES Assemblys(Assembly_id)
);

-- Record-cost2 table
CREATE TABLE Record_cost2 (
   Account_number INT,
   Dep_number INT,
   PRIMARY KEY (Account_number, Dep_number),
   FOREIGN KEY (Account_number) REFERENCES Department_account(Account_number),
   FOREIGN KEY (Dep_number) REFERENCES Department(Dep_num)
);

-- Record-cost3 table
CREATE TABLE Record_cost3 (
   Account_number INT,
   Process_Id INT,
   PRIMARY KEY (Account_number, Process_Id),
   FOREIGN KEY (Account_number) REFERENCES Process_account(Account_number),
   FOREIGN KEY (Process_Id) REFERENCES Process(Process_Id)
);
```

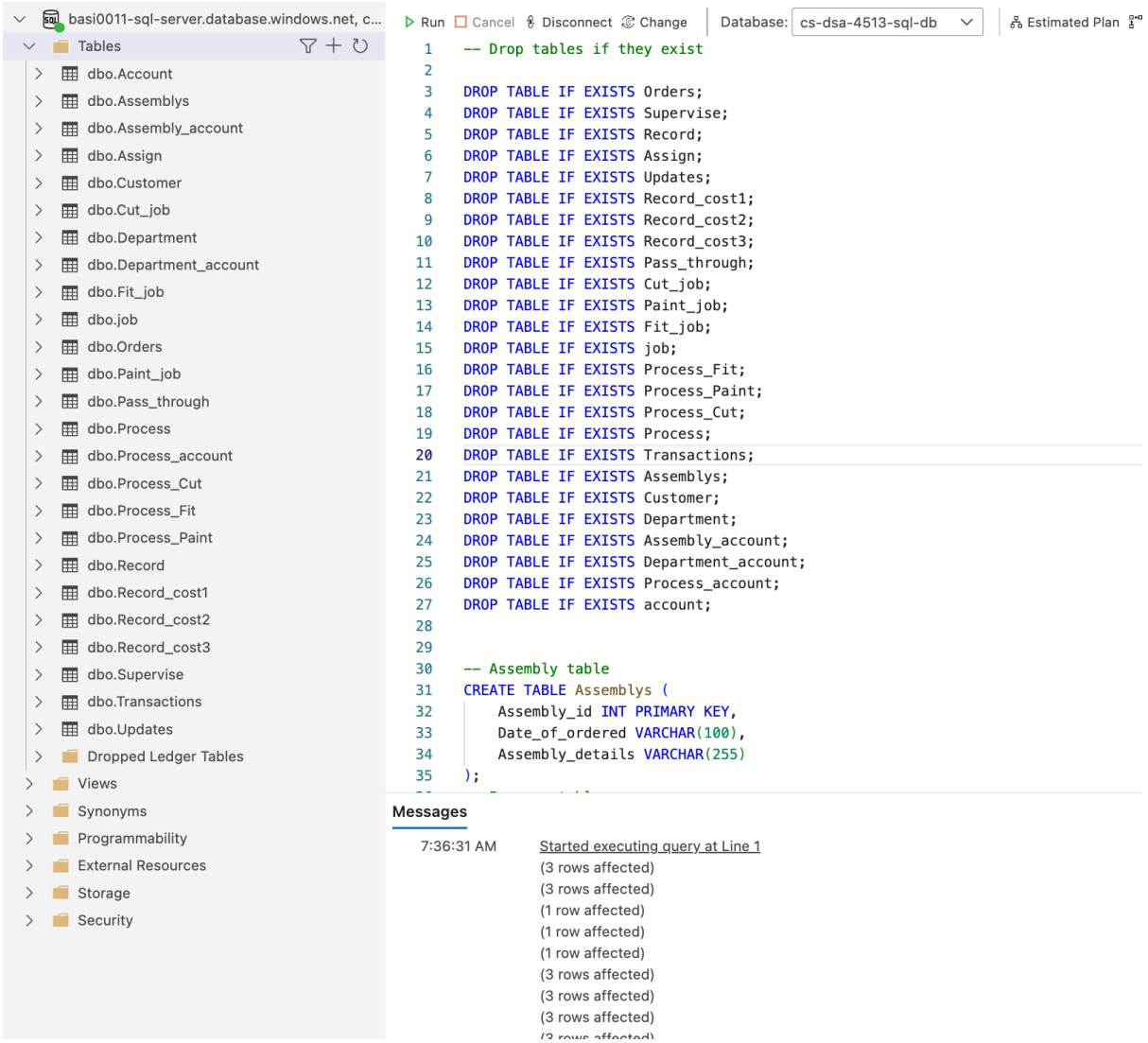
Inserting only for testing then I will delete those.

```
-- Sample data for Assemblys table
INSERT INTO Assemblys (Assembly_id, Date_of_ordered, Assembly_details) VALUES
(1, '2023-11-01', 'Sample assembly details 1'),
(2, '2023-11-02', 'Sample assembly details 2'),
(3, '2023-11-03', 'Sample assembly details 3');

-- Sample data for Process table
INSERT INTO Process (Process_Id, Process_data) VALUES
(1, 'process 1'),
(2, 'process 2'),
(3, 'process 3');

-- Sample data for Process_Fit table
INSERT INTO Process_Fit (Process_Id, Fit_type) VALUES
(1, 'Type A');

-- Sample data for Process_Paint table
INSERT INTO Process_Paint (Process_Id, Paint_type, Painting_method) VALUES
(2, 'Type X', 'Method A');

-- Sample data for Process_Cut table
INSERT INTO Process_Cut (Process_Id, Cutting_type, Machine_type) VALUES
(3, 'Cut Type 1', 'Machine X');

-- Sample data for Pass_through table
INSERT INTO Pass_through (Assembly_id, Process_Id) VALUES
(1, 1),
(2, 2),
(3, 3);

-- Sample data for Customer table
INSERT INTO Customer (Names, Addresses, Category) VALUES
('Customer 1', 'Address 1', 5),
('Customer 2', 'Address 2', 1),
('Customer 3', 'Address 3', 3);

-- Sample data for Orders table
INSERT INTO Orders (Assembly_id, Names) VALUES
(1, 'Customer 1'),
(2, 'Customer 2'),
(3, 'Customer 3');

-- Sample data for Department table
INSERT INTO Department (Dep_num, Dep_data) VALUES
```

```
(1, 'Department Data 1'),
(2, 'Department Data 2'),
(3, 'Department Data 3');

-- Sample data for Supervise table
INSERT INTO Supervise (Process_Id, Dep_number) VALUES
(1, 1),
(2, 2),
(3, 3);

-- Sample data for job table
INSERT INTO job (Job_no, Commence_Date, Completed_Date) VALUES
(1, '2023-11-01', '2023-11-05'),
(2, '2023-11-02', '2023-11-06'),
(3, '2023-11-03', '2023-11-07');

-- Sample data for Cut_job table
INSERT INTO Cut_job (Job_no, Type_of_machine, Amount_of_time, Material, Labor_time)
VALUES
(1, 'Machine A', 10, 'Material A', 40);

-- Sample data for Paint_job table
INSERT INTO Paint_job (Job_no, Color, Volume, Labor_time) VALUES
(2, 'Blue', 150, 50);

-- Sample data for Fit_job table
INSERT INTO Fit_job (Job_no, Labor_time) VALUES
(3, 60);

-- Sample data for Transactions table
INSERT INTO Transactions (Transaction_num, Sup_cost) VALUES
(1, 100.00),
(2, 150.00),
(3, 200.00);

-- Sample data for Record table
INSERT INTO Record (Transaction_num, Job_number) VALUES
(1, 1),
(2, 2),
(3, 3);

-- Sample data for Assign table
INSERT INTO Assign (Assembly_id, Process_Id, Job_number) VALUES
(1, 1, 1),
(2, 2, 2),
```

```
(3, 3, 3);

-- Sample data for account table
INSERT INTO Account (Account_number, Established_date) VALUES
(1, '2023-11-01'),
(2, '2023-11-02'),
(3, '2023-11-03');

-- Sample data for Assembly_account table
INSERT INTO Assembly_account (Account_number, Details_1) VALUES
(1, 100.00);

-- Sample data for Department_account table
INSERT INTO Department_account (Account_number, Details_2) VALUES
(2, 120.00);

-- Sample data for Process_account table
INSERT INTO Process_account (Account_number, Details_3) VALUES
(3, 130.00);

-- Sample data for Updates table
INSERT INTO Updates (Transaction_num, Account_Number) VALUES
(1, 1),
(2, 2),
(3, 3);

-- Sample data for Record_cost1 table
INSERT INTO Record_cost1 (Account_number, Assembly_id) VALUES
(1, 1);

-- Sample data for Record_cost2 table
INSERT INTO Record_cost2 (Account_number, Dep_number) VALUES
(2, 2);

-- Sample data for Record_cost3 table
INSERT INTO Record_cost3 (Account_number, Process_Id) VALUES
(3, 3);
```

The creation of the tables:



Inserting only for testing then I will delete those. Viewing the tables after simple insertion

```
505  ------------------------------------------------------------------------------------------------------------------------
506
507
508  SELECT * FROM Customer;
509  SELECT * FROM Department;
510  SELECT * FROM Process;
511      SELECT * FROM Process_Cut;
512      SELECT * FROM Process_Paint;
513      SELECT * FROM Process_Fit;
514  SELECT * FROM Supervise;
515  SELECT * FROM Assemblys;
516  SELECT * FROM Pass_through;
517  SELECT * FROM Orders;
518  SELECT * FROM Account;
519      SELECT * FROM Process_account;
520      SELECT * FROM Record_cost3;
521      SELECT * FROM Assembly_account;
522      SELECT * FROM Record_cost1;
523      SELECT * FROM Department_account;
524      SELECT * FROM Record_cost2;
525  SELECT * FROM job;
526      SELECT * FROM Cut_job;
527      SELECT * FROM Fit_job;
528      SELECT * FROM Paint_job;
529  SELECT * FROM Record;
530  SELECT * FROM Assign;
531  SELECT * FROM Transactions;
532  SELECT * FROM Updates;
533
```

Results | Messages

	Names	Addresses	Category
1	Customer 1	Address 1	5
2	Customer 2	Address 2	1
3	Customer 3	Address 3	3

	Dep_num	Dep_data
1	1	Department Data 1
2	2	Department Data 2
3	3	Department Data 3

Results grid

	Process_Id	Process_data
1	1	process 1
2	2	process 2
3	3	process 3

	Process_Id	Cutting_type	Machine_type
1	3	Cut Type 1	Machine X

	Process_Id	Paint_type	Painting_method
1	2	Type X	Method A

Results grid

	Process_Id	Fit_type
1	1	Type A

	Process_Id	Dep_number
1	1	1
2	2	2
3	3	3

	Assembly_id	Date_of_ordered	Assembly_details
1	1	2023-11-01	Sample assembly details 1
2	2	2023-11-02	Sample assembly details 2
3	3	2023-11-03	Sample assembly details 3

	Assembly_id	Process_Id
1	1	1
2	2	2
3	3	3

	Assembly_id	Names
1	1	Customer 1
2	2	Customer 2
3	3	Customer 3

	Account_number	Established_date
1	1	2023-11-01
2	2	2023-11-02
3	3	2023-11-03

	Account_number	details_3
1	3	130.00

	Account_number	Process_Id
1	3	3

	Account_number	details_1
1	1	100.00

	Account_number	Assembly_id
1	1	1

	Account_number	details_2
1	2	120.00

	Account_number	Dep_number
1	2	2

	Job_no	Commence_Date	Completed_Date
1	1	2023-11-01	2023-11-05
2	2	2023-11-02	2023-11-06
3	3	2023-11-03	2023-11-07

	Job_no	type_of_machine	amount_of_time	material	labor_time
1	1	Machine A	10	Material A	40

	Job_no	labor_time
1	3	60

	Job_no	color	volume	labor_time
1	2	Blue	150	50

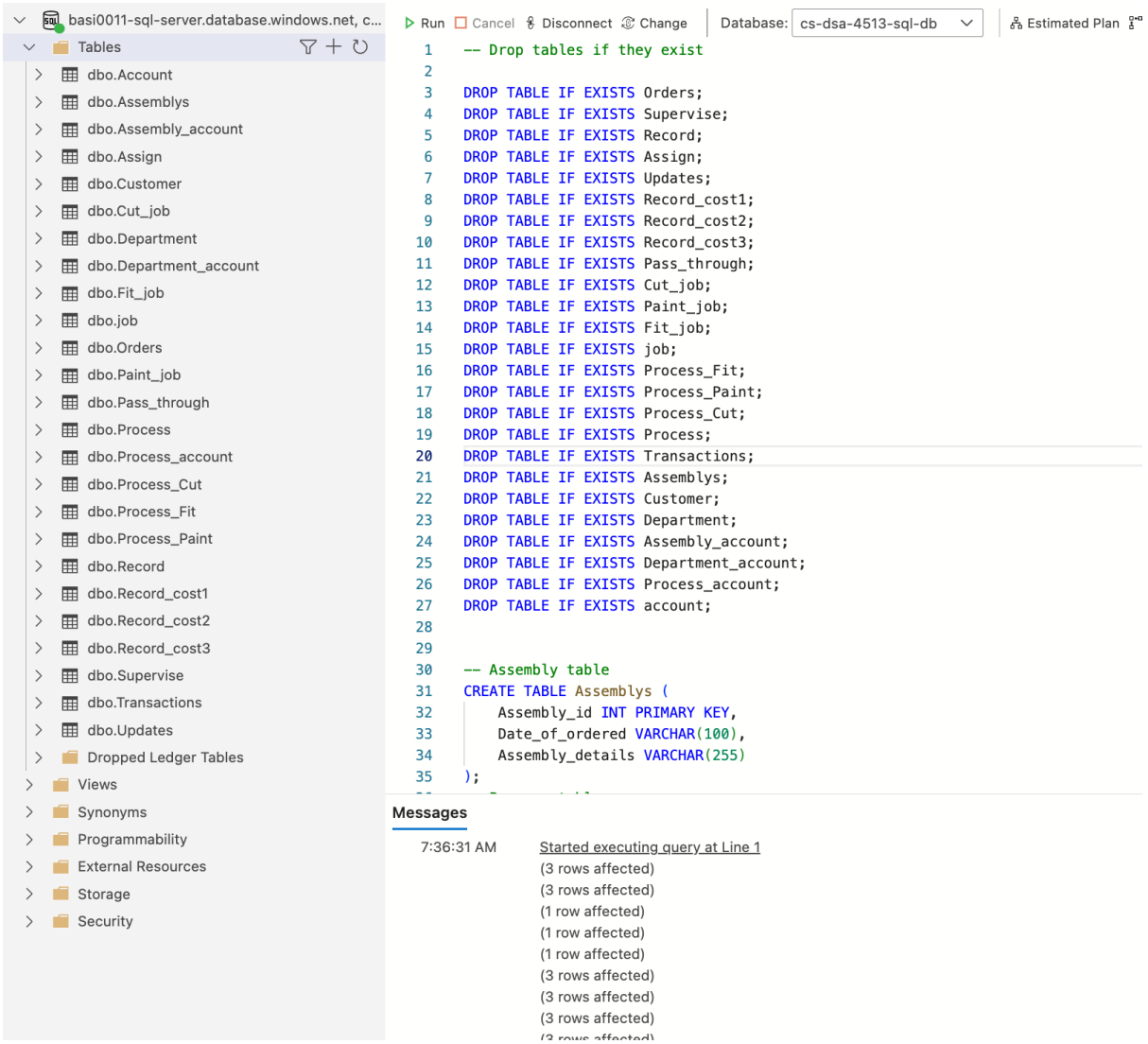
	Transaction_num	job_number
1	1	1
2	2	2
3	3	3

	Assembly_id	Process_Id	job_number
1	1	1	1
2	2	2	2
3	3	3	3

	Transaction_num	Sup_cost
1	1	100.00
2	2	150.00
3	3	200.00

	Transaction_num	Account_Number
1	1	1
2	2	2
3	3	3

The creation of the tables:



