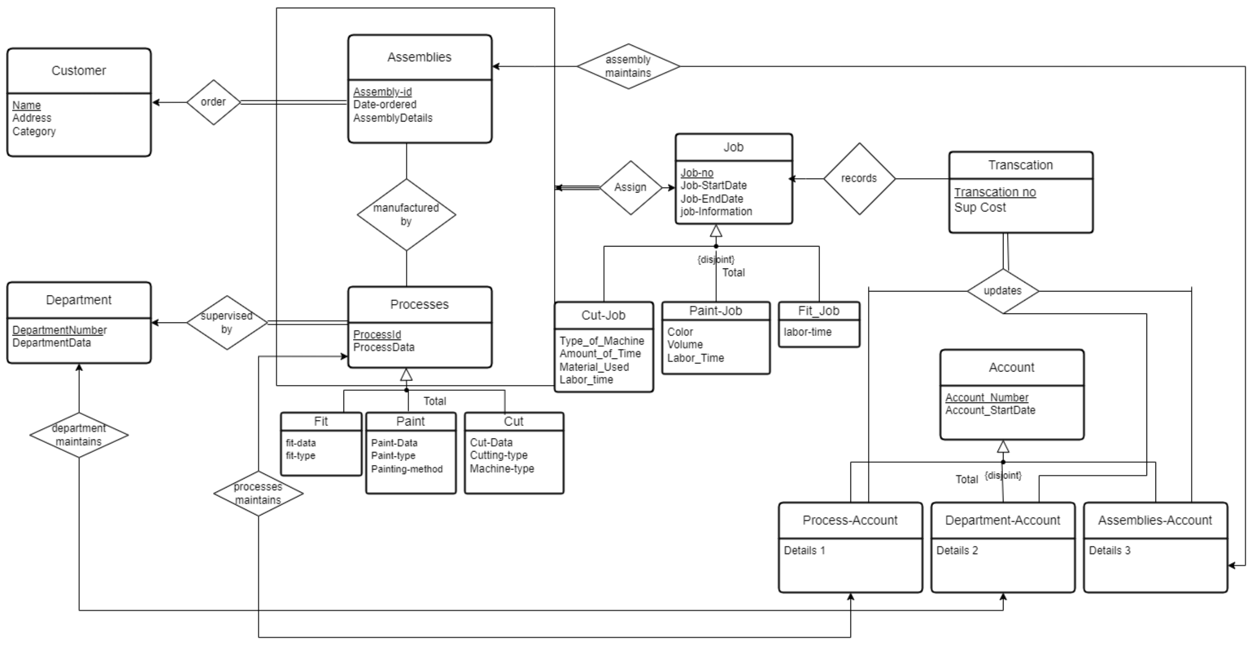
Tasks Performed	Page Number
Task 1. ER Diagram	2
Task 2. Relational Database Schemas	3
Task 3.	
3.1. Discussion of storage structures for tables	3-8
3.2.Discussion of storage structures for tables (Azure SQL Database)	
Task 4. SQL statements and screenshots showing the creation of tables in Azure SQL Database	8-13
Task 5.	
5,1 SQL statements (and Transact SQL stored procedures, if any) 51-54 Implementing all queries (1-15 and error checking) 5,2 The Java source program and screenshots showing 55-60 its successful compilation	14-24
5,2 The Java source program and screenshots showing 55-60 its successful compilation	25-41
Task 6.	
6.1. Java program Execution 61-90 6.1. Screenshots Showing the testing of query 1	42-87
6.2. Screenshots showing the testing of query 2	
.	
.	
.	
.	
.	
6.14. Screenshots showing the testing of query 14	
6.15. Screenshots showing the testing of the import and export options .	
6.16. Screenshots showing the testing of three types of errors	
6.17. Screenshots showing the testing of the quit option	
Task 7.	
Web database application and its execution	88-98

7.1. Web database application source program and screenshots showing Its successful compilation
7.2. Screenshots showing the testing of the Web database application 9

Task 1. (75 points): Design an ER diagram to represent the Job-Shop Accounting database defined in part I.



Task 2. (25 Points): Convert the ER diagram in Task 1 to a Relational Database (i.e. a set of relational schemas).

Customer(CustomerName , Address , Category)
Order(Assembly-Id , CustomerName)
Assemblies(Assembly-Id , Date-Ordered , Assembly Details)
manufactured(Assembly-Id , ProcessId)
Processes(ProcessId , ProcessData)
Fit(ProcessId , fit-data , fit-type)
Paint(ProcessId , Paint - Date , Paint - type , Painting-method)
Cut(ProcessId , Cut-Data , Cutting - type , Machine - type)
Department(DepartmentNumber , DepartmentData)
Supervised(ProcessId , DepartmentNumber)
Job(Job-no , Job-StartDate , Job-EndDate , job-Information)
Cut-Job(Job-no , MachineType , AmountOfTime , MaterialUsed , LaborTime)
Paint-Job(Job-no , LaborTime , Color , Volume , LaborTime)
Fit-Job(Job-no , LaborTime)
Assign(Job-no , AssemblyId , ProcessId)
Transaction(Transaction-no , SupCost)
Records(Transaction-no , Jobno)
Account(AccountNumber , StartDate)
Updates(AccountNumber , Transaction)
ProcessAccount(AccountNumber , Details 1)
DepartmentAccount(AccountNumber , Details 2)
AssembliesAccount(AccountNumber , Details 3)
ProcessMaintain(ProcesId , AccountNumber)
DepartmentMaintain(DepartmentNumber , AccountNumber)
AssemblyMaintain(AssemblyId , AccountNumber)

Task 3. (35 points): 3.1. Discuss choices of appropriate storage structures for each relational table assuming that all types of storage structures discussed in class (Lecture Topic 4) are available. For each table, identify the queries (from the list of the given queries) that access the table, the type of each of those queries (insertion, deletion, random search, or range search), the search keys (if any) involved in each of those queries, the frequency of each of those queries, your choice of the file organization for the table, and your detailed justifications. Use the following format to fill out your answers:

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	1.Insertion		30/day	Sequential Index on search key category	In all these three queries I have chosen range search because it has high frequency and the search key is category .So,i go with sequential index organization.
	12.range	Category	100/day		
Department	2.insertion		Infrequent	heap	Since it is insertions queries I go with heap and it is infrequent
Processes	3.Insertion		Infrequent	heap	Heap organization would be easy
Supervised	3.Insertion		Infrequent	Dynamic	Here we have

	11.random search	ProcessId	100/day	hashing on search key processId	high frequency for 11 th query which is random search on process-id dynamic hashing that would be more appropriate for supervised.
	10.Random search	ProcessId	20/day	ProcessId	
fit	3.Insertion		Infrequent	heap	Here we have high frequency on insert query so we choose heap
Assemblies	4.Insertion		40/day	Dynamic hashing	Here we are doing random search on assemblyId and it has high frequency compared to insertion and I go with dynamic hashing
	11.random search	AssemblyId	100/day		
Manufactured	4.Insertion		40/day	Dynamic hashing on search key Assembly-Id	Random search is good for dynamic hashing because we create buckets on hash key we can easily retrieve the given value here the hash key is
	11.random Search 8.Random Search	AssemblyId	100/day		

					assembly-id
Account	5.Insertion		10/day	heap	Here we have high frequency on insert query so we choose heap
Assemblies-Account	5.Insertion		10/day	Dynamic hashing on search key account no	Here the hash key we are creating buckets on is account number
	8.Update	Account no	50/day		
	9.Random Search	Account no	200/day		
Job	6.Insertion		50/day	B-tree index on search key job no	Here we have random and update search where update has high frequency so, we go with B-tree index
	7.Update	Job no	50/day		
	10.Random	End date	20/day		
Assign	6.Insertion		50/day	Dynamic hashing	In this case we have same frequency for both queries but we have random search and dynamic hashing is appropriate over heap
	8.Random search	JobNo	50/day		
Transaction	8.Insertion		50/day	heap	Here we have high frequency on insert query so we choose heap

Updates	8.Insertion		50/day	heap	Here we have high frequency on insert query so we choose heap
Assembly-maintains	8.Random search	AssemblyId	200/day	Dynamic Hashing on search key assembly-id	Here we have only random search query and it has very high frequency so we go with dynamic hashing.
Cut Jobs	13.Delete 7.Insert	Job no	1/month 50/day	B-tree on jobno	B-tree file organization we usually use for updation and deletion
Paint Job	14.Update 7.Insert	Job no	1/week 50/day	Heap	Here we have high frequency on insert query so we choose heap
Cut	3.Insert		Infrequent	Insert	We just have heap here we can go with heap file organization
Fit Job	7.Insert		50/day	Insert	We just have insert here we can go with heap file organization
Process-maintains	8.Random search	Process-id	50/day	Dynamic hashing	We have only random search

					here we can go with dynamic hashing on hash key process-id
Department_maintains	8.Random Search	Department Number	50/day	Random search	We have only random search here we can go with dynamic hashing on hash key Department-Number
Department_account	5.Insertion		10/day	B tree	Here also we have high frequency for update we can go with B-tree index
	8.Update		50/day		
Process_Account	5.Insertion		10/day	B tree	Same as above because of 8th query we have high frequency for update we can go with b-tree for easy updation.
	8.Update		50/day		
Order	4.Insertion		40/day	heap	Here we have high frequency on insert query so we choose heap

3.2. Discuss choices of storage structures for each relational table when implementing it in Azure SQL Database (if different from the previous choices specified in 3.1). Part of this task is for you to find and study the relevant documentation on your own.
It is not different from the previous choices and I choose to have above storage structures for my relational tables.

Task 4. (23 points): Construct SQL statements to create tables and implement them on Azure SQL Database. All Create statements must include appropriate constraints as defined in Task 2. For each table, you must include SQL statements that create the same storage structure as the one you selected for Azure SQL Database implementation in Task 3.2 (e.g., if you have decided that a table X must have an index on attribute Y, then you must include an SQL statement to create an index on attribute Y for table X).

```
CREATE TABLE Customer (
    CustomerName VARCHAR(255) PRIMARY KEY,
    Address VARCHAR(255),
    Category INT
);

CREATE INDEX CustomerIndex ON Customer(Category);

CREATE TABLE Assemblies (
    AssemblyId INT PRIMARY KEY,
    DateOrdered DATE,
    AssemblyDetails TEXT
);
CREATE INDEX AssembliesIndex ON Assemblies(DateOrdered);

CREATE TABLE Orders (
    AssemblyId INT,
    CustomerName VARCHAR(255),
    PRIMARY KEY (AssemblyId),
    FOREIGN KEY (AssemblyId) REFERENCES Assemblies(AssemblyId),
    FOREIGN KEY (CustomerName) REFERENCES Customer(CustomerName)
);
```

```
CREATE TABLE Processes (
    ProcessId INT PRIMARY KEY,
    ProcessData TEXT
);

CREATE TABLE Manufactured (
    AssemblyId INT,
    ProcessId INT,
    PRIMARY KEY (AssemblyId, ProcessId),
    FOREIGN KEY (AssemblyId) REFERENCES Assemblies(AssemblyId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId)
);

CREATE INDEX Manufactured_hashing_index
ON Manufactured (AssemblyId)

CREATE TABLE Fit (
    ProcessId INT,
    FitData TEXT,
    FitType VARCHAR(100),
    PRIMARY KEY (ProcessId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId)
);

CREATE TABLE Paint (
    ProcessId INT,
    PaintData TEXT,
    PaintType VARCHAR(100),
    PaintingMethod VARCHAR(100),
    PRIMARY KEY (ProcessId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId)
);

CREATE TABLE Cut (
    ProcessId INT,
    CutData TEXT,
    CuttingType VARCHAR(255),
    MachineType VARCHAR(255),
    PRIMARY KEY (ProcessId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId)
);

CREATE TABLE Department (
    DepartmentNumber INT PRIMARY KEY,
    DepartmentData TEXT
);
```

```
CREATE TABLE Supervised (
    ProcessId INT,
    DepartmentNumber INT,
    PRIMARY KEY (ProcessId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId),
    FOREIGN KEY (DepartmentNumber) REFERENCES Department(DepartmentNumber)
);

CREATE INDEX Supervised_hashing_index
ON Supervised (ProcessId);


CREATE TABLE Job (
    JobNo INT PRIMARY KEY,
    JobStartDate DATE,
    JobEndDate DATE,
    JobInformation TEXT
);

CREATE INDEX Job_BTree
ON Job (JobNo);


CREATE TABLE Assign (
    JobNo INT,
    AssemblyId INT,
    ProcessId INT,
    PRIMARY KEY (JobNo),
    FOREIGN KEY (JobNo) REFERENCES Job(JobNo),
    FOREIGN KEY (AssemblyId) REFERENCES Assemblies(AssemblyId),
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId)
);

DROP TABLE CutJob;

CREATE TABLE CutJob (
    JobNo INT PRIMARY KEY,
    MachineType VARCHAR(255),
    AmountOfTime FLOAT,
    MaterialUsed VARCHAR(255),
    LaborTime FLOAT,
    FOREIGN KEY (JobNo) REFERENCES Job(JobNo)
);

CREATE INDEX CutJob_BTreeindex
ON CutJob (JobNo);
```

```
DROP TABLE PaintJob
CREATE TABLE PaintJob (
    JobNo INT PRIMARY KEY,
    LaborTime FLOAT,
    Color VARCHAR(255),
    Volume FLOAT,
    FOREIGN KEY (JobNo) REFERENCES Job(JobNo)
);

CREATE INDEX PaintJob_BTreeindex
ON PaintJob (JobNo);


DROP TABLE FitJob;

CREATE TABLE FitJob (
    JobNo INT PRIMARY KEY,
    LaborTime FLOAT,
    FOREIGN KEY (JobNo) REFERENCES Job(JobNo)
);

CREATE TABLE Transactions (
    TransactionNo INT PRIMARY KEY,
    SupCost DECIMAL(10, 2)
);

CREATE TABLE Records (
    TransactionNo INT,
    JobNo INT,
    PRIMARY KEY (TransactionNo),
    FOREIGN KEY (TransactionNo) REFERENCES Transactions(TransactionNo),
    FOREIGN KEY (JobNo) REFERENCES Job(JobNo)
);

CREATE TABLE Account (
    AccountNumber INT PRIMARY KEY,
    StartDate DATE
);

CREATE INDEX Account_hashing
ON Account(AccountNumber)

CREATE TABLE Updates (
    AccountNumber INT,
    TransactionNo INT,
    PRIMARY KEY (AccountNumber, TransactionNo),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber),
    FOREIGN KEY (TransactionNo) REFERENCES Transactions(TransactionNo)
);

CREATE TABLE ProcessAccount (
    AccountNumber INT,
    Details1 TEXT,
    PRIMARY KEY (AccountNumber),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);

CREATE TABLE DepartmentAccount (
    AccountNumber INT,
    Details2 TEXT,
    PRIMARY KEY (AccountNumber),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);

CREATE TABLE AssembliesAccount (
    AccountNumber INT,
    Details3 TEXT,
    PRIMARY KEY (AccountNumber),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);

CREATE INDEX AssembliesAccount_hashing
ON AssembliesAccount(AccountNumber);


CREATE TABLE ProcessMaintain (
    ProcessId INT PRIMARY KEY,
    AccountNumber INT,
    FOREIGN KEY (ProcessId) REFERENCES Processes(ProcessId),
    FOREIGN KEY (AccountNumber) REFERENCES ProcessAccount(AccountNumber)
);

CREATE INDEX ProcessMantain_hashing
ON ProcessMaintain (ProcessId)

CREATE TABLE DepartmentMaintain (
    DepartmentNumber INT PRIMARY KEY,
    AccountNumber INT,
    FOREIGN KEY (DepartmentNumber) REFERENCES Department(DepartmentNumber),
    FOREIGN KEY (AccountNumber) REFERENCES DepartmentAccount(AccountNumber)
);
```

```
CREATE INDEX DepartmentMaintain_hashing
ON DepartmentMaintain ( DepartmentNumber)

CREATE TABLE AssemblyMaintain (
    AssemblyId INT PRIMARY KEY,
    AccountNumber INT,
    FOREIGN KEY (AssemblyId) REFERENCES Assemblies(AssemblyId),
    FOREIGN KEY (AccountNumber) REFERENCES AssembliesAccount(AccountNumber)
);

CREATE INDEX AssembliesMaintains_hashing
ON AssemblyMaintain (AssemblyId)
```

Task 5. (Task 5 and Task 6 together = 119 points): Write SQL statements for all queries (1-14) defined in part I. Write a Java application program that uses JDBC and Azure SQL Database to implement all SQL queries (options 1-14), two additional queries for import and export (options 15- 16), and the “Quit” option (option 17) as specified in the menu given below. You are free to pick any file format you wish to use for file import and export options. The program will stop execution only when the user chooses the “Quit” option; otherwise, all options must be available for the user to choose at all times. Your program must be commented properly.