```
19    DROP TABLE IF EXISTS Process;
20    DROP TABLE IF EXISTS Transactions;
```

Messages

```
8:08:57 AM    Started executing query at Line 1
              (3 rows affected)
              (3 rows affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (3 rows affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
              (3 rows affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
              Total execution time: 00:00:00.931
```

Task 5: ***Codes***

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

For All the queries I used transact SQL. so whenever in Java I want to use the SQLs, I will easily call the procedure.

```
1   ------------------------------------------------------------------------------------------
2   ---Procedure 01 Enter a new customer
3   DROP PROCEDURE IF EXISTS option_1;
4   GO
5   CREATE PROCEDURE option_1
6       @Names VARCHAR(20),
7       @Addresses VARCHAR(20),
8       @Category INT
9   AS
10  BEGIN
11  INSERT INTO Customer (Names, Addresses, Category) VALUES (@Names, @Addresses, @Category); -- inserting the values
12  END
13  --Executing Procedure1
14  GO
15  EXEC option_1 @Names = 'Marshall', @Addresses = 'Houston Ave', @Category = 3;
16  ------------------------------------------------------------------------------------------
17
18   ------------------------------------------------------------------------------------------
19   ---Procedure 02 Enter a new department
20   DROP PROCEDURE IF EXISTS option_2;
21   GO
22
23   CREATE PROCEDURE option_2
24       @Dep_num INT,
25       @Dep_data VARCHAR(255)
26   AS
27   BEGIN
28       INSERT INTO Department (Dep_num, Dep_data) VALUES (@Dep_num, @Dep_data); -- inserting the values
29   END
30   GO
31
32   --Executing Procedure2
33   GO
34   EXEC option_2 @Dep_num = 13, @Dep_data = 'New Department';
35   ------------------------------------------------------------------------------------------
36
```

```sql
-------------------------------------------------------------------------------------
---Procedure 03 Enter new process-id and its department
DROP PROCEDURE IF EXISTS option_3;
GO

CREATE PROCEDURE option_3
    @CutPaintFit INT,
    @Process_Id INT,
    @Dep_number INT,
    @Process_data VARCHAR(255),
    @Type VARCHAR(50),
    @Information VARCHAR(50)
AS
BEGIN
    INSERT INTO Process (Process_Id, Process_data) VALUES (@Process_Id, @Process_data); -- inserting the values
    IF @CutPaintFit = 1 --if it is cut
        INSERT INTO Process_Cut (Process_Id, Cutting_type, Machine_type) VALUES (@Process_Id, @Type, @Information); -- inserting the values
    IF @CutPaintFit = 2 --if it is paint
        INSERT INTO Process_Paint (Process_Id, Paint_type, Painting_method) VALUES (@Process_Id, @Type, @Information); -- inserting the values
    IF @CutPaintFit = 3 --if it is fit
        INSERT INTO Process_Fit (Process_Id, Fit_type) VALUES (@Process_Id, @Type); -- inserting the values
    INSERT INTO Supervise (Process_Id, Dep_number) VALUES (@Process_Id, @Dep_number);
END
GO

--Executing Procedure3
GO
EXEC option_3 @CutPaintFit = 2, @Process_Id = 4, @Process_data = 'data process a', @Dep_number = 2, @Type = 'Type A', @Information = 'Information';
-------------------------------------------------------------------------------------------------------
```

```sql
-------------------------------------------------------------------------------------------------------
--Procedure 04 Enter new assembly with processes
DROP PROCEDURE IF EXISTS option_4;
GO

CREATE PROCEDURE option_4
    @Assembly_id INT,
    @Customer_name VARCHAR(100),
    @Assembly_details VARCHAR(255),
    @Date_ordered VARCHAR(100),
    @Process_Ids VARCHAR(MAX) -- Allowing user to input Process Ids separated by commas
AS
BEGIN
    -- Inserting into Assembly table
    INSERT INTO Assemblys (Assembly_id, date_of_ordered, Assembly_details)
    VALUES (@Assembly_id, @Date_ordered, @Assembly_details);

    -- Splitting the process IDs string and inserting into Pass_through table
    DECLARE @ProcessId INT;
    DECLARE processIdsCursor CURSOR FOR
    SELECT value FROM STRING_SPLIT(@Process_Ids, ',');

    OPEN processIdsCursor;
    FETCH NEXT FROM processIdsCursor INTO @ProcessId;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Assigning each process to the assembly
        INSERT INTO Pass_through (Assembly_id, Process_Id)
        VALUES (@Assembly_id, @ProcessId);

        FETCH NEXT FROM processIdsCursor INTO @ProcessId;
    END

    CLOSE processIdsCursor;
    DEALLOCATE processIdsCursor;

    -- Inserting into Order table associating the assembly with the customer
    INSERT INTO Orders (Assembly_id, Names)
    VALUES (@Assembly_id, @Customer_name);
END
GO

--Executing Procedure4
GO
EXEC option_4 @Assembly_id = 111, @Customer_name = 'Marshall', @Assembly_details = 'Assembly Description', @Date_ordered = '2023-11-15', @Process_Ids = '3';

-- Enter a new assembly with its customer-name, assembly-details, assembly-id, date-ordered, and associate it with a specific process by providing its Process_Id.
-------------------------------------------------------------------------------------------------------
```

```
119  ------------------------------------------------------------------------------------------------------------------------
120  --Procedure: Create new account and associate with an entity
121  DROP PROCEDURE IF EXISTS option_5;
122  GO
123
124  CREATE PROCEDURE option_5
125      @Account_number INT,
126      @Established_date DATE,
127      @Details DECIMAL(10,2),
128      @Entity_type VARCHAR(20), -- 'process', 'assembly', or 'department'
129      @Entity_id INT -- Process_Id, Assembly_id, or Dep_num based on Entity_type
130  AS
131  BEGIN
132      -- Inserting into Account table
133      INSERT INTO Account (Account_number, Established_date)
134      VALUES (@Account_number, @Established_date);
135
136      IF @Entity_type = 'process'
137      BEGIN
138          -- Associate account with a process
139          INSERT INTO Process_account (Account_number, details_3)
140          VALUES (@Account_number, @Details);
141          INSERT INTO Record_cost3 (Account_number, Process_Id)
142          VALUES (@Account_number, @Entity_id);
143      END
144      ELSE IF @Entity_type = 'assembly'
145      BEGIN
146          -- Associate account with an assembly
147          INSERT INTO Assembly_account (Account_number, details_1)
148          VALUES (@Account_number, @Details);
149          INSERT INTO Record_cost1 (Account_number, Assembly_id)
150          VALUES (@Account_number, @Entity_id);
151      END
152      ELSE IF @Entity_type = 'department'
153      BEGIN
154          -- Associate account with a department
155          INSERT INTO Department_account (Account_number, details_2)
156          VALUES (@Account_number, @Details);
157          INSERT INTO Record_cost2 (Account_number, Dep_number)
158          VALUES (@Account_number, @Entity_id);
159      END
160  END
161  GO
162
163  --Executing the procedure: Create an account and associate it with the relevant entity
164  GO
165  EXEC option_5 @Account_number = 101, @Established_date = '2023-11-15', @Details = 999.00, @Entity_type = 'process', @Entity_id = 1;
166
167  -- Create a new account and associate it with a specific entity (process, assembly, or department) by providing entity's ID and type.
168  ------------------------------------------------------------------------------------------------------------------------------------------------------------


'0  ------------------------------------------------------------------------------------------------------------------------
'1  -- Procedure for Option 6: Enter a new job
'2  DROP PROCEDURE IF EXISTS option_6;
'3  GO
'4
'5  CREATE PROCEDURE option_6
'6      @Job_no INT,
'7      @Assembly_id INT,
'8      @Process_Id INT,
'9      @Commence_Date DATE
!0  AS
!1  BEGIN
!2      -- Inserting into job table
!3      INSERT INTO job (Job_no, Commence_Date)
!4      VALUES (@Job_no, @Commence_Date);
!5
!6      -- Inserting into Assign table to associate job with assembly and process
!7      INSERT INTO Assign (Assembly_id, Process_Id, Job_number)
!8      VALUES (@Assembly_id, @Process_Id, @Job_no);
!9  END
!0  GO
!1
!2
!3  --Executing the procedure: Create an account and associate it with the relevant entity
!4  GO
!5  EXEC option_6 @Job_no = 101, @Assembly_id = 2, @Process_Id = 2, @Commence_Date = '2020-01-05';
!6  ------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```sql
-- Procedure for Option 7: Complete a job
DROP PROCEDURE IF EXISTS option_7;
GO

CREATE PROCEDURE option_7
    @Job_no INT,
    @Completed_Date DATE,
    @Job_type VARCHAR(20),
    @labor_time INT,
    @Job_details1 VARCHAR(50),
    @Job_details2 VARCHAR(50),
    @Job_details3 INT
AS
BEGIN
    -- Updating the Completed_Date in job table
    UPDATE job
    SET Completed_Date = @Completed_Date
    WHERE Job_no = @Job_no;

    -- Inserting job details based on job type
    IF @Job_type = 'cut'
    BEGIN
        INSERT INTO Cut_job (Job_no, type_of_machine, amount_of_time, material, labor_time)
        VALUES (@Job_no, @Job_details1, @Job_details3, @Job_details2, @labor_time);
    END
    ELSE IF @Job_type = 'paint'
    BEGIN
        INSERT INTO Paint_job (Job_no, color, volume, labor_time)
        VALUES (@Job_no, @Job_details1, @Job_details3, @labor_time);
    END
    ELSE IF @Job_type = 'fit'
    BEGIN
        INSERT INTO Fit_job (Job_no, labor_time)
        VALUES (@Job_no, @labor_time);
    END
END
GO



--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_7 @Job_no = 101, @Completed_Date = '2020-04-04', @Job_type = 'paint', @labor_time = 5, @Job_details1 = 'details1', @Job_details2 = 'details2', @Job_details3 = 78 ;
```

```sql
-- Procedure 08: Enter a transaction-no and its sup-cost and update all the costs (detail-1, detail-2, detail-3) of the affected accounts by adding sup-cost to their current values of detail
DROP PROCEDURE IF EXISTS option_8;
GO

CREATE PROCEDURE option_8
    @TransactionNo INT,
    @SupCost DECIMAL(10, 2),
    @JobNo INT,
    @AccountIdsAssembly VARCHAR(MAX),
    @AccountIdsDepartment VARCHAR(MAX),
    @AccountIdsProcess VARCHAR(MAX)
AS
BEGIN
    -- Inserting into Transactions table
    INSERT INTO Transactions (Transaction_num, Sup_cost)
    VALUES (@TransactionNo, @SupCost);

    -- Inserting into Record table
    INSERT INTO Record (Transaction_num, Job_number)
    VALUES (@TransactionNo, @JobNo);

    -- Inserting into Updates table for Assembly accounts
    INSERT INTO Updates (Transaction_num, Account_Number)
    SELECT @TransactionNo, Account_number FROM Assembly_account WHERE Account_number IN (SELECT value FROM STRING_SPLIT(@AccountIdsAssembly, ','));

    -- Inserting into Updates table for Department accounts
    INSERT INTO Updates (Transaction_num, Account_Number)
    SELECT @TransactionNo, Account_number FROM Department_account WHERE Account_number IN (SELECT value FROM STRING_SPLIT(@AccountIdsDepartment, ','));

    -- Inserting into Updates table for Process accounts
    INSERT INTO Updates (Transaction_num, Account_Number)
    SELECT @TransactionNo, Account_number FROM Process_account WHERE Account_number IN (SELECT value FROM STRING_SPLIT(@AccountIdsProcess, ','));

    -- Updating detail-1 in Assembly_account
    UPDATE a
    SET a.details_1 = a.details_1 + @SupCost
    FROM Assembly_account a
    INNER JOIN Updates u ON a.Account_number = u.Account_Number AND u.Transaction_num = @TransactionNo;

    -- Updating detail-2 in Department_account
    UPDATE d
    SET d.details_2 = d.details_2 + @SupCost
    FROM Department_account d
    INNER JOIN Updates u ON d.Account_number = u.Account_Number AND u.Transaction_num = @TransactionNo;

    -- Updating detail-3 in Process_account
    UPDATE p
    SET p.details_3 = p.details_3 + @SupCost
    FROM Process_account p
    INNER JOIN Updates u ON p.Account_number = u.Account_Number AND u.Transaction_num = @TransactionNo;
END
GO


--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_8 @TransactionNo = 500, @SupCost = 32.00, @JobNo = 2, @AccountIdsAssembly = '1,2', @AccountIdsDepartment = '1,2', @AccountIdsProcess = '2';
```

```sql
-----------------------------------------------------------------------------------------------------------------------
-- Procedure9: Retrieve details-1 on an assembly-id
DROP PROCEDURE IF EXISTS option_9;
GO

CREATE PROCEDURE option_9
    @Assembly_id INT
AS
BEGIN
    SELECT a.details_1
    FROM Assembly_account a
    WHERE a.Account_number = (
        SELECT Account_number
        FROM Record_cost1
        WHERE Assembly_id = @Assembly_id
    );
END
GO



--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_9 @Assembly_id = 1 ;
-----------------------------------------------------------------------------------------------------------------------------------------------------------



-----------------------------------------------------------------------------------------------------------------------
-- Procedure10: Retrieve the total labor time within a department for jobs completed in the department during a given date
DROP PROCEDURE IF EXISTS option_10;
GO

CREATE PROCEDURE option_10
    @Dep_number INT, -- Input parameter: Department number
    @Completion_Start_Date DATE, -- Input parameter: Start date for job completion
    @Completion_End_Date DATE -- Input parameter: End date for job completion
AS
BEGIN
    -- Select the total labor time for all jobs completed in the department within the given date range
    SELECT
        SUM(COALESCE(CJ.labor_time, 0) + COALESCE(PJ.labor_time, 0) + COALESCE(FJ.labor_time, 0)) AS TotalLaborTime
    FROM
        job J WITH(INDEX(idx_Job_CompletedDate))
        LEFT JOIN Cut_job CJ ON J.Job_no = CJ.Job_no
        LEFT JOIN Paint_job PJ ON J.Job_no = PJ.Job_no
        LEFT JOIN Fit_job FJ ON J.Job_no = FJ.Job_no
        JOIN Assign A ON J.Job_no = A.Job_number
        JOIN Supervise S ON A.Process_Id = S.Process_Id
    WHERE
        S.Dep_number = @Dep_number
        AND J.Completed_Date BETWEEN @Completion_Start_Date AND @Completion_End_Date;
END
GO



--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_10 @Dep_number = 1, @Completion_Start_Date = '2005-05-05', @Completion_End_Date = '2035-05-05' ;
-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

```sql
-----------------------------------------------------------------------------------------------------------------------
-- -- Procedure11: Retrieve processes through which a given assembly-id has passed so far
DROP PROCEDURE IF EXISTS option_11;
GO

CREATE PROCEDURE option_11
    @Assembly_id INT
AS
BEGIN
    SELECT
        P.Process_Id,
        P.Process_data AS ProcessDescription,
        D.Dep_num AS DepartmentNumber,
        D.Dep_data AS DepartmentDescription,
        J.Commence_Date
    FROM
        job J
        JOIN Assign A ON J.Job_no = A.Job_number
        JOIN Pass_through PT ON A.Assembly_id = PT.Assembly_id
        JOIN Process P ON PT.Process_Id = P.Process_Id
        JOIN Supervise S ON P.Process_Id = S.Process_Id
        JOIN Department D ON S.Dep_number = D.Dep_num
    WHERE
        A.Assembly_id = @Assembly_id
    ORDER BY
        J.Commence_Date;
END
GO


--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_11 @Assembly_id = 1;
-----------------------------------------------------------------------------------------------------------------------


-----------------------------------------------------------------------------------------------------------------------
-- Procedure12: Retrieve customers whose category is in a given range
DROP PROCEDURE IF EXISTS option_12;
GO

CREATE PROCEDURE option_12
    @minCategory INT, -- Input parameter: Minimum category
    @maxCategory INT -- Input parameter: Maximum category
AS
BEGIN
    SELECT Names, Addresses, Category
    FROM Customer WITH(INDEX(idx_Customer_Category))
    WHERE Category BETWEEN @minCategory AND @maxCategory
    ORDER BY Names;
END
GO

--Executing the procedure: Create an account and associate it with the relevant entity
GO
EXEC option_12 @minCategory = 2, @maxCategory = 6;
-----------------------------------------------------------------------------------------------------------------------

```

```sql
-- Procedure 13: Delete all cut-jobs whose job-no is in a given range
DROP PROCEDURE IF EXISTS option_13;
GO

CREATE PROCEDURE option_13
    @minJobNumber INT, -- Input parameter: Minimum job number
    @maxJobNumber INT -- Input parameter: Maximum job number
AS
BEGIN
    -- Get the job numbers that will be deleted from Cut_job
    DECLARE @deletedJobNumbers TABLE (Job_no INT);

    -- Insert deleted job numbers into the table variable
    INSERT INTO @deletedJobNumbers (Job_no)
    SELECT Job_no
    FROM Cut_job
    WHERE Job_no BETWEEN @minJobNumber AND @maxJobNumber;

    -- Delete from Cut_job
    DELETE FROM Cut_job
    WHERE Job_no BETWEEN @minJobNumber AND @maxJobNumber;

    -- Delete from record only if it corresponds to a deleted cut-job
    DELETE FROM Record
    WHERE Job_number IN (SELECT Job_no FROM @deletedJobNumbers);

    -- Delete from assign only if it corresponds to a deleted cut-job
    DELETE FROM Assign
    WHERE Job_number IN (SELECT Job_no FROM @deletedJobNumbers);

    -- Delete from job only if it corresponds to a deleted cut-job
    DELETE FROM job
    WHERE Job_no IN (SELECT Job_no FROM @deletedJobNumbers);

END
GO

--Executing the procedure:delete
GO
EXEC option_13 @minJobNumber = 1, @maxJobNumber = 6;
```

```sql
-- Procedure14 : Change the color of a given paint job
DROP PROCEDURE IF EXISTS option_14;
GO

CREATE PROCEDURE option_14
    @Job_no INT, -- Input parameter: Job number
    @NewColor VARCHAR(50) -- Input parameter: New color
AS
BEGIN
    UPDATE Paint_job
    SET Color = @NewColor
    WHERE Job_no = @Job_no;
END
GO

--Executing the procedure:update
GO
EXEC option_14 @Job_no = 2, @NewColor = 'Orange';
```

Results Messages

8:11:40 AM	Started executing query at Line 1
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 5
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 15
	(1 row affected)
8:11:40 AM	Started executing query at Line 22
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 31
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 34
	(1 row affected)
8:11:40 AM	Started executing query at Line 41
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 61
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 64
	(1 row affected)
	(1 row affected)
	(1 row affected)
8:11:40 AM	Started executing query at Line 72
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 110
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 113
	(1 row affected)
	(1 row affected)
	(1 row affected)
8:11:40 AM	Started executing query at Line 123
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 162
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 165
	(1 row affected)
	(1 row affected)
	(1 row affected)
8:11:40 AM	Started executing query at Line 174
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 191
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 195
	(1 row affected)
	(1 row affected)
8:11:40 AM	Started executing query at Line 202
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 236
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 241
	(1 row affected)
	(1 row affected)
8:11:40 AM	Started executing query at Line 249
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 298
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 302
	(1 row affected)
	(1 row affected)
	(1 row affected)
	(1 row affected)
	(0 rows affected)
	(1 row affected)
	(1 row affected)
	(0 rows affected)
8:11:40 AM	Started executing query at Line 310
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 324
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 329
	(1 row affected)
8:11:40 AM	Started executing query at Line 337
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 359
	Commands completed successfully.
8:11:40 AM	Started executing query at Line 365
	(1 row affected)

```
8:11:40 AM    Started executing query at Line 372
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 396
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 400
              (1 row affected)
8:11:40 AM    Started executing query at Line 407
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 419
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 422
              (3 rows affected)
8:11:40 AM    Started executing query at Line 429
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 462
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 465
              (1 row affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
              (1 row affected)
8:11:40 AM    Started executing query at Line 472
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 483
              Commands completed successfully.
8:11:40 AM    Started executing query at Line 486
              (1 row affected)
              Total execution time: 00:00:03.510
```

As it is requested all the queries for 1-14 are defined in this section and for the following commands please see the next subsection.

(15) Import: enter new customers from a data file until the file is empty (the user must be asked to enter the input file name).
(16) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen (the user must be asked to enter the output file name).

(17) Quit

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

Answer

Here is the java code for

WELCOME TO THE JOB-SHOP ACCOUNTING DATABASE SYSTEM
(1) Description of query 1
(2) Description of query 2
.
.
.
(14) Description of query 14
(15) Import: enter new customers from a data file until the file is empty (the user must be asked to enter the input file name).
(16) Export: Retrieve the customers (in name order) whose category is in a given range and output them to a data file instead of screen (the user must be asked to enter the output file name).
(17) Quit