SQL QUERIES:

```sql
--1st query
CREATE PROCEDURE FIRSTQUERY
    @CustomerName VARCHAR(50),
    @Address VARCHAR(100),
    @Category INT
AS
BEGIN
    INSERT INTO Customers (CustomerName, Address, Category)
    VALUES (@CustomerName, @Address, @Category);
END;

--2nd query
CREATE PROCEDURE SECONDQUERY
    @DepartmentNumber INT,
    @DepartmentData VARCHAR(200)

AS
BEGIN
    INSERT INTO Customers (DepartmentNumber,  DepartmentData)
    VALUES (@DepartmentNumber, @DepartmentData);
END;


--3rd query
DROP PROCEDURE THIRDQUERY
CREATE PROCEDURE THIRDQUERY
    @ProcessId INT,
    @ProcessData NVARCHAR(MAX),
    @DepartmentNumber INT,
   -- @DepartmentData NVARCHAR(MAX),
    @InsertFit BIT,
    @FitData NVARCHAR(MAX), -- Corrected: Added data type
    @FitType NVARCHAR(100),
    @InsertCut BIT,
    @CutData NVARCHAR(MAX),
    @CuttingType NVARCHAR(MAX),
    @MachineType NVARCHAR(MAX),
    @InsertPaint BIT,
    @PaintData NVARCHAR(MAX),
    @PaintType NVARCHAR(100),
    @PaintingMethod NVARCHAR(300)

AS
BEGIN
SET NOCOUNT ON;
    BEGIN TRY
```

```
        BEGIN TRANSACTION;
    -- Insert into Processes table
    INSERT INTO Processes (ProcessId, ProcessData)
    VALUES (@ProcessId, @ProcessData);


    -- Insert into Department table
    --INSERT INTO Department (DepartmentNumber, DepartmentData)
    --VALUES (@DepartmentNumber, @DepartmentData);


    -- Insert into Supervised table
    INSERT INTO Supervised (ProcessId, DepartmentNumber)
    VALUES (@ProcessId, @DepartmentNumber);

    -- Conditionally insert into Fit table
    IF @InsertFit = 1
    BEGIN
        INSERT INTO Fit (ProcessId, FitData, FitType)
        VALUES (@ProcessId, @FitData, @FitType);
    END

    -- Conditionally insert into Cut table
    IF @InsertCut = 1
    BEGIN
        INSERT INTO Cut (ProcessId , CutData, CuttingType, MachineType)
        VALUES (@ProcessId , @CutData, @CuttingType, @MachineType);
    END

    -- Conditionally insert into Paint table
    IF @InsertPaint = 1
    BEGIN
        INSERT INTO Paint (ProcessId , PaintData, PaintType, PaintingMethod)
        VALUES (@ProcessId ,@PaintData, @PaintType, @PaintingMethod);
    END
    COMMIT;
    END TRY
    BEGIN CATCH
        -- An error occurred, roll back the transaction
        IF @@TRANCOUNT > 0
            ROLLBACK;

        -- Raise the error
        THROW;
    END CATCH;
END;
```

```
--4th query
 DROP PROCEDURE FOURTHQUERY
 CREATE PROCEDURE FOURTHQUERY
    @AssemblyId INT,
    @DateOrdered DATE,
    @AssemblyDetails NVARCHAR(MAX),
    @CustomerName NVARCHAR(MAX),
    @ProcessId INT
AS
BEGIN
SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
    -- Insert into Assemblies table
    INSERT INTO Assemblies (AssemblyId, DateOrdered, AssemblyDetails)
    VALUES (@AssemblyId, @DateOrdered, @AssemblyDetails);


    -- Insert into Customer table
    --INSERT INTO Customer (CustomerName)
    --VALUES (@CustomerName);


    -- Insert into Orders table
    INSERT INTO Orders (CustomerName, AssemblyId)
    VALUES (@CustomerName, @AssemblyId);

    -- Insert into Manufactured table
    INSERT INTO Manufactured (AssemblyId, ProcessId)
    VALUES (@AssemblyId, @ProcessId);
    COMMIT;
    END TRY
    BEGIN CATCH
        -- An error occurred, roll back the transaction
        IF @@TRANCOUNT > 0
            ROLLBACK;

        -- Raise the error
        THROW;
    END CATCH;
END;

--5 th query
DROP PROCEDURE FIFTHQUERY
CREATE PROCEDURE FIFTHQUERY
    @AccountNumber INT,
    @StartDate DATE,
    --@Details3 NVARCHAR(MAX),
    --@Details1 NVARCHAR(MAX),
    --@Details2 NVARCHAR(MAX),
    @InsertDepartment INT = 0,
    @InsertProcesses INT = 0,
    @InsertAssemblies INT = 0,
    @AssemblyId INT,
    @ProcessId INT,
    @DepartmentNumber INT
AS
BEGIN
SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
    -- Insert into Account table
    INSERT INTO Account (AccountNumber, StartDate)
    VALUES (@AccountNumber, @StartDate);

    IF @InsertAssemblies = 1
    BEGIN
        INSERT INTO AssembliesAccount (AccountNumber, Details3)
        VALUES (@AccountNumber, '0');
        INSERT INTO AssemblyMaintain(AssemblyId , AccountNumber)
        VALUES (@AssemblyId , @AccountNumber)
    END

    IF @InsertProcesses = 1
    BEGIN
        INSERT INTO ProcessAccount (AccountNumber, Details1)
        VALUES (@AccountNumber, '0');
        INSERT INTO ProcessMaintain(ProcessId , AccountNumber)
        VALUES (@ProcessId , @AccountNumber)
    END

    IF @InsertDepartment = 1
    BEGIN
        INSERT INTO DepartmentAccount (AccountNumber, Details2)
        VALUES (@AccountNumber, '0');
        INSERT INTO DepartmentMaintain(DepartmentNumber , AccountNumber)
        VALUES (@DepartmentNumber , @AccountNumber)
    END
    COMMIT;
    END TRY
    BEGIN CATCH
        -- An error occurred, roll back the transaction
        IF @@TRANCOUNT > 0
            ROLLBACK;
        -- Raise the error
        THROW;
    END CATCH;
END;

--6 TH QUERY
DROP PROCEDURE SIXTHQUERY
CREATE PROCEDURE SIXTHQUERY
    @AssemblyId INT,
    --@DateOrdered DATE,
    --@AssemblyDetails NVARCHAR(MAX),
    @ProcessId INT,
    --@ProcessData NVARCHAR(MAX),
    @JobNo INT,
    @JobStartDate DATE
AS
BEGIN
SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
    -- Insert into Job table
    INSERT INTO Job (JobNo, JobStartDate)
    VALUES (@JobNo, @JobStartDate);

    -- Insert into Assign table
    INSERT INTO Assign (JobNo, AssemblyId, ProcessId)
    VALUES (@JobNo, @AssemblyId, @ProcessId);
    COMMIT;
    END TRY
    BEGIN CATCH
        -- An error occurred, roll back the transaction
        IF @@TRANCOUNT > 0
            ROLLBACK;

        -- Raise the error
        THROW;
    END CATCH;
END;

EXEC SEVENTHQUERY @JobNo=?, @JobEndDate=?, @JobInformation=?,
@InsertFitJob=?,@LaborTime=?, @InsertCutJob=?, @TypeOfMachine=?, @AmountOfTime=?,
@MaterialUsed=?,@LaborTime1=? , @InsertPaintJob=?, @Color=?, @Volume=?, @LaborTime2=?;
--7th query
DROP PROCEDURE SEVENTHQUERY
CREATE PROCEDURE SEVENTHQUERY
    @JobNo INT,
    @JobEndDate NVARCHAR(10),
    @JobInformation NVARCHAR(300),
    @InsertFitJob BIT,
    @LaborTime FLOAT,
    @InsertCutJob BIT,
    @TypeOfMachine NVARCHAR(100),
    @AmountOfTime FLOAT,
    @MaterialUsed NVARCHAR(100),
    @LaborTime1 FLOAT,
    @InsertPaintJob BIT,
    @Color NVARCHAR(255),
    @Volume FLOAT,
    @LaborTime2 FLOAT
AS
BEGIN
SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
    -- Check if JobNo exists
    IF NOT EXISTS (SELECT 1 FROM Job WHERE JobNo = @JobNo)
    BEGIN
        -- Raise an error because JobNo doesn't exist
        THROW 51000, 'The specified JobNo does not exist.', 1;
        RETURN; -- Terminate the procedure
    END

    -- Update Job
    UPDATE Job
    SET JobEndDate = @JobEndDate,
    JobInformation = @JobInformation
    WHERE JobNo = @JobNo;

    IF @InsertCutJob = 1
    BEGIN
        INSERT INTO CutJob (JobNo,MachineType, AmountOfTime, MaterialUsed, LaborTime)
        VALUES (@JobNo,@TypeOfMachine, @AmountOfTime, @MaterialUsed, @LaborTime1);
    END

    IF @InsertPaintJob = 1
    BEGIN
        INSERT INTO PaintJob (JobNo,Color, Volume, LaborTime)
        VALUES (@JobNo,@Color, @Volume, @LaborTime2);
    END

    IF @InsertFitJob = 1
    BEGIN
        INSERT INTO FitJob (JobNo,LaborTime)
        VALUES (@JobNo,@LaborTime);
    END
    COMMIT;
    END TRY
    BEGIN CATCH
        -- An error occurred, roll back the transaction
        IF @@TRANCOUNT > 0
            ROLLBACK;

        -- Raise the error
        THROW;
    END CATCH;
END;


CREATE PROCEDURE EIGHTQUERY
    @TransactionNo INT,
    @SupCost INT,
    @JobNo INT
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION
            DECLARE @AssignedProcessID INT;
            DECLARE @AssignedAssemblyID INT;
            DECLARE @AssignedDepartmentID INT;
            DECLARE @PAccountID INT;
            DECLARE @AAccountID INT;
            DECLARE @DAccountID INT;

            SELECT
                @AssignedProcessID = A.ProcessId,
                @AssignedAssemblyID = A.AssemblyId,
                @AssignedDepartmentID = S.DepartmentNumber
            FROM
                Assign A
            INNER JOIN
                Supervised S ON A.ProcessId = S.ProcessId
            WHERE
                A.JobNo = @JobNo;

            SELECT @PAccountID = AccountNumber FROM ProcessMaintain WHERE ProcessId =
@AssignedProcessID;
            SELECT @AAccountID = AccountNumber FROM AssemblyMaintain WHERE AssemblyId
= @AssignedAssemblyID;
```

```
            SELECT @DAccountID = AccountNumber FROM DepartmentMaintain WHERE
DepartmentNumber = @AssignedDepartmentID;


            INSERT INTO Transactions (TransactionNo, SupCost)
            VALUES (@TransactionNo, @SupCost);

            INSERT INTO Records (JobNo, TransactionNo)
            VALUES (@TransactionNo, @JobNo)

            INSERT INTO Updates (AccountNumber, TransactionNo)
            VALUES
            (@PAccountID, @TransactionNo),
            (@AAccountID, @TransactionNo),
            (@DAccountID, @TransactionNo);

            UPDATE AssembliesAccount SET Details3 += @SupCost WHERE AccountNumber =
@AAccountID;
            UPDATE DepartmentAccount SET Details2 += @SupCost WHERE AccountNumber =
@DAccountID;
            UPDATE ProcessAccount SET Details1 += @SupCost WHERE AccountNumber =
@PAccountID;
        COMMIT;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK;
        THROW;
    END CATCH;
END;

--9 th query
CREATE PROCEDURE NINETHQUERY
AS
BEGIN
   SELECT aa.Details3 FROM AssembliesAccount AS aa , Assemblies AS a ,
AssemblyMaintain AS m
   WHERE m.AssemblyId = a.AssemblyId AND m.AccountNumber = aa.AccountNumber
END;

--9th query with assembly id as parameter
DROP PROCEDURE NINETHQUERY
CREATE PROCEDURE NINETHQUERY
    @AssemblyId INT
AS
BEGIN
    DECLARE @AccountNumber INT;
    -- Retrieve AccountNumber based on the provided AssemblyId
    SELECT @AccountNumber = m.AccountNumber
    FROM Assemblies AS a
    INNER JOIN AssemblyMaintain AS m ON m.AssemblyId = a.AssemblyId
    WHERE a.AssemblyId = @AssemblyId;

    -- If @AccountNumber is NULL, it means there was no matching record
    IF @AccountNumber IS NOT NULL
    BEGIN
        -- Now you have the AccountNumber, and you can use it to retrieve Details3
        SELECT aa.Details3
        FROM AssembliesAccount AS aa
        WHERE aa.AccountNumber = @AccountNumber;
    END
    ELSE
    BEGIN
        -- Handle the case when there is no matching record
        PRINT 'No matching record found for the provided AssemblyId.';
    END
END;

--10 th query
DROP PROCEDURE TENTHQUERY
CREATE PROCEDURE TENTHQUERY
    @DepartmentNumber INT,
    @JobEndDate DATE
AS
BEGIN
SELECT SUM(jt.LaborTime) as 'Total Labor Time' FROM Assign AS a , Job AS j, Processes
AS p , Supervised AS s , Department AS d,
(
    SELECT JobNo, LaborTime FROM CutJob
    UNION
    SELECT JobNo, LaborTime FROM FitJob
    UNION
    SELECT JobNo, LaborTime FROM PaintJob
) as jt

WHERE p.ProcessId = a.ProcessId AND a.JobNo = jt.JobNo AND s.ProcessId = p.ProcessId
AND
s.DepartmentNumber = d.DepartmentNumber and d.DepartmentNumber = @Departmentnumber and
j.JobEndDate = @JobEndDate

END;

EXEC TENTHQUERY @DepartmentNumber=2020,@JobEndDate='2023-11-13'
```

```
DROP Procedure Query10;
--11 th query
CREATE PROCEDURE ELEVENTHQUERY
    @AssemblyId INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Assemblies WHERE AssemblyId = @AssemblyId)
    BEGIN
        SELECT p.ProcessId , d.DepartmentNumber  FROM Processes AS p , Assemblies AS a
,
         Department AS d , Manufactured AS m , Supervised AS s
        WHERE m.AssemblyId = a.AssemblyId and m.ProcessId = p.ProcessId AND
s.ProcessId = p.ProcessId and
         s.DepartmentNumber = d.DepartmentNumber AND a.AssemblyId = @AssemblyId
        ORDER BY a.DateOrdered
    END
    ELSE
    BEGIN
        THROW 51000, 'The AssemblyId does not exist.', 1;
    END
END;

--12 TH TH QUERY
DROP PROCEDURE TWELVETHQUERY
CREATE PROCEDURE TWELVETHQUERY
    @CategoryFrom INT,
    @CategoryTo INT
AS
BEGIN
    SELECT CustomerName , Address
    FROM Customer
    WHERE Category >= @CategoryFrom AND Category <= @CategoryTo
    ORDER BY CustomerName;
END;

--13 th query
DROP PROCEDURE THIRTEENTHQUERY
CREATE PROCEDURE THIRTEENTHQUERY
    @JobNoFrom INT,
    @JobNoTo INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM CutJob WHERE JobNo BETWEEN @JobNoFrom AND @JobNoTo)
    BEGIN

    DELETE FROM CutJob
```

```sql
    WHERE JobNo BETWEEN @JobNoFrom AND @JobNoTo;
    END
    ELSE
    BEGIN
    THROW 51000, 'No matching CutJobs found for the specified JobNo range.', 1;
    END
END;

--14 th query
CREATE PROCEDURE FOURTHTEENQUERY
@JobNo INT,
@NewColor VARCHAR(20)
AS
BEGIN
    IF EXISTS (SELECT 1 FROM PaintJob WHERE JobNo = @JobNo)
    BEGIN
     UPDATE PaintJob
     SET Color = @NewColor
     WHERE JobNo = @JobNo
    END
    ELSE
    BEGIN
    THROW 51000, 'The JobNo does not exist in the paint-table.', 1;
    END
END;
```