As can be seen, each option is defined by case: “num”

```
import java.sql.Connection;
import java.math.BigDecimal;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
public class sample {
  // Database credentials
  final static String HOSTNAME = "basi0011-sql-server.database.windows.net";
  final static String DBNAME = "cs-dsa-4513-sql-db";
  final static String USERNAME = "basi0011";
  final static String PASSWORD = "-------------------------";
  // Database connection string
  final static String URL =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNa
meInCertificate=*.database.windows.net;loginTimeout=30;",
        HOSTNAME, DBNAME, USERNAME, PASSWORD);
  // User input prompt//
  final static String PROMPT =
        "\nPlease select one of the options below: \n" +
        "1) Insert new Customer (option1); \n" +
        "2) Insert new Department (option2); \n" +
        "3) Enter a new process-id and its department (option3); \n" +
        "4) Enter a new assembly (option4); \n" +
        "5) Create a new account (option5); \n" +
        "6) Enter a new job (option6); \n" +
        "7) Completion of a job (option7); \n" +
        "8) Enter a transaction-no and its sup-cost and update all the costs (option8); \n" +
        "9) Retrieve the total cost incurred on an assembly-id (option9); \n" +
        "10) Retrieve the total labor time within a department (option10); \n" +
        "11) Retrieve the processes through which a given assembly-id (option11); \n" +
        "12) Retrieve the customers whose category is in a given range(option12); \n" +
        "13) Delete all cut-jobs whose job-no is in a given range (option13); \n" +
```

```
        "14) Change the color of a given paint job (option14); \n" +
        "15) Import: enter new customers from a data file until the file is empty (option15); \n" +
        "16) Export: Retrieve the customers whose category is in a given range (option16); \n" +
        "17) Quit!";
  public static void main(String[] args) throws SQLException {
    System.out.println("Welcome to the sample application!");
    final Scanner sc = new Scanner(System.in); // Scanner is used to collect the user input
    String option = ""; // Initialize user option selection as nothing
    while (!option.equals("4")) { // As user for options until option 3 is selected
      System.out.println(PROMPT); // Print the available options
      option = sc.next(); // Read in the user option selection
      switch (option) { // Switch between different options
        case "1": // Insert a new Customer  option
          // Collect the new performer data from user
          System.out.println("Please enter Customer Name:");
          sc.nextLine();
          final String Names = sc.nextLine(); // Read in the user input of Customer Name
          System.out.println("Please enter Customer Address:");
          final String Addresses = sc.nextLine(); // Read in the user input of Customer Address
          System.out.println("Please enter Customer category:");
          final int Category = sc.nextInt(); // Read in user input of Customer category

          System.out.println("Connecting to the database...");
          // Get a database connection and prepare a query statement
          try (final Connection connection = DriverManager.getConnection(URL)) {
            try (
              final PreparedStatement statement = connection.prepareStatement("EXEC option_1 @Names = ?,
@Addresses = ?, @Category = ?;")) { //calling option_1 query
              statement.setString(1, Names); //inserting data
              statement.setString(2, Addresses);
              statement.setInt(3, Category);

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

              // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
              statement.executeUpdate(); //want to insert values into query first


              //close statements
              statement.close();
              connection.close();
            }

          }
          break;
        case "2": // Insert a new Department  option
          // Collect the new department data from user
          System.out.println("Please enter Department Number:");
          final int Dep_num = sc.nextInt(); // Read in the user input of department Number
          System.out.println("Please enter Department data:");
          sc.nextLine();
          final String Dep_data = sc.nextLine(); // Read in the user input of Department Data

          System.out.println("Connecting to the database...");
          // Get a database connection and prepare a query statement
          try (final Connection connection = DriverManager.getConnection(URL)) {
            try (
              final PreparedStatement statement = connection.prepareStatement("EXEC option_2 @Dep_num = ?,
@Dep_data = ?;")) { //calling option_2 query
              statement.setInt(1, Dep_num); //inserting data
              statement.setString(2, Dep_data);

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

              // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
              statement.executeUpdate(); //want to insert values into query first
```

```
                    //close statements
                    statement.close();
                    connection.close();
                }

            }
            break;
        case "3": // Insert a new Department  option
            // Collect the new process-id and its department
            System.out.println("Please enter 1 for cut, 2 for paint and 3 for fit:");
            final int CutPaintFit = sc.nextInt(); // Read in the user input of Process ID

            System.out.println("Please enter Process ID:");
            final int Process_Id = sc.nextInt(); // Read in the user input of Process ID

            System.out.println("Please enter Department number:");
            final int Dep_number = sc.nextInt(); // Read in the user input of Dep_number
            System.out.println("Please enter Process data:");
            sc.nextLine();
            final String Process_data = sc.nextLine(); // Read in the user input of Process Data
            System.out.println("Please enter Type:");
            final String Type = sc.nextLine(); // Read in the user input of Type
            System.out.println("Please enter the information of Machine Type(cut) or Painting method(paint):");
            final String Information = sc.nextLine(); // Read in the user input of Information

            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (
                    final PreparedStatement statement = connection.prepareStatement("EXEC option_3 @CutPaintFit = ?,
@Process_Id = ?, @Dep_number = ?, @Process_data = ?, @Type = ?, @Information = ?;")) { //calling option_3 query
                    statement.setInt(1, CutPaintFit); //inserting data
                        statement.setInt(2, Process_Id); //inserting data
                    statement.setInt(3, Dep_number); //inserting data
                    statement.setString(4, Process_data);
                    statement.setString(5, Type);
                    statement.setString(6, Information);

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

                    // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                    statement.executeUpdate(); //want to insert values into query first


                    //close statements
                    statement.close();
                    connection.close();
                }

            }
            break;
        case "4": // Insert a new Assembly option
            // Collect the new Assembly and its association
            System.out.println("Please enter Assembly ID:");
            final int assemblyId4 = sc.nextInt(); // Read in the user input of Assembly ID
            System.out.println("Please enter Customer name:");
            sc.nextLine();
            final String customerName4 = sc.nextLine(); // Read in the user input of Customer name
            System.out.println("Please enter Assembly details:");
            final String assemblyDetails = sc.nextLine(); // Read in the Assembly details
            System.out.println("Please enter the Date_ordered:");
            final String dateOrdered = sc.nextLine();
            // Read in the user input of Process IDs separated by commas
            System.out.println("Please enter Process IDs separated by commas:");
```

```
            final String processIdsInput = sc.nextLine();
            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (
                    final PreparedStatement statement = connection.prepareStatement("EXEC option_4 @Assembly_id = ?,
@Customer_name = ?, @Assembly_details = ?, @Date_ordered = ?, @Process_Ids = ?;")) {
                    // calling option_4 query
                    statement.setInt(1, assemblyId4); // inserting data
                    statement.setString(2, customerName4); // inserting data
                    statement.setString(3, assemblyDetails); // inserting data
                    statement.setString(4, dateOrdered);
                    statement.setString(5, processIdsInput);
                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                    statement.executeUpdate(); // want to insert values into query first
                    // close statements
                    statement.close();
                }
            }
            break;
        case "5": // Insert a new Account option
            // Collect the new account data from the user
            System.out.println("Please enter Account Number:");
            final int Account_number = sc.nextInt(); // Read in the user input of Account Number
            sc.nextLine();
            System.out.println("Please enter Established Date:");
            final String Established_date = sc.nextLine();

            System.out.println("Please enter Account Details (initial cost):");
            final BigDecimal Account_details = sc.nextBigDecimal(); // Read in the user input of Account Details
            sc.nextLine();
            System.out.println("Please enter Entity Type ('process', 'assembly', or 'department'):");
            final String Entity_type = sc.nextLine(); // Read in the user input of Entity Type
            System.out.println("Please enter Entity ID:");
            final int Entity_id = sc.nextInt(); // Read in the user input of Entity ID
            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (
                    final PreparedStatement statement = connection.prepareStatement("EXEC option_5 @Account_number = ?,
@Established_date = ?, @Details = ?, @Entity_type = ?, @Entity_id = ?;")) { // calling Create_Account stored procedure
                    statement.setInt(1, Account_number); // inserting data
                    statement.setString(2, Established_date);
                    statement.setBigDecimal(3, Account_details);
                    statement.setString(4, Entity_type);
                    statement.setInt(5, Entity_id);
                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                    statement.executeUpdate(); // want to insert values into the query first
                    // close statements
                    statement.close();
                    connection.close();
                }
            }
            break;
        case "6": // Enter a new job
            // Collect the new job data from user
            System.out.println("Please enter Job Number:");
            final int Job_no = sc.nextInt(); // Read in the user input of Job Number
            System.out.println("Please enter Assembly ID:");
            final int Assembly_id6 = sc.nextInt(); // Read in the user input of Assembly ID
            System.out.println("Please enter Process ID:");
            final int Process_Id6 = sc.nextInt(); // Read in the user input of Process ID
            System.out.println("Please enter Commence Date (YYYY-MM-DD):");
            final String Commence_Date = sc.next(); // Read in the user input of Commence Date
```

```
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement = connection.prepareStatement("EXEC option_6 @Job_no = ?,
@Assembly_id = ?, @Process_Id = ?, @Commence_Date = ?;")) {
                        statement.setInt(1, Job_no); // Inserting data
                        statement.setInt(2, Assembly_id6);
                        statement.setInt(3, Process_Id6);
                        statement.setString(4, Commence_Date);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                        statement.executeUpdate();
                        // Close statements
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "7": // Complete a job
                // Collect the completion data from the user
                System.out.println("Please enter Job Number:");
                final int Job_no7 = sc.nextInt(); // Read in the user input of Job Number
                System.out.println("Please enter Completed Date (YYYY-MM-DD):");
                final String Completed_Date = sc.next(); // Read in the user input of Completed Date
                System.out.println("Please enter Job Type (cut/paint/fit):");
                final String Job_type = sc.next(); // Read in the user input of Job Type
                System.out.println("Please enter Labor Time:");
                final int Labor_time = sc.nextInt(); // Read in the user input of Labor Time

                System.out.println("Please enter Machine Type(cut) or Color(paint):");
                final String Job_details1 = sc.next(); // Read in the user input of Machine Type(cut) or Color(paint) Details

                System.out.println("Please enter Material(cut):");
                final String Job_details2 = sc.next(); // Read in the user input of Material(cut) Details

                System.out.println("Please enter Amount of time(cut) or Volume(paint):");
                final int Job_details3 = sc.nextInt(); // Read in the user input of Amount of time(cut) or Volume(paint) Details
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement = connection.prepareStatement("EXEC option_7 @Job_no = ?,
@Completed_Date = ?, @Job_type = ?, @labor_time = ?, @Job_details1 = ?, @Job_details2 = ? , @Job_details3 = ?;")) {
                        statement.setInt(1, Job_no7); // Inserting data
                        statement.setString(2, Completed_Date);
                        statement.setString(3, Job_type);
                        statement.setInt(4, Labor_time); // Inserting data
                        statement.setString(5, Job_details1);
                        statement.setString(6, Job_details2);
                        statement.setInt(7, Job_details3);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                        statement.executeUpdate();
                        // Close statements
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "8": // Enter a transaction-no and its sup-cost and update all the costs
                System.out.println("Please enter Transaction Number:");
                final int transactionNumber = sc.nextInt(); // Read in the user input of Transaction Number
                System.out.println("Please enter Sup Cost:");
                final BigDecimal supCost = sc.nextBigDecimal(); // Read in the user input of Sup Cost
                System.out.println("Please enter Job Number:");
```

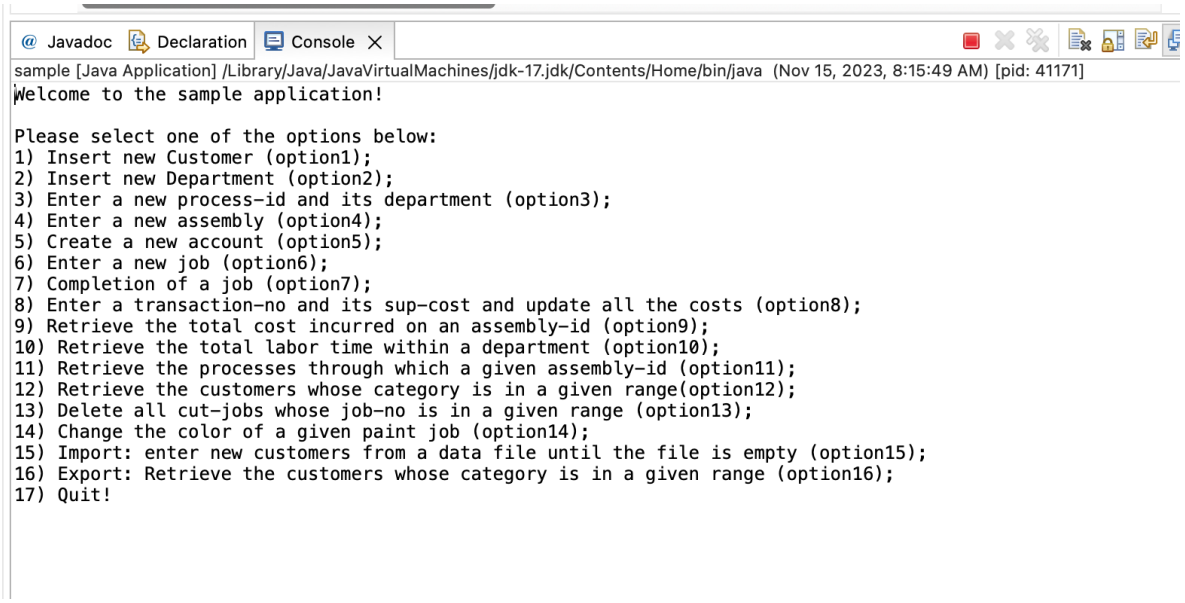
```
                final int jobNumber = sc.nextInt(); // Read in the user input of Job Number
                sc.nextLine(); // Consume the newline character left by sc.nextInt()
                System.out.println("Please enter Account IDs for Assembly (separated by comma):");
                final String accountIdsAssembly = sc.nextLine(); // Read in the user input of Account IDs for Assembly
                System.out.println("Please enter Account IDs for Department (separated by comma):");
                final String accountIdsDepartment = sc.nextLine(); // Read in the user input of Account IDs for Department
                System.out.println("Please enter Account IDs for Process (separated by comma):");
                final String accountIdsProcess = sc.nextLine(); // Read in the user input of Account IDs for Process
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement = connection.prepareStatement(
                            "EXEC option_8 @TransactionNo = ?, @SupCost = ?, @JobNo = ?, @AccountIdsAssembly = ?,
@AccountIdsDepartment = ?, @AccountIdsProcess = ?;")) {
                        statement.setInt(1, transactionNumber); // Inserting data
                        statement.setBigDecimal(2, supCost);
                        statement.setInt(3, jobNumber);
                        statement.setString(4, accountIdsAssembly);
                        statement.setString(5, accountIdsDepartment);
                        statement.setString(6, accountIdsProcess);
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query, executeUpdate is for INSERT, DELETE, and UPDATE statements
                        statement.executeUpdate();
                        // Close the statement and connection
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "9": // Retrieve details-1 on an assembly-id
                // Collect the data from the user
                System.out.println("Please enter Assembly ID:");
                final int Assembly_id9 = sc.nextInt(); // Read in the user input of Assembly ID
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement = connection.prepareStatement("EXEC option_9 @Assembly_id = ?;")) {
                        statement.setInt(1, Assembly_id9); // Inserting data
                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query, executeQuery is for SELECT statements
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            // Process the result set if needed
                            while (resultSet.next()) {
                                // Retrieve details-1
                                BigDecimal details1 = resultSet.getBigDecimal("details_1");

                                // Print or use the retrieved data as needed
                                System.out.println("Details-1 for Assembly ID " + Assembly_id9 + ": " + details1);
                            }
                        }
                        // Close statements
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "10": //Retrieve the total labor time within a department for jobs completed in the department during a given date
                // Collect the data from the user
             System.out.println("Please enter Department Number:");
                final int depNumber = sc.nextInt(); // Read in the user input of Department Number
                System.out.println("Please enter Start Date for Job Completion (YYYY-MM-DD):");
                final String completionStartDate = sc.next(); // Read in the user input of Start Date for Job Completion
                System.out.println("Please enter End Date for Job Completion (YYYY-MM-DD):");
                final String completionEndDate = sc.next(); // Read in the user input of End Date for Job Completion
                System.out.println("Connecting to the database...");
```

```
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement = connection.prepareStatement(
                            "EXEC option_10 @Dep_number = ?, @Completion_Start_Date = ?, @Completion_End_Date = ?;")) {
                        statement.setInt(1, depNumber); // Inserting data
                        statement.setString(2, completionStartDate);
                        statement.setString(3, completionEndDate);
                        System.out.println("Dispatching the query...");
                        // Execute the stored procedure and capture the result set
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            // Process the result set
                            if (resultSet.next()) {
                                int totalLaborTime = resultSet.getInt("TotalLaborTime");
                                System.out.println("Total Labor Time: " + totalLaborTime);
                            } else {
                                System.out.println("No data found.");
                            }
                        }
                        // Close the statement and connection
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "11": // Retrieve processes through which a given assembly-id has passed so far
                // Collect input data from the user
                System.out.println("Please enter Assembly ID:");
                final int assemblyId = sc.nextInt(); // Read in the user input of Assembly ID
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement = connection.prepareStatement("EXEC option_11 @Assembly_id = ?"))
{
                        statement.setInt(1, assemblyId); // Set Assembly ID as a parameter
                        System.out.println("Dispatching the query...");
                        // Execute the stored procedure
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            // Process the result set
                            while (resultSet.next()) {
                                int processId = resultSet.getInt("Process_Id");
                                String processDescription = resultSet.getString("ProcessDescription");
                                int departmentNumber = resultSet.getInt("DepartmentNumber");
                                String departmentDescription = resultSet.getString("DepartmentDescription");
                                String dateCommenced = resultSet.getString("Commence_Date");
                                System.out.println("Process ID: " + processId);
                                System.out.println("Process Description: " + processDescription);
                                System.out.println("Department Number: " + departmentNumber);
                                System.out.println("Department Description: " + departmentDescription);
                                System.out.println("Date Commenced: " + dateCommenced);
                                System.out.println("-----");
                            }
                        }
                        // Close the statement and connection
                        statement.close();
                        connection.close();
                    }
                }
                break;
            case "12": // Retrieve customers whose category is in a given range
                // Collect input data from the user
                System.out.println("Please enter the minimum category:");
                final int minCategory = sc.nextInt(); // Read in the user input of minimum category
                System.out.println("Please enter the maximum category:");
                final int maxCategory = sc.nextInt(); // Read in the user input of maximum category
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
```

```
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement = connection.prepareStatement("EXEC option_12 @minCategory = ?,
@maxCategory = ?")) {
                    statement.setInt(1, minCategory); // Set minimum category as a parameter
                    statement.setInt(2, maxCategory); // Set maximum category as a parameter
                    System.out.println("Dispatching the query...");
                    // Execute the stored procedure
                    try (final ResultSet resultSet = statement.executeQuery()) {
                        // Process the result set
                        while (resultSet.next()) {
                            String customerName = resultSet.getString("Names");
                            String customerAddress = resultSet.getString("Addresses");
                            int customerCategory = resultSet.getInt("Category");
                            System.out.println("Customer Name: " + customerName);
                            System.out.println("Customer Address: " + customerAddress);
                            System.out.println("Customer Category: " + customerCategory);
                            System.out.println("-----");
                        }
                    }
                    // Close the statement and connection
                    statement.close();
                    connection.close();
                }
            }
            break;
        case "13": // Delete all cut-jobs whose job-no is in a given range
            // Collect input data from the user
            System.out.println("Please enter the minimum job number:");
            final int minJobNumber = sc.nextInt(); // Read in the user input of minimum job number
            System.out.println("Please enter the maximum job number:");
            final int maxJobNumber = sc.nextInt(); // Read in the user input of maximum job number
            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement = connection.prepareStatement("EXEC option_13 @minJobNumber = ?,
@maxJobNumber = ?")) {
                    statement.setInt(1, minJobNumber); // Set minimum job number as a parameter
                    statement.setInt(2, maxJobNumber); // Set maximum job number as a parameter
                    System.out.println("Dispatching the query...");
                    // Execute the stored procedure
                    int rowsAffected = statement.executeUpdate();
                    System.out.println("Deleted " + rowsAffected + " cut-jobs.");
                    // Close the statement and connection
                    statement.close();
                    connection.close();
                }
            }
            break;
        case "14":
            System.out.println("Please enter the job number:");
            int jobNumber14 = sc.nextInt();
            sc.nextLine(); // Consume the newline character
            System.out.println("Please enter the new color:");
            String newColor = sc.nextLine();
            System.out.println("Connecting to the database...");
            try (Connection connection = DriverManager.getConnection(URL)) {
                System.out.println("Dispatching the query...");
                try (PreparedStatement statement14 = connection.prepareStatement("EXEC option_14 @Job_no = ?,
@NewColor = ?;")) {
                    statement14.setInt(1, jobNumber14);
                    statement14.setString(2, newColor);
                    System.out.println("Executing the query...");
                    statement14.executeUpdate();
                    System.out.println("Color updated successfully!");
                }
            } catch (SQLException e) {
```

```
                    System.out.println("Error executing the query: " + e.getMessage());
                }
                break;
            case "15":
                System.out.println("Please enter the input file name:");
                final String inputFile = sc.next(); // Read in the user input of the input file name
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (BufferedReader br = new BufferedReader(new
FileReader("/Users/mohammadaminbasiri/eclipse-workspace/SampleAzureSQLProject/" + inputFile + ".csv"))) {
                        String line;
                        String split = ",";
                        while ((line = br.readLine()) != null) {
                            String[] customer = line.split(split);
                            try (final PreparedStatement statement = connection.prepareStatement(
                                    "INSERT INTO Customer (Names, Addresses, Category) VALUES (?, ?, ?)")) {
                                // Populate the query template with the data collected from the user
                                statement.setString(1, customer[0]);
                                statement.setString(2, customer[1]);
                                statement.setInt(3, Integer.parseInt(customer[2]));
                                System.out.println("Dispatching the query ...");
                                // Execute the query
                                final int rowsInserted = statement.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted in customer table.", rowsInserted));
                            }
                        }
                    } catch (FileNotFoundException e) {
                        System.out.println("File not found!");
                        e.printStackTrace();
                    } catch (IOException e) {
                        System.out.println("Error reading file.");
                        e.printStackTrace();
                    }
                }
                break;
            case "16":
                System.out.println("Please enter the lower category:");
                final int lowerCategory = sc.nextInt(); // Read in the user input of lower category
                System.out.println("Please enter the upper category:");
                final int upperCategory = sc.nextInt(); // Read in the user input of upper category
                System.out.println("Please enter the output file name:");
                final String outputFile = sc.next(); // Read in the user input of the output file name
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement = connection.prepareStatement(
                            "SELECT Names as cname, Addresses as caddress, Category FROM Customer WHERE Category
BETWEEN ? AND ? ORDER BY Names");
                         PrintWriter writer = new PrintWriter(new FileWriter(outputFile + ".csv"))) {
                        statement.setInt(1, lowerCategory);
                        statement.setInt(2, upperCategory);
                        System.out.println("Dispatching the query ...");
                        // Execute the query
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            while (resultSet.next()) {
                                StringBuilder sub = new StringBuilder();
                                String name = resultSet.getString("cname");
                                sub.append(name);
                                sub.append(',');
                                String address = resultSet.getString("caddress");
                                sub.append(address);
                                sub.append(',');
                                int category = resultSet.getInt("Category");
                                sub.append(category);
                                sub.append('\n');
```

```
                        writer.write(sub.toString());
                    }
                    System.out.println("Data exported to " + outputFile + ".csv");
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("File not found!");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Error writing to file.");
            e.printStackTrace();
        }
        break;
    case "17": // Do nothing, the while loop will terminate upon the next iteration
        System.out.println("Exiting! Good-buy!");
        System.exit(0);
        break;
    default: // Unrecognized option, re-prompt the user for the correct one
        System.out.println(String.format(
            "Unrecognized option: %s\n" +
            "Please try again!",
            option));
        break;
    }
  }
  sc.close(); // Close the scanner before exiting the application
 }
}
```



```java
24
25      // Database connection string
26      final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;passwor
27              HOSTNAME, DBNAME, USERNAME, PASSWORD);
28
29
30      // User input prompt//
31      final static String PROMPT =
32              "\nPlease select one of the options below: \n" +
33              "1) Insert new Customer (option1); \n" +
34              "2) Insert new Department (option2); \n" +
35              "3) Enter a new process-id and its department (option3); \n" +
36              "4) Enter a new assembly (option4); \n" +
37              "5) Create a new account (option5); \n" +
38              "6) Enter a new job (option6); \n" +
39              "7) Completion of a job (option7); \n" +
40              "8) Enter a transaction-no and its sup-cost and update all the costs (option8); \n" +
41              "9) Retrieve the total cost incurred on an assembly-id (option9); \n" +
42              "10) Retrieve the total labor time within a department (option10); \n" +
43              "11) Retrieve the processes through which a given assembly-id (option11); \n" +
44              "12) Retrieve the customers whose category is in a given range(option12); \n" +
45              "13) Delete all cut-jobs whose job-no is in a given range (option13); \n" +
46              "14) Change the color of a given paint job (option14); \n" +
47              "15) Import: enter new customers from a data file until the file is empty (option15);
48              "16) Export: Retrieve the customers whose category is in a given range (option16); \n
49              "17) Quit!";
50
51      public static void main(String[] args) throws SQLException {
52
53          System.out.println("Welcome to the sample application!");
54
55          final Scanner sc = new Scanner(System.in); // Scanner is used to collect the user input
56          String option = ""; // Initialize user option selection as nothing
57          while (!option.equals("4")) { // As user for options until option 3 is selected
58              System.out.println(PROMPT); // Print the available options
59              option = sc.next(); // Read in the user option selection
60
61              switch (option) { // Switch between different options
62                  case "1": // Insert a new Customer  option
63                      // Collect the new performer data from user
64                      System.out.println("Please enter Customer Name:");
65                      sc.nextLine();
66                      final String Names = sc.nextLine(); // Read in the user input of Customer Nam
67
68                      System.out.println("Please enter Customer Address:");
69                      final String Addresses = sc.nextLine(); // Read in the user input of Customer
70
71                      System.out.println("Please enter Customer category:");
72                      final int Category = sc.nextInt(); // Read in user input of Customer category
```

Javadoc | Declaration | Console

```
sample [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java  (Nov 15, 2023, 8:15:49 AM) [pid: 41171]
Welcome to the sample application!

Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
```

Task 6: ***Java Program Execution***

6.1) Screenshots showing the testing of query 1

I have performed 5 queries like this.

```
Welcome to the sample application!

Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
1
Please enter Customer Name:
Customer 1
Please enter Customer Address:
Add Cust 1
Please enter Customer category:
1
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
508    SELECT * FROM Customer;
```

Results | Messages

	Names	Addresses	Category
1	Ali	Jenkins Ave	5
2	Customer 1	Add Cust 1	1
3	Customer 4	address 4	2
4	Marshall	Houston Ave	8
5	Sean	Devon Energy Hall	6

6.2) Screenshots showing the testing of query 2

I have performed 5 queries like this.

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
2
Please enter Department Number:
2
Please enter Department data:
Department 2
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
509    SELECT * FROM Department;
```

Results | Messages

	Dep_num	Dep_data
1	1	ECE
2	2	Department 2
3	3	DSA
4	4	CS
5	5	dep5

6.3) Screenshots showing the testing of query 3

I have performed 10 queries like this.

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
3
Please enter 1 for cut, 2 for paint and 3 for fit:
1
Please enter Process ID:
1
Please enter Department number:
1
Please enter Process data:
data 1
Please enter Type:
cut1
Please enter the information of Machine Type(cut) or Painting method(paint):
Machine1
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
SELECT * FROM Department;
SELECT * FROM Process;
    SELECT * FROM Process_Cut;
    SELECT * FROM Process_Paint;
    SELECT * FROM Process_Fit;
SELECT * FROM Supervise;
```

Results Messages

	Process_Id	Process_data
1	1	data 1
2	2	process data 2
3	3	data 3
4	4	process4
5	5	data5
6	6	process d6
7	7	data 7
8	8	process 8
9	9	data9
10	10	data10

	Process_Id	Cutting_type	Machine_type
1	1	cut1	Machine1
2	2	type 2	cutting machine
3	3	Type cut 3	machine 3

	Process_Id	Paint_type	Painting_method
1	4	good painting	spray
2	5	bad painting	pencil

	Process_Id	Fit_type
1	6	fit type
2	7	fitting type
3	8	overfit
4	9	underfit
5	10	type10

	Process_Id	Dep_number
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	1
7	7	2
8	8	3
9	9	4
10	10	5

6.4) Screenshots showing the testing of query 4

I have performed 10 queries like this.

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
4
Please enter Assembly ID:
1
Please enter Customer name:
Marshall
Please enter Assembly details:
assembly1
Please enter the Date_ordered:
2011-01-01
Please enter Process IDs separated by commas:
1,9,10
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
515  SELECT * FROM Assemblys;
516  SELECT * FROM Pass_through;
517  SELECT * FROM Orders;
```

Results Messages

	Assembly_id	Date_of_ordered	Assembly_details
1	1	2011-01-01	assembly1
2	2	2002-02-02	detail2
3	3	2003-03-03	detail3
4	4	2004-04-04	Assembly4
5	5	2005-05-05	detail5
6	6	2006-06-06	detail 6
7	7	2007-07-07	details7
8	8	2008-08-08	d8
9	9	2009-09-09	detail9
10	10	2010-10-10	detail 10

	Assembly_id	Process_Id
1	1	1
2	1	9
3	1	10
4	2	2
5	3	3
6	4	1
7	4	2
8	4	3
9	4	4
10	5	5
11	6	4
12	6	6

	Assembly_id	Names
1	1	Marshall
2	3	Ali
3	4	Marshall
4	4	Sean
5	5	Marshall
6	6	Customer 4
7	7	Ali
8	8	Customer 1
9	9	Sean
10	10	Marshall

6.5) Screenshots showing the testing of query 5

I have performed 10 queries like this.

```
Welcome to the sample application!

Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
5
Please enter Account Number:
6
Please enter Established Date:
2006-06-06
Please enter Account Details (initial cost):
60
Please enter Entity Type ('process', 'assembly', or 'department'):
assembly
Please enter Entity ID:
6
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
518  SELECT * FROM Account;
519      SELECT * FROM Process_account;
520      SELECT * FROM Record_cost3;
521      SELECT * FROM Assembly_account;
522      SELECT * FROM Record_cost1;
523      SELECT * FROM Department_account;
524      SELECT * FROM Record_cost2;
```

Results | Messages

	Account_number	Established_date
1	1	2001-01-01
2	2	2002-02-02
3	3	2003-03-03
4	4	2004-04-04
5	5	2005-05-05
6	6	2006-06-06
7	7	2007-07-07
8	8	2008-08-08
9	9	2009-09-09
10	10	2010-10-10

Results grid

	Account_number	details_3
1	1	0.00
2	2	20.00
3	3	30.00
4	4	40.00
5	5	50.00

	Account_number	Process_Id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

	Account_number	details_1
1	6	60.00
2	7	70.00
3	8	80.00

	Account_number	Assembly_id
1	6	6
2	7	7
3	8	8

	Account_number	details_2
1	9	90.00
2	10	100.00

Results grid

	Account_number	Dep_number
1	9	1
2	10	2

6.6) Screenshots showing the testing of query 6

I have performed 10 queries like this.

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
6
Please enter Job Number:
5
Please enter Assembly ID:
5
Please enter Process ID:
5
Please enter Commence Date (YYYY-MM-DD):
2005-05-05
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
525    SELECT * FROM Assign;
526    SELECT * FROM job;
527        SELECT * FROM Cut_job;
528        SELECT * FROM Fit_job;
529        SELECT * FROM Paint_job;
```

Results | Messages

	Assembly_id	Process_Id	job_number
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	8	8	8
8	9	9	9
9	10	10	10

Results grid

	Job_no	Commence_Date	Completed_Date
1	1	2001-01-01	*NULL*
2	2	2002-02-02	*NULL*
3	3	2003-03-03	*NULL*
4	4	2004-04-04	*NULL*
5	5	2005-05-05	*NULL*
6	6	2006-06-06	*NULL*
7	7	2007-07-07	*NULL*
8	8	2008-08-08	*NULL*
9	9	2009-09-09	*NULL*
10	10	2010-10-10	*NULL*

6.7) Screenshots showing the testing of query 7

I have performed 10 queries like this.

```
Welcome to the sample application!

Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
7
Please enter Job Number:
2
Please enter Completed Date (YYYY-MM-DD):
2012-02-02
Please enter Job Type (cut/paint/fit):
cut
Please enter Labor Time:
12
Please enter Machine Type(cut) or Color(paint):
cut2
Please enter Material(cut):
wood
Please enter Amount of time(cut) or Volume(paint):
2
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
526  SELECT * FROM job;
527      SELECT * FROM Cut_job;
528      SELECT * FROM Fit_job;
529      SELECT * FROM Paint_job;
```

Results Messages

	Job_no	Commence_Date	Completed_Date
1	1	2001-01-01	2011-01-01
2	2	2002-02-02	2012-02-02
3	3	2003-03-03	2013-03-03
4	4	2004-04-04	2014-04-04
5	5	2005-05-05	2015-05-05
6	6	2006-06-06	2016-06-06
7	7	2007-07-07	2017-07-07
8	8	2008-08-08	2018-08-08
9	9	2009-09-09	2019-09-09
10	10	2010-10-10	2020-10-10

	Job_no	type_of_machine	amount_of_time	material	labor_time
1	1	cutting	1	knife	11
2	2	cut2	2	wood	12

	Job_no	labor_time
1	6	16
2	7	17
3	8	18
4	9	19
5	10	20

	Job_no	color	volume	labor_time
1	3	orange	3	13
2	4	blue	4	14
3	5	red	5	15

6.8) Screenshots showing the testing of query 8

I have performed 10 queries like this.