JAVA CODE:

```java
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.FileReader;
import java.util.InputMismatchException;
public class project {
  // Database credentials
  final static String HOSTNAME = "nand0019.database.windows.net";
  final static String DBNAME = "cs-dsa-4513-sql-db";
  final static String USERNAME = "nand0019";
  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;trustServerCe
rtificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
        HOSTNAME, DBNAME, USERNAME, PASSWORD);
  // Query templates
  final static String QUERY_TEMPLATE_1 = "INSERT INTO Customer " +
                              "VALUES (?, ?, ?);";
  final static String QUERY_TEMPLATE_2 = "INSERT INTO Department " +
                              "VALUES (?, ?);";
  final static String QUERY_TEMPLATE_3 = "EXEC THIRDQUERY @ProcessId=?, @ProcessData=?,
@DepartmentNumber=?, @InsertFit=?,\r\n"
              + "           @FitData=?, @FitType=?, @InsertCut=?, @CutData=?, @CuttingType=?,
@MachineType=?, \r\n"
              + "           @InsertPaint=?, @PaintData=?, @PaintType=?, @PaintingMethod=?;";
  final static String QUERY_TEMPLATE_4 = "EXEC FOURTHQUERY
@AssemblyId=?,@DateOrdered=?,@AssemblyDetails=?,@CustomerName=? , @ProcessId=?;";
  final static String QUERY_TEMPLATE = "INSERT INTO Manufactured (AssemblyId, ProcessId)
VALUES (?, ?);";
  final static String QUERY_TEMPLATE_5 = "EXEC FIFTHQUERY @AccountNumber = ?, @StartDate
=
?,@InsertAssemblies=?,@InsertProcesses=?,@InsertDepartment=?,@ProcessId=?,@AssemblyId=?,@
DepartmentNumber=?;";
  final static String QUERY_TEMPLATE_6 = "EXEC SIXTHQUERY
@AssemblyId=?,@ProcessId=?,@JobNo=?,@JobStartDate=?;";
  final static String QUERY_TEMPLATE_7 = "EXEC SEVENTHQUERY @JobNo=?, @JobEndDate=?,
@JobInformation=?, @InsertFitJob=?,@LaborTime=?, @InsertCutJob=?, @TypeOfMachine=?,
@AmountOfTime=?, @MaterialUsed=?,@LaborTime1=? , @InsertPaintJob=?, @Color=?, @Volume=?,
@LaborTime2=?;";
  final static String QUERY_TEMPLATE_8 =  "EXEC EIGHTQUERY
@TranscationNo=?,@SupCost=?,@UpdateProcessAccount=?,@AccountNumber1=?,@UpdateDepartm
entAccount=?,@AccountNumber2=?,@UpdateAssembliesAccount=?,@AccountNumber3=?;";
  final static String QUERY_TEMPLATE_9 = "EXEC NINETHQUERY @AssemblyId = ?;";
  final static String QUERY_TEMPLATE_10 = "EXEC TENTHQUERY @DepartmentNumber=? ,
@JobEndDate=?;";
  final static String QUERY_TEMPLATE_11 = "EXEC ELEVENTHQUERY @AssemblyId = ?;";
  final static String QUERY_TEMPLATE_12 = "EXEC TWELVETHQUERY @CategoryFrom=? ,
@CategoryTo=?;";
  final static String QUERY_TEMPLATE_13 = "EXEC THIRTEENTHQUERY @JobNoFrom=? ,
@JobNoTo=?;";
  final static String QUERY_TEMPLATE_14 = "EXEC FOURTHTEENQUERY @JobNo=? ,
@NewColor=?;";
  private static final String OUTPUT_FILE_PATH = "C:/Users/nandi/Downloads/outputfile.txt";
  // User input prompt//
  final static String PROMPT =
        "\nPlease select one of the options below: \n" +
        "1) Insert new Customer; \n" +
        "2) Insert new Department; \n" +
          "3) Insert process-id and its department together with its type; \n" +
          "4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and dateordered
and associate it with one or more processes; \n" +
          "5)Create a new account and associate it with the process, assembly, or department;\n"+
          "6)Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced;\n"+
           "7)At the completion of a job, enter the date it completed and the information relevant to the
type\r\n"
           + "of job ;\n"+
           "8)Enter a transaction-no and its sup-cost and update all the costs (details) of the affected\r\n"
           + "accounts by adding sup-cost to their current values of details;\n"+
           "9)Retrieve the total cost incurred on an assembly-id;\n"+
           "10)Retrieve the total labor time within a department for jobs completed in the department during
a\r\n"
           + "given date;\n"+
           "11) Retrieve the processes through which a given assembly-id has passed so far (in
datecommenced order) \r\n"
           + "and the department responsible for each process;\n"+
           "12)Retrieve the customers (in name order) whose category is in a given range;\n"+
           "13)Delete all cut-jobs whose job-no is in a given range;\n"+
           "14)Change the color of a given paint job;\n"+
            "15)Import: enter new customers from a data file until the file is empty.;\n"+
            "16)  Export: Retrieve the customers (in name order) whose category is in a given range and\r\n"
            + "output them to a data file instead of screen ;\n"+
            "17)Exit";

   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("14")) { // 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
           sc.nextLine(); // Consume the newline character left by next()
           switch (option) { // Switch between different options
               case "1": // Insert a new student option
                   // Collect the new student data from the user
                   System.out.println("Please enter Customer Name:");
                   final String name = sc.nextLine(); // Read in the user input of student ID
                   System.out.println("Please enter Customer Address:");

                   final String address = sc.nextLine(); // Read in user input of student First Name (white-spaces
allowed).
                   System.out.println("Please enter Category:");
                   // No need to call nextLine extra time here, because the preceding nextLine consumed the
newline character.
                   final int category = sc.nextInt(); // Read in user input of student Last Name (white-spaces
allowed).
```

```
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_1)) {
                        // Populate the query template with the data collected from the user
                        statement.setString(1, name);
                        statement.setString(2, address);
                        statement.setInt(3, category);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    } catch(SQLException e) {
                       System.err.println("Error Occured: "  + e.getMessage());

                    }
                }
                break;
            case "2":

                   System.out.println("Please enter Department Number:");
                   int departmentNo = 0;
                try {
                      departmentNo = sc.nextInt();
                } catch (InputMismatchException e) {
                    System.err.println("Error: Please enter a valid integer for the Department Number.");
                    break;
                }// Read in the user input of student ID
                sc.nextLine();
                System.out.println("Please enter Department Data:");
                // Preceding nextInt, nextFloar, etc. do not consume new line characters from the user input.
                // We call nextLine to consume that newline character, so that subsequent nextLine doesn't
return nothing.
                final String deptData = sc.nextLine(); // Read in user input of student First Name
(white-spaces allowed).
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (
                        final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_2)) {
                        // Populate the query template with the data collected from the user
                        statement.setInt(1, departmentNo);
                        statement.setString(2, deptData);

                        System.out.println("Dispatching the query...");
                        // Actually execute the populated query
                        final int rows_inserted = statement.executeUpdate();
                        System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                    }
                    catch(SQLException e) {
                      System.err.println("Error Occured: "  + e.getMessage());

                    }

                }
                break;
            case "3":
                System.out.println("Please enter ProcessId:");
                final int ProcessId = sc.nextInt();
                sc.nextLine();
                System.out.println("Please enter ProcessData:");
                final String ProcessData = sc.nextLine();
                System.out.println("Please enter the DepartmentNumber which you have previously entered
:");
                final int DepartmentNumber = sc.nextInt();
                sc.nextLine();
                /*
                System.out.println("Please enter the DepartmentData :");
                final String DepartmentData = sc.nextLine();
                */
                System.out.println("Please type 0 or 1 to continue with the fit type insert:");
                final int InsertFit = sc.nextInt();
                sc.nextLine();
                String FitData = null;
                String FitType = null;
                if (InsertFit == 1) {
                    System.out.println("Please enter FitData:");
                    FitData = sc.nextLine();
                    System.out.println("Please enter FitType:");
                    FitType = sc.nextLine();
                }
                System.out.println("Please type 0 or 1 to continue with the Cut type insert:");
                final int InsertCut = sc.nextInt();
                sc.nextLine();
                String CutData = null;
                String CuttingType = null;
                String MachineType = null;
                if (InsertCut == 1) {
                    System.out.println("Please enter CutData");
                    CutData = sc.nextLine();
                    System.out.println("Please enter CuttingType:");
                    CuttingType = sc.nextLine();
                    System.out.println("Please enter MachineType:");
                    MachineType = sc.nextLine();
                }
```

```
                System.out.println("Please type 0 or 1 to continue with the Paint type insert:");
                final int InsertPaint = sc.nextInt();
                sc.nextLine();
                String PaintData = null;
                String PaintType = null;
                String PaintingMethod = null;
                if (InsertPaint == 1) {
                  System.out.println("Please enter PaintData");
                  PaintData = sc.nextLine();
                  System.out.println("Please enter PaintType:");
                  PaintType = sc.nextLine();
                  System.out.println("Please enter PaintingMethod:");
                  PaintingMethod = sc.nextLine();
                }
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                  try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_3)) {
                    // Populate the query template with the data collected from the user
                    statement.setInt(1, ProcessId);
                    statement.setString(2, ProcessData);
                    statement.setInt(3, DepartmentNumber);
                   //statement.setString(4, DepartmentData);
                    statement.setInt(4, InsertFit);
                    statement.setString(5, FitData); // These variables are now in scope
                    statement.setString(6, FitType); // These variables are now in scope
                    statement.setInt(7, InsertCut);
                    statement.setString(8, CutData); // These variables are now in scope
                    statement.setString(9, CuttingType); // These variables are now in scope
                    statement.setString(10, MachineType); // These variables are now in scope
                    statement.setInt(11, InsertPaint);
                    statement.setString(12, PaintData); // These variables are now in scope
                    statement.setString(13, PaintType); // These variables are now in scope
                    statement.setString(14, PaintingMethod); // These variables are now in scope
                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final int rows_inserted = statement.executeUpdate();
                    System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                  }
                  catch(SQLException e) {
                   System.err.println("Error Occured: "  + e.getMessage());

                  }
                }
                break;
              case "4":
                   System.out.println("Please enter AssemblyId:");
                   final int AssemblyId = sc.nextInt();
```

```
                    sc.nextLine();
                    System.out.println("Please enter DateOrdered:");
                    final String DateOrdered = sc.nextLine();
                    System.out.println("Please enter AssemblyDetails:");
                    final String AssemblyDetails = sc.nextLine();
                    System.out.println("Please enter the CustomerName which you have previously
entered:");
                    final String CustomerName = sc.nextLine();
                    System.out.println("How many processes do you want to enter?");
                    int numProcesses = sc.nextInt();
                    sc.nextLine(); // Consume the newline character
                     System.out.println("Please enter the ProcessIds Which you have previously
entered(separated by spaces):");
                     String processIdsInput = sc.nextLine();
                     String[] processIdsArray = processIdsInput.split(" ");


                    System.out.println("Connecting to the database...");
                    try (final Connection connection = DriverManager.getConnection(URL)) {
                       try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_4 )){
                                statement.setInt(1, AssemblyId);
                              statement.setString(2, DateOrdered);
                              statement.setString(3, AssemblyDetails);
                              statement.setString(4, CustomerName);
                              statement.setInt(5, Integer.parseInt(processIdsArray[0])); // Insert the first process
                              final int rows_inserted = statement.executeUpdate();
                          }
                       catch(SQLException e) {
                    System.err.println("Error Occured: "  + e.getMessage());

                  }

                       if (numProcesses > 1) {
                     for (int i = 1; i < numProcesses; i++) {
                        try (final PreparedStatement manufacturedStatement =
connection.prepareStatement(QUERY_TEMPLATE)) {
                            manufacturedStatement.setInt(1, AssemblyId);
                            manufacturedStatement.setInt(2, Integer.parseInt(processIdsArray[i]));
                            final int rows_inserted = manufacturedStatement.executeUpdate();
                        }
                        catch(SQLException e) {
                              System.err.println("Error Occured: "  + e.getMessage());

                        }
                     }
                  }
```

```
        }
    break;

case "5":
    System.out.println("Please enter AccountNumber");
    final int AccountNumber = sc.nextInt();
    sc.nextLine();
    System.out.println("Please enter  Start Date:");
    final String AccountStartDate = sc.nextLine();

    System.out.println("Please type 0 or 1 to continue with Assemblies Account:");

    final int InsertAssemblies = sc.nextInt();

    sc.nextLine();
    int AssemblyId11 = 0;

    if (InsertAssemblies == 1) {
          /*
        System.out.println("Please enter Details3");
        Details3 = sc.nextLine();
        */
          System.out.println("Please Enter the existing AssemblyId:");

           AssemblyId11 = sc.nextInt();

    }


    System.out.println("Please type 0 or 1 to continue with Processes Account:");

    final int InsertProcesses = sc.nextInt();

    sc.nextLine();
    int PROCESSID = 0;

    if (InsertProcesses == 1) {
          /*
        System.out.println("Please enter Details1");
        Details1 = sc.nextLine();
        */
           System.out.println("Please Enter EXISTING Process Id:");

         PROCESSID = sc.nextInt();
    }
```

```
                System.out.println("Please type 0 or 1 to continue with the Department Account:");

                final int InsertDepartment = sc.nextInt();

                sc.nextLine();
                int DEPARTMENTNUMBER = 0;

                if (InsertDepartment == 1) {
                    /*
                  System.out.println("Please enter Details 2:");
                  Details2 = sc.nextLine();
                  */
                    System.out.println("Please Enter existing Department Number:");

                   DEPARTMENTNUMBER = sc.nextInt();
                }



                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                  try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_5)) {
                      // Populate the query template with the data collected from the user
                      statement.setInt(1, AccountNumber);
                      statement.setString(2, AccountStartDate);
                      statement.setInt(3, InsertAssemblies);
                      //statement.setString(4, Details3);
                      statement.setInt(4, InsertProcesses);
                      //statement.setString(6, Details1);
                      statement.setInt(5, InsertDepartment);
                      //statement.setString(8, Details2);
                      statement.setInt(6, PROCESSID);
                      statement.setInt(7, AssemblyId11);
                      statement.setInt(8, DEPARTMENTNUMBER);


                      System.out.println("Dispatching the query...");
                      // Actually execute the populated query
                      final int rows_inserted = statement.executeUpdate();
                      System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                  }
                  catch(SQLException e) {
                    System.err.println("Error Occured: "  + e.getMessage());

                  }
                }
                break;
```

```
        case "6":
                System.out.println("Please enter AssemblyId:");
                final int AssemblyId1 = sc.nextInt();
                sc.nextLine();
                /*
                System.out.println("Please enter DateOrdered:");
                final String DateOrdered1 = sc.nextLine();
                System.out.println("Please enter AssemblyDetails:");
                final String AssemblyDetails1 = sc.nextLine();
                */
                 System.out.println("Please enter ProcessId:");
           final int ProcessId1 = sc.nextInt();
           sc.nextLine();
           /*
           System.out.println("Please enter ProcessData:");
           final String ProcessData1 = sc.nextLine();
           */
           System.out.println("Please enter JobNo:");
           final int JobNo = sc.nextInt();
           sc.nextLine();
           System.out.println("Please enter Job Start Date:");
           final String JobStartDate = sc.nextLine();

          System.out.println("Connecting to the database...");
          // Get a database connection and prepare a query statement
          try (final Connection connection = DriverManager.getConnection(URL)) {
             try (
                final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_6)) {
                // Populate the query template with the data collected from the user
                statement.setInt(1, AssemblyId1);
                //statement.setString(2, DateOrdered1);
                //statement.setString(3, AssemblyDetails1);
                statement.setInt(2, ProcessId1);
               // statement.setString(5, ProcessData1);
                statement.setInt(3, JobNo);
                statement.setString(4, JobStartDate);

                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
             }
             catch(SQLException e) {
               System.err.println("Error Occured: "  + e.getMessage());

             }
          }
          break;
```

```
case "7":
        System.out.println("Please enter JobNo:");
    final int JobNo1 = sc.nextInt();
    sc.nextLine();
    System.out.println("Please enter JobEndDate:");
    final String JobEndDate = sc.nextLine();

    System.out.println("Please enter JobInformation:");
    final String JobInformation = sc.nextLine();
    System.out.println("Please type 0 or 1 to continue with the fit-job type insert:");
    final int InsertFitType = sc.nextInt();
    sc.nextLine();
    float LaborTime = 0.0f;
    if (InsertFitType == 1) {
        System.out.println("Please enter LaborTime:");
        LaborTime = sc.nextFloat();

    }
    System.out.println("Please type 0 or 1 to continue with the Cut-job type insert:");
    final int InsertCutjob = sc.nextInt();
    sc.nextLine();
    String TypeOfMachine = null;
    float AmountOfTime = 0.0f;
    String MaterialUsed = null;
    float LaborTime1 = 0.0f;
    if (InsertCutjob == 1) {
        System.out.println("Please enter Type of Machine");
        TypeOfMachine = sc.nextLine();
        System.out.println("Please enter AmountOfTime:");
        AmountOfTime = sc.nextFloat();
        System.out.println("Please enter Material Used:");
        MaterialUsed = sc.nextLine();
        sc.nextLine();
        System.out.println("Please enter Labor Time:");
        LaborTime1 = sc.nextFloat();
    }
    System.out.println("Please type 0 or 1 to continue with the Paint-job type insert:");
    final int InsertPaintJob = sc.nextInt();
    sc.nextLine();
    String Color = null;
    float Volume = 0.0f;;
    float LaborTime2 = 0.0f;
    if (InsertPaintJob == 1) {
        System.out.println("Please enter Color");
        Color = sc.nextLine();
        System.out.println("Please enter Volume:");
        Volume = sc.nextFloat();
        System.out.println("Please enter LaborTime:");
        LaborTime2 = sc.nextFloat();
```

```
                }
                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                   try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_7)) {
                      // Populate the query template with the data collected from the user
                      statement.setInt(1, JobNo1);
                      statement.setString(2, JobEndDate);
                      statement.setString(3, JobInformation);
                      statement.setInt(4, InsertFitType);
                      statement.setFloat(5, LaborTime);
                      statement.setInt(6, InsertCutjob); // These variables are now in scope
                      statement.setString(7, TypeOfMachine); // These variables are now in scope
                      statement.setFloat(8, AmountOfTime);
                      statement.setString(9, MaterialUsed);
                      statement.setFloat(10, LaborTime1);// These variables are now in scope
                      statement.setInt(11, InsertPaintJob); // These variables are now in scope
                      statement.setString(12, Color); // These variables are now in scope
                      statement.setFloat(13, Volume);
                      statement.setFloat(14, LaborTime2); // These variables are now in scope

                      System.out.println("Dispatching the query...");
                      // Actually execute the populated query
                      final int rows_inserted = statement.executeUpdate();
                      System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
                   }
                   catch(SQLException e) {
                     System.err.println("Error Occured: "  + e.getMessage());

                   }
                }
                break;
             case "8":
                System.out.println("Please enter TranscationNo:");
                final int TranscationNo1 = sc.nextInt();
                sc.nextLine();
                System.out.println("Please enter SupCost:");
                final int SupCost1 = sc.nextInt();
                System.out.println("Please enter JobNo:");
                final int JobNo2 = sc.nextInt();

                System.out.println("Connecting to the database...");
                // Get a database connection and prepare a query statement
                try (final Connection connection = DriverManager.getConnection(URL)) {
                   try (final PreparedStatement statement = connection.prepareStatement("{CALL
EIGHTQUERY(?,?,?)}")) {
                      // Populate the query template with the data collected from the user
                      statement.setInt(1, TranscationNo1);
                statement.setInt(2, SupCost1);
                statement.setInt(3, JobNo2);
                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
            }
            catch(SQLException e) {
               System.err.println("Error Occured: "  + e.getMessage());

            }
        }

               break;
        case "9":
            System.out.println("Please enter AssemblyId:");
            final int AssemblyID = sc.nextInt();
            sc.nextLine();
            System.out.println("Connecting to the database...");
            try (final Connection connection = DriverManager.getConnection(URL)) {
                try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_9)) {
                    // Populate the query template with the data collected from the user
                    statement.setInt(1, AssemblyID);
                    System.out.println("Dispatching the query...");

                    // Execute the query and store the result in a ResultSet
                    try (final ResultSet resultSet = statement.executeQuery()) {
                        System.out.println("Details 3:");
                        // Unpack the tuples returned by the database and print them out to the user
                        while (resultSet.next()) {
                            System.out.println(resultSet.getString(1));
                        }
                    }
                }
                catch(SQLException e) {
                   System.err.println("Error Occured: "  + e.getMessage());

                }
            }
            break;
        case "10":
            System.out.println("Please enter Department Number:");
            final int DptNo = sc.nextInt();
            sc.nextLine();
            System.out.println("Please enter JobEndDate:");
            final String JobEndingDate = sc.nextLine();
            System.out.println("Connecting to the database...");
            try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_10)) {
                        // Populate the query template with the data collected from the user
                        statement.setInt(1, DptNo);

                        statement.setString(2, JobEndingDate);
                        System.out.println("Dispatching the query...");

                        // Execute the query and store the result in a ResultSet
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            System.out.println("Total Labor Time :");
                            // Unpack the tuples returned by the database and print them out to the user
                            while (resultSet.next()) {
                                    System.out.println(String.format("%s   ",
                                        resultSet.getString(1)
                                        ));
                            }
                        }
                    }
                    catch(SQLException e) {
                      System.err.println("Error Occured: "  + e.getMessage());

                    }
                }
            case "11":
                System.out.println("Please enter AssemblyId:");
                final int ASSEMBLYID = sc.nextInt();

                System.out.println("Connecting to the database...");
                try (final Connection connection = DriverManager.getConnection(URL)) {
                    try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_11)) {
                        // Populate the query template with the data collected from the user
                        statement.setInt(1, ASSEMBLYID);

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

                        // Execute the query and store the result in a ResultSet
                        try (final ResultSet resultSet = statement.executeQuery()) {
                            System.out.println("ProcessId | Department Number :");
                            // Unpack the tuples returned by the database and print them out to the user
                            while (resultSet.next()) {
                                    System.out.println(String.format("%s | %s |  ",
                                        resultSet.getString(1),
                                        resultSet.getString(2)));
                            }
                        }
                    }
                    catch(SQLException e) {
```