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
8
Please enter Transaction Number:
1
Please enter Sup Cost:
0.11
Please enter Job Number:
1
Please enter Account IDs for Assembly (separated by comma):
6,7
Please enter Account IDs for Department (separated by comma):
9
Please enter Account IDs for Process (separated by comma):
1,2,3
Connecting to the database...
Dispatching the query...
```

The results are as follows:

```
518  SELECT * FROM Account;
519      SELECT * FROM Process_account;
520      SELECT * FROM Record_cost3;
521      SELECT * FROM Assembly_account;
522      SELECT * FROM Record_cost1;
523      SELECT * FROM Department_account;
524      SELECT * FROM Record_cost2;
```

Results | Messages

	Account_number	Established_date
1	1	2001-01-01
2	2	2002-02-02
3	3	2003-03-03
4	4	2004-04-04
5	5	2005-05-05
6	6	2006-06-06
7	7	2007-07-07
8	8	2008-08-08
9	9	2009-09-09
10	10	2010-10-10

Results grid

	Account_number	details_3
1	1	0.11
2	2	20.11
3	3	30.11
4	4	40.44
5	5	50.88

	Account_number	Process_Id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

	Account_number	details_1
1	6	61.65
2	7	70.88
3	8	80.33

	Account_number	Assembly_id
1	6	6
2	7	7
3	8	8

Results

	Account_number	details_2
1	9	91.43
2	10	101.11

	Account_number	Dep_number
1	9	1
2	10	2

```
530  SELECT * FROM Record;
531  SELECT * FROM Transactions;
532  SELECT * FROM Updates;
```

Results | Messages

	Transaction_num	job_number
1	1	1
2	2	8
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	2
9	9	9
10	10	10

Results grid

	Transaction_num	Sup_cost
1	1	0.11
2	2	0.22
3	3	0.33
4	4	0.44
5	5	0.55
6	6	0.66
7	7	0.77
8	8	0.88
9	9	0.99
10	10	1.11

	Transaction_num	Account_Number
1	1	1
2	1	2
3	1	3
4	1	6
5	1	7
6	1	9
7	3	5
8	3	8
9	3	9
10	4	4
11	5	5
12	6	6
13	7	7
14	8	6
15	9	9
16	10	10

Results grid

6.9) Screenshots showing the testing of query 9

```
Welcome to the sample application!

Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
9
Please enter Assembly ID:
6
Connecting to the database...
Dispatching the query...
Details-1 for Assembly ID 6: 61.65
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
9
Please enter Assembly ID:
7
Connecting to the database...
Dispatching the query...
Details-1 for Assembly ID 7: 70.88
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
9
Please enter Assembly ID:
8
Connecting to the database...
Dispatching the query...
Details-1 for Assembly ID 8: 80.33
```

6.10) Screenshots showing the testing of query 10

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
10
Please enter Department Number:
1
Please enter Start Date for Job Completion (YYYY-MM-DD):
2000-01-01
Please enter End Date for Job Completion (YYYY-MM-DD):
2023-01-01
Connecting to the database...
Dispatching the query...
Total Labor Time: 27
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
10
Please enter Department Number:
2
Please enter Start Date for Job Completion (YYYY-MM-DD):
2000-01-01
Please enter End Date for Job Completion (YYYY-MM-DD):
2100-01-01
Connecting to the database...
Dispatching the query...
Total Labor Time: 12
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
10
Please enter Department Number:
3
Please enter Start Date for Job Completion (YYYY-MM-DD):
2011-05-05
Please enter End Date for Job Completion (YYYY-MM-DD):
2020-01-01
Connecting to the database...
Dispatching the query...
Total Labor Time: 31
```

6.11) Screenshots showing the testing of query 11

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
11
Please enter Assembly ID:
1
Connecting to the database...
Dispatching the query...
Process ID: 1
Process Description: data 1
Department Number: 1
Department Description: ECE
Date Commenced: 2001-01-01
-----
Process ID: 9
Process Description: data9
Department Number: 4
Department Description: CS
Date Commenced: 2001-01-01
-----
Process ID: 10
Process Description: data10
Department Number: 5
Department Description: dep5
Date Commenced: 2001-01-01
-----
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
11
Please enter Assembly ID:
3
Connecting to the database...
Dispatching the query...
Process ID: 3
Process Description: data 3
Department Number: 3
Department Description: DSA
Date Commenced: 2003-03-03
-----
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
11
Please enter Assembly ID:
5
Connecting to the database...
Dispatching the query...
Process ID: 5
Process Description: data5
Department Number: 5
Department Description: dep5
Date Commenced: 2005-05-05
-----
```

6.12) Screenshots showing the testing of query 12

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
12
Please enter the minimum category:
1
Please enter the maximum category:
9
Connecting to the database...
Dispatching the query...
Customer Name: Ali
Customer Address: Jenkins Ave
Customer Category: 5
-----
Customer Name: Customer 1
Customer Address: Add Cust 1
Customer Category: 1
-----
Customer Name: Customer 4
Customer Address: address 4
Customer Category: 2
-----
Customer Name: Marshall
Customer Address: Houston Ave
Customer Category: 8
-----
Customer Name: Sean
Customer Address: Devon Energy Hall
Customer Category: 6
-----
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
12
Please enter the minimum category:
5
Please enter the maximum category:
8
Connecting to the database...
Dispatching the query...
Customer Name: Ali
Customer Address: Jenkins Ave
Customer Category: 5
-----
Customer Name: Marshall
Customer Address: Houston Ave
Customer Category: 8
-----
Customer Name: Sean
Customer Address: Devon Energy Hall
Customer Category: 6
-----
```

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
12
Please enter the minimum category:
5
Please enter the maximum category:
6
Connecting to the database...
Dispatching the query...
Customer Name: Ali
Customer Address: Jenkins Ave
Customer Category: 5
-----
Customer Name: Sean
Customer Address: Devon Energy Hall
Customer Category: 6
-----
```

6.13) Screenshots showing the testing of query 13

First I will delete using the range

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
13
Please enter the minimum job number:
2
Please enter the maximum job number:
8
Connecting to the database...
Dispatching the query...
Deleted 1 cut-jobs.
```

This was before:

```
525  SELECT * FROM Assign;
526  SELECT * FROM job;
527      SELECT * FROM Cut_job;
```

Results Messages

	Assembly_id	Process_Id	job_number
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	8	8	8
8	9	9	9
9	10	10	10

	Job_no	Commence_Date	Completed_Date
1	1	2001-01-01	2011-01-01
2	2	2002-02-02	2012-02-02
3	3	2003-03-03	2013-03-03
4	4	2004-04-04	2014-04-04
5	5	2005-05-05	2015-05-05
6	6	2006-06-06	2016-06-06
7	7	2007-07-07	2017-07-07
8	8	2008-08-08	2018-08-08
9	9	2009-09-09	2019-09-09
10	10	2010-10-10	2020-10-10

	Job_no	type_of_machine	amount_of_time	material	labor_time
1	1	cutting	1	knife	11
2	2	cut2	2	wood	12

This is after: as you can see the cut job in that range is deleted:

```
525  SELECT * FROM Assign;
526  SELECT * FROM job;
527      SELECT * FROM Cut_job;
```

Results | Messages

	Assembly_id	Process_Id	job_number
1	1	1	1
2	3	3	3
3	4	4	4
4	5	5	5
5	6	6	6
6	8	8	8
7	9	9	9
8	10	10	10

	Job_no	Commence_Date	Completed_Date
1	1	2001-01-01	2011-01-01
2	3	2003-03-03	2013-03-03
3	4	2004-04-04	2014-04-04
4	5	2005-05-05	2015-05-05
5	6	2006-06-06	2016-06-06
6	7	2007-07-07	2017-07-07
7	8	2008-08-08	2018-08-08
8	9	2009-09-09	2019-09-09
9	10	2010-10-10	2020-10-10

	Job_no	type_of_machine	amount_of_time	material	labor_time
1	1	cutting	1	knife	11

Results grid

Now I will try with a range that there are no cut jobs there:

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
13
Please enter the minimum job number:
3
Please enter the maximum job number:
8
Connecting to the database...
Dispatching the query...
Deleted 0 cut-jobs.
```

The tables will remain unchanged:

```
525  SELECT * FROM Assign;
526  SELECT * FROM job;
527      SELECT * FROM Cut_job;
```

Results Messages

	Assembly_id	Process_Id	job_number
1	1	1	1
2	3	3	3
3	4	4	4
4	5	5	5
5	6	6	6
6	8	8	8
7	9	9	9
8	10	10	10

	Job_no	Commence_Date	Completed_Date
1	1	2001-01-01	2011-01-01
2	3	2003-03-03	2013-03-03
3	4	2004-04-04	2014-04-04
4	5	2005-05-05	2015-05-05
5	6	2006-06-06	2016-06-06
6	7	2007-07-07	2017-07-07
7	8	2008-08-08	2018-08-08
8	9	2009-09-09	2019-09-09
9	10	2010-10-10	2020-10-10

	Job_no	type_of_machine	amount_of_time	material	labor_time
1	1	cutting	1	knife	11

Now I will try to delete all:

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
13
Please enter the minimum job number:
1
Please enter the maximum job number:
10
Connecting to the database...
Dispatching the query...
Deleted 1 cut-jobs.
```

```
525  SELECT * FROM Assign;
526  SELECT * FROM job;
527      SELECT * FROM Cut_job;
```

Results | Messages

	Assembly_id	Process_Id	job_number
1	3	3	3
2	4	4	4
3	5	5	5
4	6	6	6
5	8	8	8
6	9	9	9
7	10	10	10

	Job_no	Commence_Date	Completed_Date
1	3	2003-03-03	2013-03-03
2	4	2004-04-04	2014-04-04
3	5	2005-05-05	2015-05-05
4	6	2006-06-06	2016-06-06
5	7	2007-07-07	2017-07-07
6	8	2008-08-08	2018-08-08
7	9	2009-09-09	2019-09-09
8	10	2010-10-10	2020-10-10

	Job_no	type_of_machine	amount_of_time	material	labor_time

Results grid

Also the records table will be updated as well:

```
530  SELECT * FROM Record;
```

Results | Messages

	Transaction_num	job_number
1	2	8
2	3	3
3	4	4
4	5	5
5	6	6
6	7	7
7	9	9
8	10	10

Results grid

6.14) Screenshots showing the testing of query 14

Before doing anything these are the samples inside the paint job

```
529    SELECT * FROM Paint_job;
```

Results | Messages

	Job_no	color	volume	labor_time
1	3	orange	3	13
2	4	blue	4	14
3	5	red	5	15

Results grid

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
14
Please enter the job number:
3
Please enter the new color:
Purple
Connecting to the database...
Dispatching the query...
Executing the query...
Color updated successfully!
```

```
529    SELECT * FROM Paint_job;
```

Results | Messages

	Job_no	color	volume	labor_time
1	3	Purple	3	13
2	4	blue	4	14
3	5	red	5	15

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
14
Please enter the job number:
4
Please enter the new color:
Black
Connecting to the database...
Dispatching the query...
Executing the query...
Color updated successfully!
```

```
529    SELECT * FROM Paint_job;
```

Results | Messages

	Job_no	color	volume	labor_time
1	3	Purple	3	13
2	4	Black	4	14
3	5	red	5	15

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
14
Please enter the job number:
5
Please enter the new color:
White
Connecting to the database...
Dispatching the query...
Executing the query...
Color updated successfully!
```

```
529    SELECT * FROM Paint_job;
```

Results | Messages

	Job_no	color	volume	labor_time
1	3	Purple	3	13
2	4	Black	4	14
3	5	White	5	15

6.15) Screenshots showing the testing of the import and export options

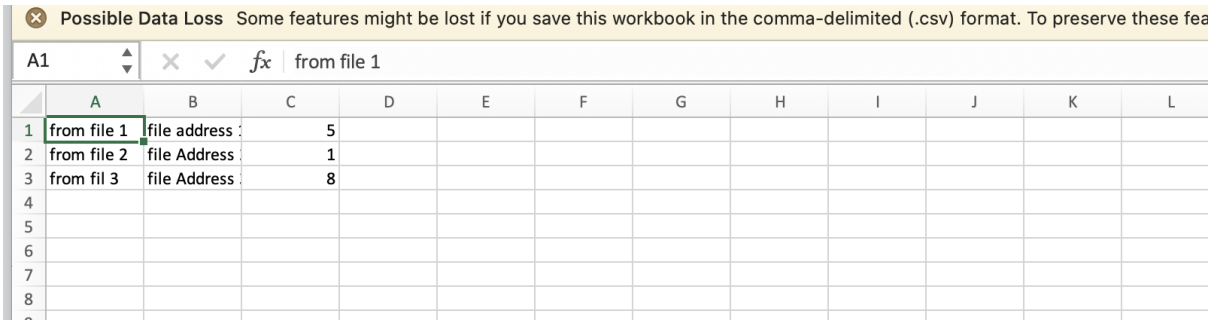
This is my customer table before doing any thing.

```
508  SELECT * FROM Customer;
```

Results | Messages

	Names	Addresses	Category
1	Ali	Jenkins Ave	5
2	Customer 1	Add Cust 1	1
3	Customer 4	address 4	2
4	Marshall	Houston Ave	8
5	Sean	Devon Energy Hall	6

And this is my excel file named test2.csv



After importing:

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
15
Please enter the input file name:
test2
Connecting to the database...
Dispatching the query ...
Done. 1 rows inserted in customer table.
Dispatching the query ...
Done. 1 rows inserted in customer table.
Dispatching the query ...
Done. 1 rows inserted in customer table.
```

```
508  SELECT * FROM Customer;
```

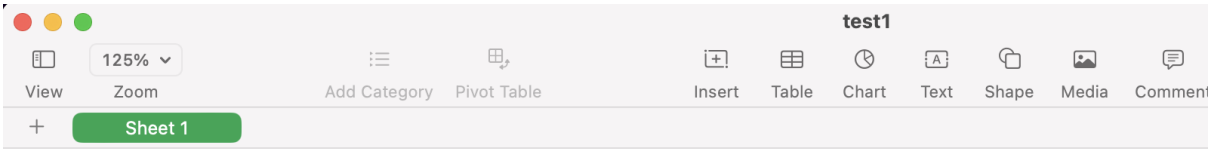
Results | Messages

	Names	Addresses	Category
1	Ali	Jenkins Ave	5
2	Customer 1	Add Cust 1	1
3	Customer 4	address 4	2
4	from fil 3	file Address 3	8
5	from file 1	file address 1	5
6	from file 2	file Address 2	1
7	Marshall	Houston Ave	8
8	Sean	Devon Energy Hall	6

Devon Energy Hall

Now I want to export it to test1.csv

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
16
Please enter the lower category:
2
Please enter the upper category:
8
Please enter the output file name:
test1
Connecting to the database...
Dispatching the query ...
Data exported to test1.csv
```



test1

Ali	**Jenkins Ave**	**5**
Customer 4	address 4	2
from fil 3	file Address 3	8
from file 1	file address 1	5
Marshall	Houston Ave	8
Sean	Devon Energy Hall	6

6.16) Screenshots showing the testing of three types of errors

Primary key violation

```
GO
CREATE PROCEDURE option_1
    @Names VARCHAR(20),
    @Addresses VARCHAR(20),
    @Category INT
AS
BEGIN
INSERT INTO Customer (Names, Addresses, Category) VALUES (@Names, @Addresses, @Category); -- inserting the values
END
--Executing Procedure1
GO
EXEC option_1 @Names = 'Marshall', @Addresses = 'Houston Ave', @Category = 3;
-------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------
---Procedure 02 Enter a new department
DROP PROCEDURE IF EXISTS option_2;
GO

CREATE PROCEDURE option_2
    @Dep_num INT,
    @Dep_data VARCHAR(255)
AS
BEGIN
    INSERT INTO Department (Dep_num, Dep_data) VALUES (@Dep_num, @Dep_data); -- inserting the values
END
```

Messages

11:05:26 AM Started executing query at Line 15

Msg 2627, Level 14, State 1, Procedure option_1, Line 7

Violation of PRIMARY KEY constraint 'PK__Customer__44C0348725DA224E'. Cannot insert duplicate key in object 'dbo.Customer'. The duplicate key value is (Marshall).

The statement has been terminated.

Total execution time: 00:00:00.088

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
1
Please enter Customer Name:
Marshall
Please enter Customer Address:
sdasd
Please enter Customer category:
5
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__Customer__
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLS
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerSta
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedS
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.do
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7620)
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerC
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerSt
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServer
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLS
        at sample.main(sample.java:88)
```

Another primary key violation

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
2
Please enter Department Number:
4
Please enter Department data:
qwerty
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Violation of PRIMARY KEY constraint 'PK__Departme
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQ
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerS
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePrepare
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7620)
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServe
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServer
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServ
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQ
        at sample.main(sample.java:122)
```

```
    INSERT INTO Supervise (Process_Id, Dep_number) VALUES (@Process_Id, @Dep_number);
END
GO

--Executing Procedure3
GO
EXEC option_3 @CutPaintFit = 2, @Process_Id = 4, @Process_data = 'data process a', @Dep_number = 2, @Type = 'Type A', @Information = 'Information';
-------------------------------------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------------------------
--Procedure 04 Enter new assembly with processes
DROP PROCEDURE IF EXISTS option_4;
GO

CREATE PROCEDURE option_4
```

ges

```
06:12 AM    Started executing query at Line 62
            Commands completed successfully.
06:12 AM    Started executing query at Line 64
            Msg 2627, Level 14, State 1, Procedure option_3, Line 11
            Violation of PRIMARY KEY constraint 'PK__Process__35F35AA4F675FA9E'. Cannot insert duplicate key in object 'dbo.Process'. The duplicate key value is (4).
            The statement has been terminated.
            Msg 2627, Level 14, State 1, Procedure option_3, Line 15
            Violation of PRIMARY KEY constraint 'PK__Process___35F35AA43157F8DE'. Cannot insert duplicate key in object 'dbo.Process_Paint'. The duplicate key value is (4).
            The statement has been terminated.
            Msg 2627, Level 14, State 1, Procedure option_3, Line 18
            Violation of PRIMARY KEY constraint 'PK__Supervis__35F35AA4692CB9DA'. Cannot insert duplicate key in object 'dbo.Supervise'. The duplicate key value is (4).
            The statement has been terminated.
            Total execution time: 00:00:00.148
```

Error in wrong data type cannot use int instead of string

```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
6
Please enter Job Number:
88
Please enter Assembly ID:
88
Please enter Process ID:
88
Please enter Commence Date (YYYY-MM-DD):
8
Connecting to the database...
Dispatching the query...
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException: Error converting data type nvarchar to date.
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(SQLS
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(SQLServerSta
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedS
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement$PrepStmtExecCmd.do
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.TDSCommand.execute(IOBuffer.java:7620)
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerConnection.executeCommand(SQLServerC
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeCommand(SQLServerSt
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerStatement.executeStatement(SQLServer
        at com.microsoft.sqlserver.jdbc@11.2.3.jre17/com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.executeUpdate(SQLS
        at sample.main(sample.java:297)
```

We cannot do deletion because of foreign key constraint

```
19    DROP TABLE IF EXISTS Process;
```

Messages

10:57:30 AM Started executing query at Line 19

Msg 3726, Level 16, State 1, Line 1

Could not drop object 'Process' because it is referenced by a FOREIGN KEY constraint.

Total execution time: 00:00:00.091

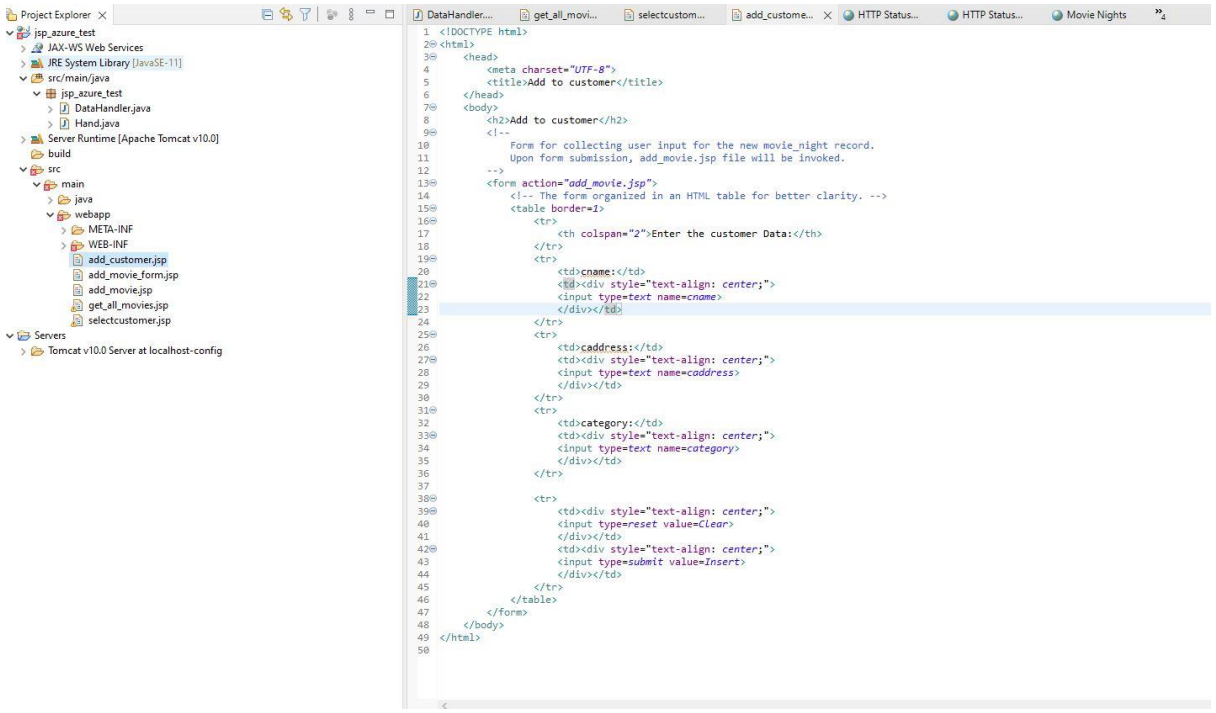
6.17) Screenshots showing the testing of the quit option

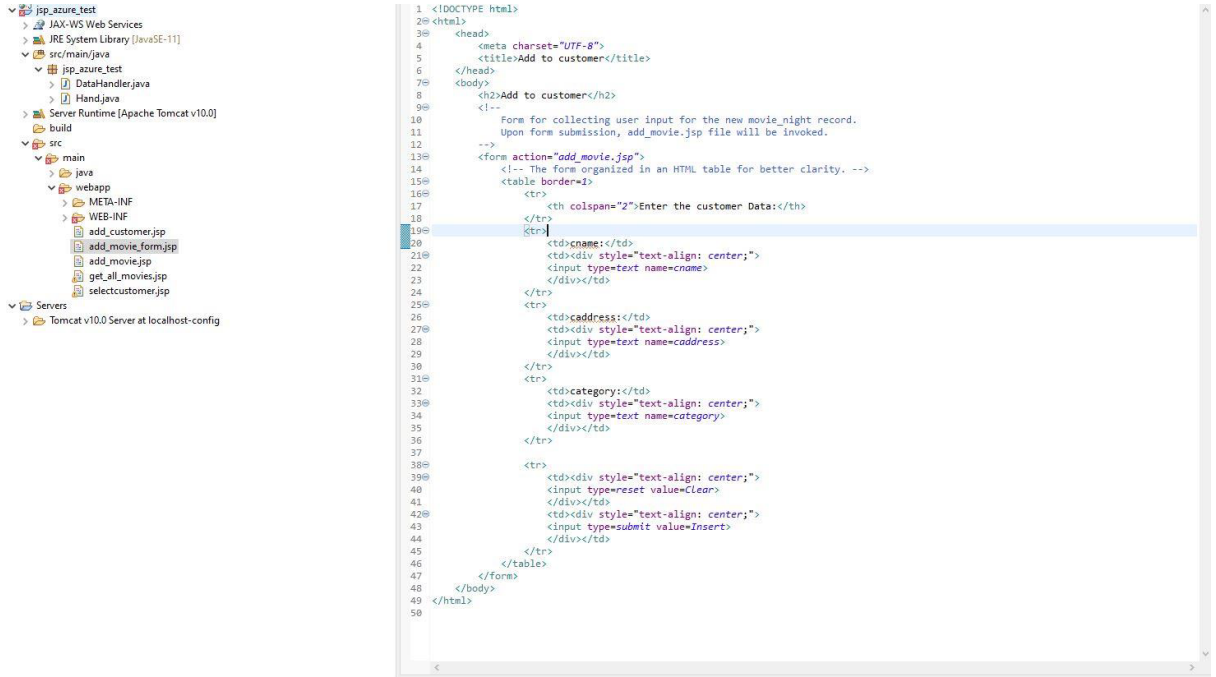
```
Please select one of the options below:
1) Insert new Customer (option1);
2) Insert new Department (option2);
3) Enter a new process-id and its department (option3);
4) Enter a new assembly (option4);
5) Create a new account (option5);
6) Enter a new job (option6);
7) Completion of a job (option7);
8) Enter a transaction-no and its sup-cost and update all the costs (option8);
9) Retrieve the total cost incurred on an assembly-id (option9);
10) Retrieve the total labor time within a department (option10);
11) Retrieve the processes through which a given assembly-id (option11);
12) Retrieve the customers whose category is in a given range(option12);
13) Delete all cut-jobs whose job-no is in a given range (option13);
14) Change the color of a given paint job (option14);
15) Import: enter new customers from a data file until the file is empty (option15);
16) Export: Retrieve the customers whose category is in a given range (option16);
17) Quit!
17
Exiting! Good-buy!
```

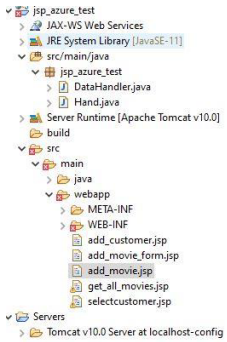
Task 7: ***Web database application and its execution***

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

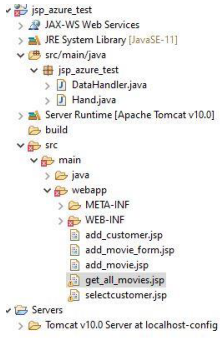
Answer



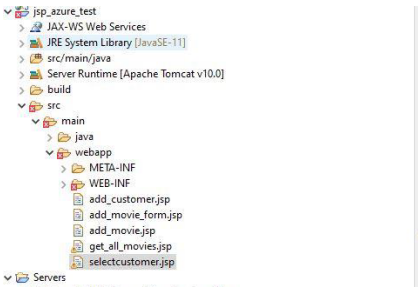





```jsp
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="jsp_azure_test.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