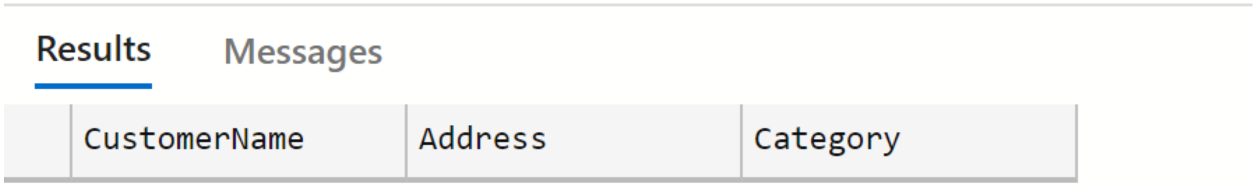
```
                System.err.println("Error Occured: "  + e.getMessage());

               }
            }
            break;
         case "12":
            System.out.println("Please enter CategoryFrom:");
            final int CategoryFrom = sc.nextInt();
            System.out.println("Please enter CategoryTo:");
            final int CategoryTo = sc.nextInt();

            System.out.println("Connecting to the database...");
            try (final Connection connection = DriverManager.getConnection(URL)) {
               try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_12)) {
                  // Populate the query template with the data collected from the user
                  statement.setInt(1, CategoryFrom);
                  statement.setInt(2, CategoryTo);
                  System.out.println("Dispatching the query...");

                  // Execute the query and store the result in a ResultSet
                  try (final ResultSet resultSet = statement.executeQuery()) {
                           System.out.println("CustomerName| Address");

                     // Unpack the tuples returned by the database and print them out to the user
                     while (resultSet.next()) {
                        System.out.println(String.format("%s | %s | ",
                           resultSet.getString(1),
                           resultSet.getString(2)));
                     }
                  }
               }
               catch(SQLException e) {
                System.err.println("Error Occured: "  + e.getMessage());

               }
            }
            break;
         case "13":
            System.out.println("Please enter JobNoFrom:");
            final int JobNoFrom = sc.nextInt();
            System.out.println("Please enter JobNoTo:");
            final int JobNoTo = sc.nextInt();

            System.out.println("Connecting to the database...");
            try (final Connection connection = DriverManager.getConnection(URL)) {
               try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_13)) {
                  // Populate the query template with the data collected from the user
                statement.setInt(1, JobNoFrom);
                statement.setInt(2, JobNoTo);
                System.out.println("Dispatching the query...");

                // Execute the query and store the result in a ResultSet
                final int rows_deleted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows deleted.", rows_deleted));
                  }
              catch(SQLException e) {
                System.err.println("Error Occured: "  + e.getMessage());

              }
                }
            break;
          case "14":
            System.out.println("Please enter Job Number:");
            final int JobNO = sc.nextInt();
            sc.nextLine();
            System.out.println("Please enter New Color:");
            final String NewColor = sc.nextLine();

            System.out.println("Connecting to the database...");
            try (final Connection connection = DriverManager.getConnection(URL)) {
              try (final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_14)) {
                // Populate the query template with the data collected from the user
                statement.setInt(1, JobNO);
                statement.setString(2, NewColor);
                System.out.println("Dispatching the query...");

                // Execute the query and store the result in a ResultSet
                final int Updated = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows Updated.", Updated));
                  }
              catch(SQLException e) {
                System.err.println("Error Occured: "  + e.getMessage());

              }
                }
              break;
          case "15":
            System.out.println("Please enter the input text file");
            String fileName = sc.nextLine();

            System.out.println("Please enter CategoryFrom:");
            final int CategoryFrom1 = sc.nextInt();
            System.out.println("Please enter CategoryTo:");
            final int CategoryTo1 = sc.nextInt();
            System.out.println("Connecting to the database...");
```

```
                try (final Connection connection = DriverManager.getConnection(URL);
                     final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_12);
                     PrintWriter writer = new PrintWriter(new FileWriter(fileName))) {
                    // Populate the query template with the data collected from the user
                    statement.setInt(1, CategoryFrom1);
                    statement.setInt(2, CategoryTo1);
                    System.out.println("Dispatching the query...");
                    // Execute the query and store the result in a ResultSet
                    try (final ResultSet resultSet = statement.executeQuery()) {
                        System.out.println("CustomerNames:");
                        while (resultSet.next()) {
                            String customerName = resultSet.getString(1);
                            String Address = resultSet.getString(2);

                            writer.println(customerName);
                            writer.println(Address);

                            System.out.println(customerName);
                            System.out.println(Address);

                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
                break;
            case "16":
                System.out.println("Please enter the input text file");
                 String inputFileName = sc.nextLine();
                try (Scanner fileScanner = new Scanner(new FileReader(inputFileName))) {
                    while (fileScanner.hasNext()) {
                        // Assuming each line in the file contains data for a new record
                        String[] data = fileScanner.nextLine().split(",");
                        if (data.length == 3) {
                            // Populate the query template with data from the file
                            try (final Connection connection = DriverManager.getConnection(URL);
                                 final PreparedStatement statement =
connection.prepareStatement(QUERY_TEMPLATE_1)) {
                                statement.setString(1, data[0].trim());
                                statement.setString(2, data[1].trim());
                                statement.setInt(3, Integer.parseInt(data[2].trim()));
                                final int rowsInserted = statement.executeUpdate();
                                System.out.println(String.format("Done. %d rows inserted.", rowsInserted));
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                        } else {
                            System.out.println("Invalid data format in the input file.");
                        }
                    }
                } catch (FileNotFoundException e) {
                    System.out.println("Input file not found.");
                    e.printStackTrace();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                break;
            case "17": // Do nothing, the while loop will terminate upon the next iteration
                System.out.println("Exiting! Good-buy!");
                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
    }
}
```