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


    /*
     * If the user hasn't filled out all the time, movie name and duration. This is very simple checking.
     */
    if (cname.equals("") || caddress.equals("") || category.equals("")) {
        response.sendRedirect("add_movie_form.jsp");
    } else {
      // int duration = Integer.parseInt(durationString);

        // Now perform the query with the data from the form.
        boolean success = handler.addMovie(cname, caddress, category);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the course</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The Customer Table:</h2>

            <ul>
                <li>cname: <%=cname%></li>
                <li>caddress: <%=caddress%></li>
                <li>category: <%=category%></li>

            </ul>

            <h2>New Customer successfully inserted.</h2>

            <a href="get_all_movies.jsp">See all Customers.</a>
            <%
        }
    }
    %>
    </body>
</html>
```

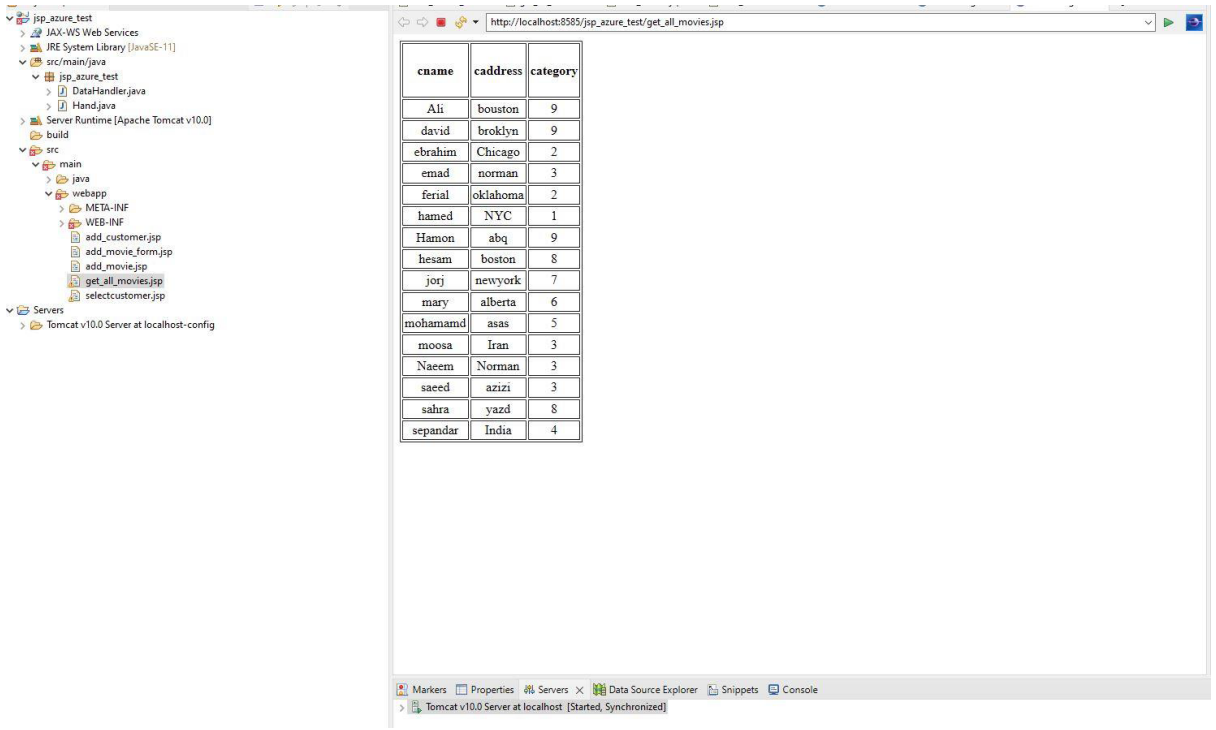



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Movie Nights</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.DataHandler"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get all the movies from the database
            final DataHandler handler = new DataHandler();
            final ResultSet movies = handler.getAllMovies();
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>cname</h4>
              </td>
              <td align="center">
                <h4>caddress</h4>
              </td>
              <td align="center">
                <h4>category</h4>
              </td>


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


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



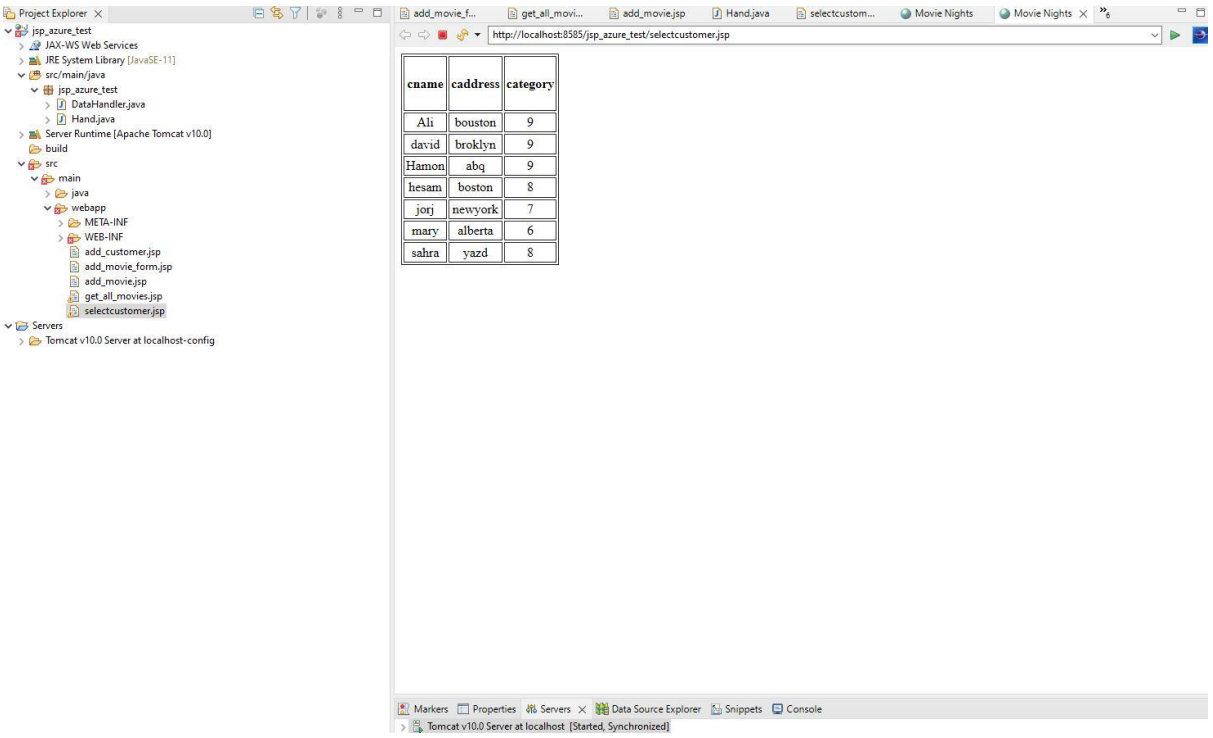

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Movie Nights</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.Hand"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get all the movies from the database
            final Hand handler = new Hand();
            final ResultSet movies = handler.selectcustomer();
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>cname</h4>
              </td>
              <td align="center">
                <h4>caddress</h4>
              </td>
              <td align="center">
                <h4>category</h4>
              </td>


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


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

7.2) Screenshots showing the testing of the Web database application



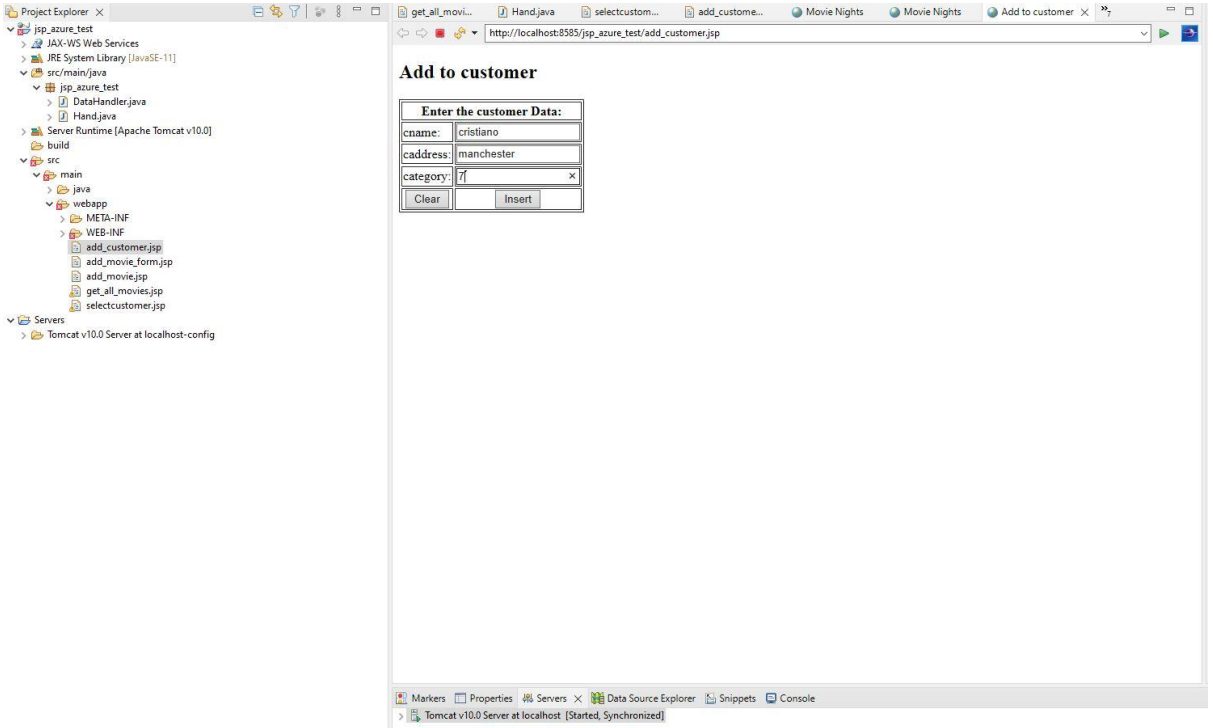
cname	caddress	category
Ali	bouston	9
david	broklyn	9
ebrahim	Chicago	2
emad	norman	3
ferial	oklahoma	2
hamed	NYC	1
Hamon	abq	9
hesam	boston	8
jorj	newyork	7
mary	alberta	6
mohamamd	asas	5
moosa	Iran	3
Naeem	Norman	3
saeed	azizi	3
sahra	yazd	8
sepandar	India	4

Query 12 with the range of 5,9

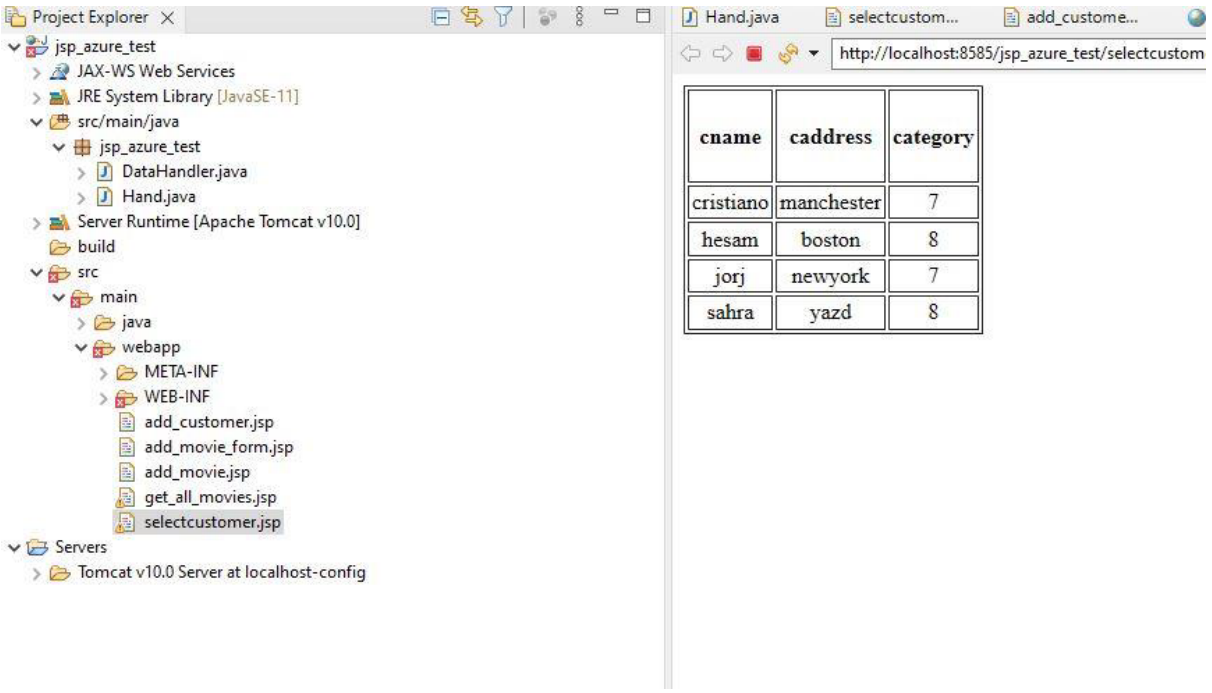


cname	caddress	category
Ali	bouston	9
david	broklyn	9
Hamon	abq	9
hesam	boston	8
jorj	newyork	7
mary	alberta	6
sahra	yazd	8

Running query 1 and adding a new customer.



Running query 12 again with range 6,8: As can be seen, the new customer is inside our new list.


Project Explorer
jsp_azure_test
JAX-WS Web Services
JRE System Library [JavaSE-11]
src/main/java
jsp_azure_test
DataHandler.java
Hand.java
Server Runtime [Apache Tomcat v10.0]
build
src
main
java
webapp
META-INF
WEB-INF
add_customer.jsp
add_movie_form.jsp
add_movie.jsp
get_all_movies.jsp
selectcustomer.jsp
Servers
Tomcat v10.0 Server at localhost-config
Hand.java
selectcustom...
add_custome...
http://localhost:8585/jsp_azure_test/selectcustom
cname
caddress
category
cristiano
manchester
7
hesam
boston
8
jorj
newyork
7
sahra
yazd
8