Task 5. Run the program created for Tasks 4 to test its correctness as follows: To populate the database, perform 5 queries for each type (1, 2) and 10 queries for each type (3, 4, 5, 6, 7, 8) and show the contents of the affected tables after the 5 queries of each type (1, 2) are completed and after the 10 queries for each type (3, 4, 5, 6, 7, 8) are completed.

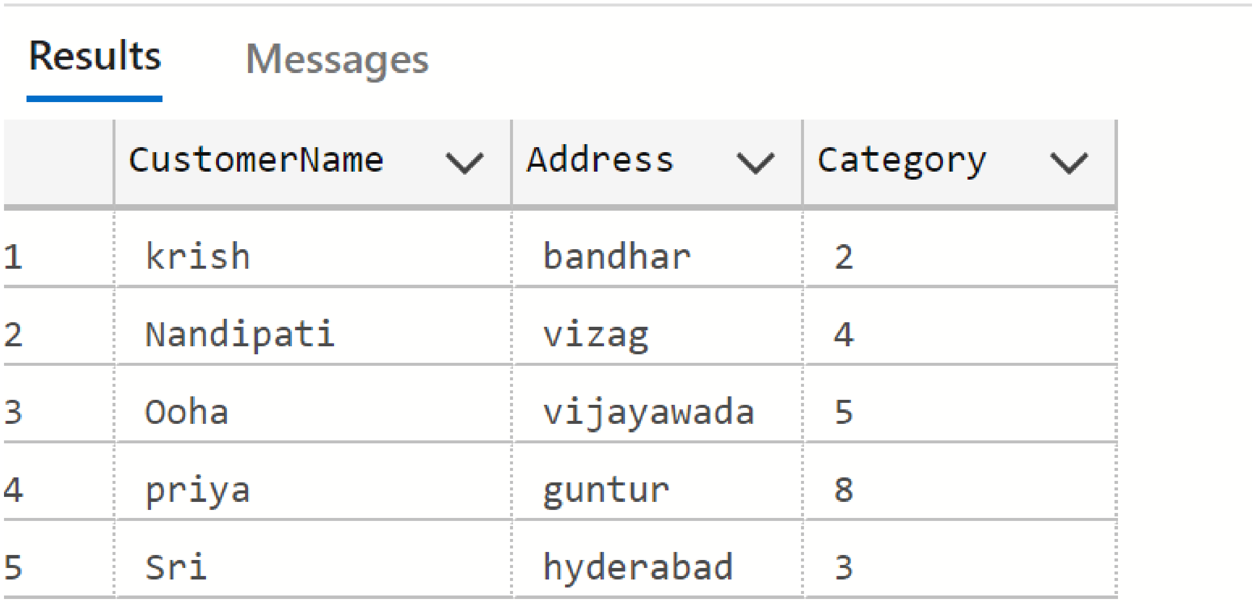
1)BEFORE INSERTION:

To populate the database, perform 5 queries for each type (1, 2)


Results Messages

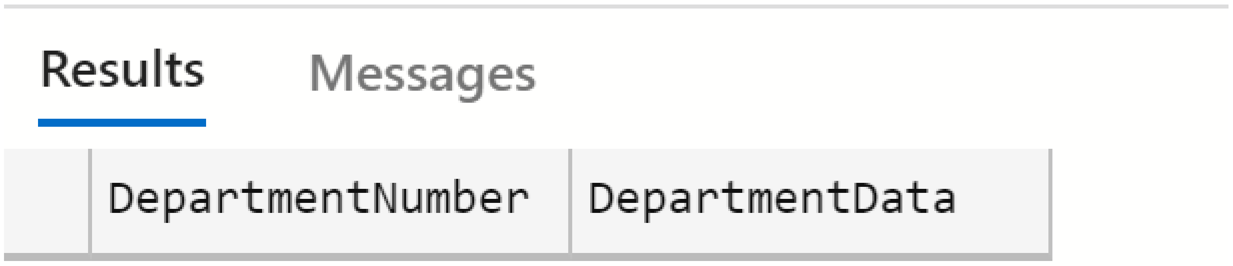
	CustomerName	Address	Category

AFTER PERFORMING FIVE QUERIES:


Results Messages

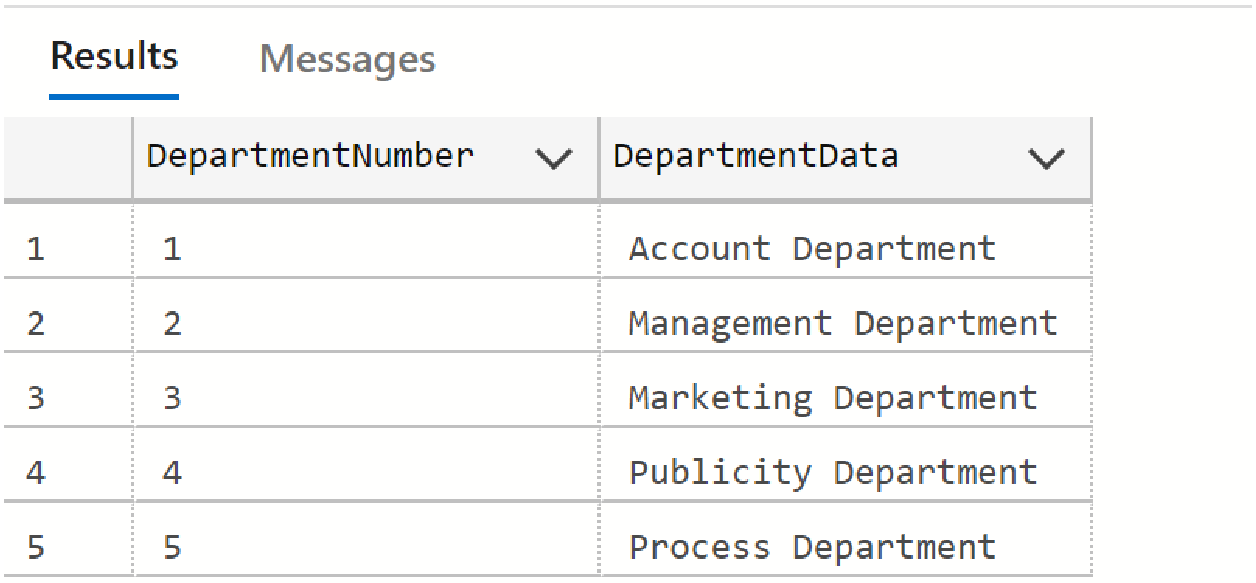
	CustomerName	Address	Category
1	krish	bandhar	2
2	Nandipati	vizag	4
3	Ooha	vijayawada	5
4	priya	guntur	8
5	Sri	hyderabad	3

2)BEFORE INSERTION:


Results Messages

	DepartmentNumber	DepartmentData

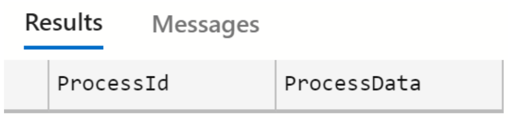
AFTER INSERTING FIVE QUERIES:


Results Messages

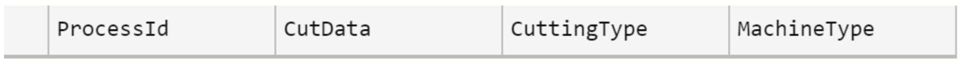
	DepartmentNumber	DepartmentData
1	1	Account Department
2	2	Management Department
3	3	Marketing Department
4	4	Publicity Department
5	5	Process Department

10 queries for each type (3, 4, 5, 6, 7, 8):

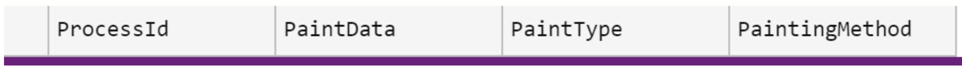
3)BEFORE PERFORMING ANY QUERIES:



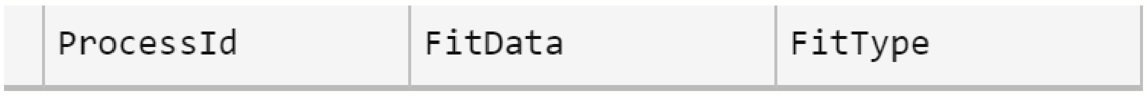
	ProcessId	ProcessData



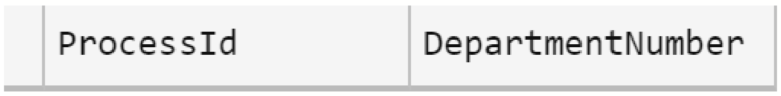
	ProcessId	CutData	CuttingType	MachineType



	ProcessId	PaintData	PaintType	PaintingMethod



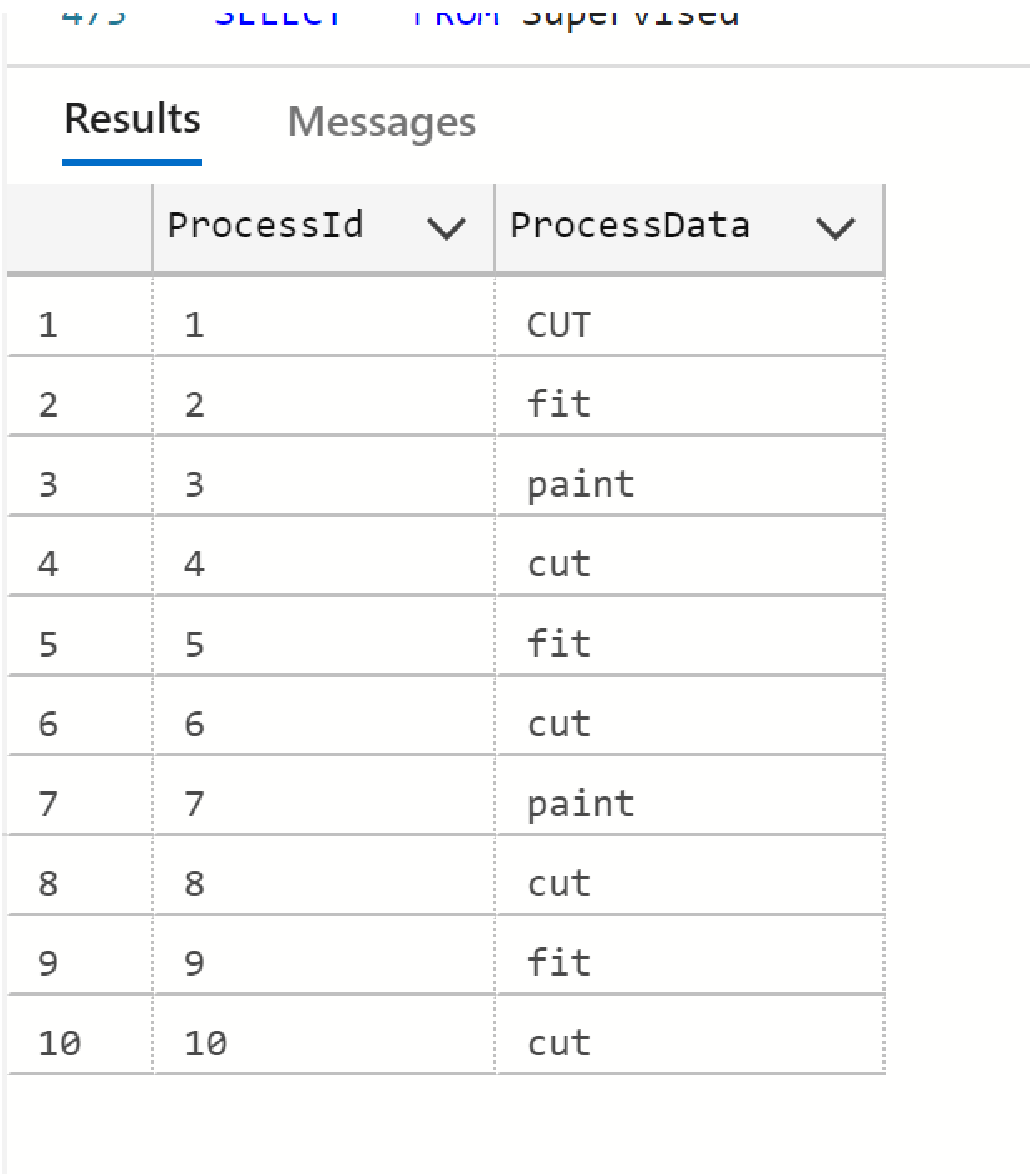
	ProcessId	FitData	FitType



	ProcessId	DepartmentNumber

AFTER INSERTING 10 QUERIES:

Processes Table After Insertion:

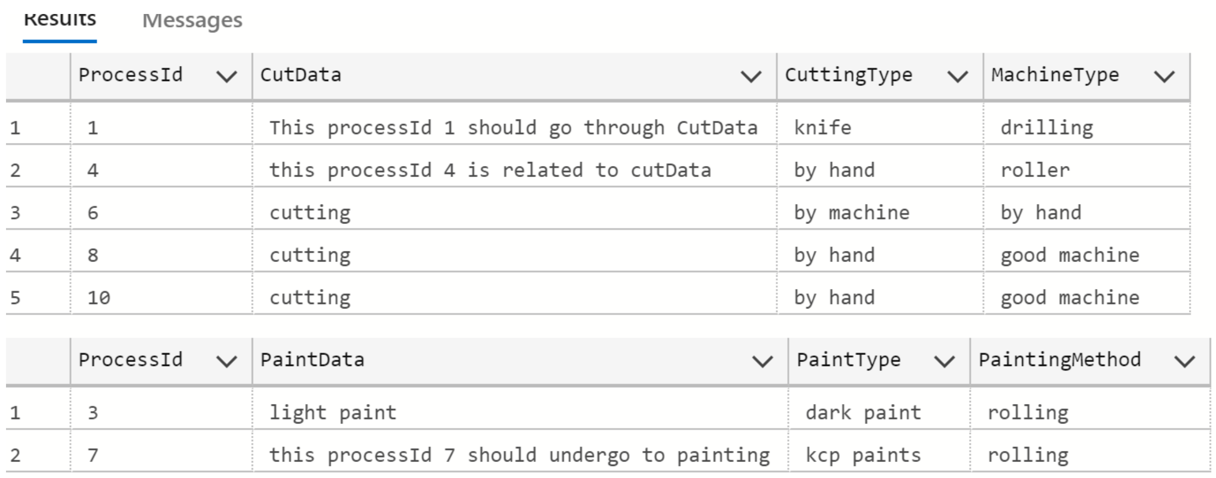


```
SELECT      FROM Supervised
```

Results   Messages

	ProcessId	ProcessData
1	1	CUT
2	2	fit
3	3	paint
4	4	cut
5	5	fit
6	6	cut
7	7	paint
8	8	cut
9	9	fit
10	10	cut

CUT TABLE & PAINT TABLE :

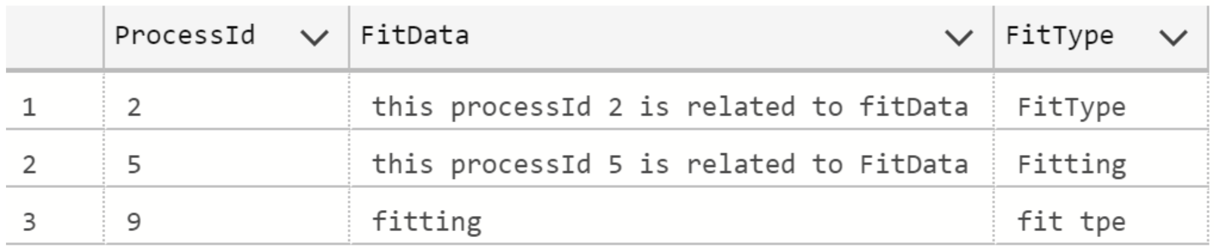


Results Messages

	ProcessId	CutData	CuttingType	MachineType
1	1	This processId 1 should go through CutData	knife	drilling
2	4	this processId 4 is related to cutData	by hand	roller
3	6	cutting	by machine	by hand
4	8	cutting	by hand	good machine
5	10	cutting	by hand	good machine

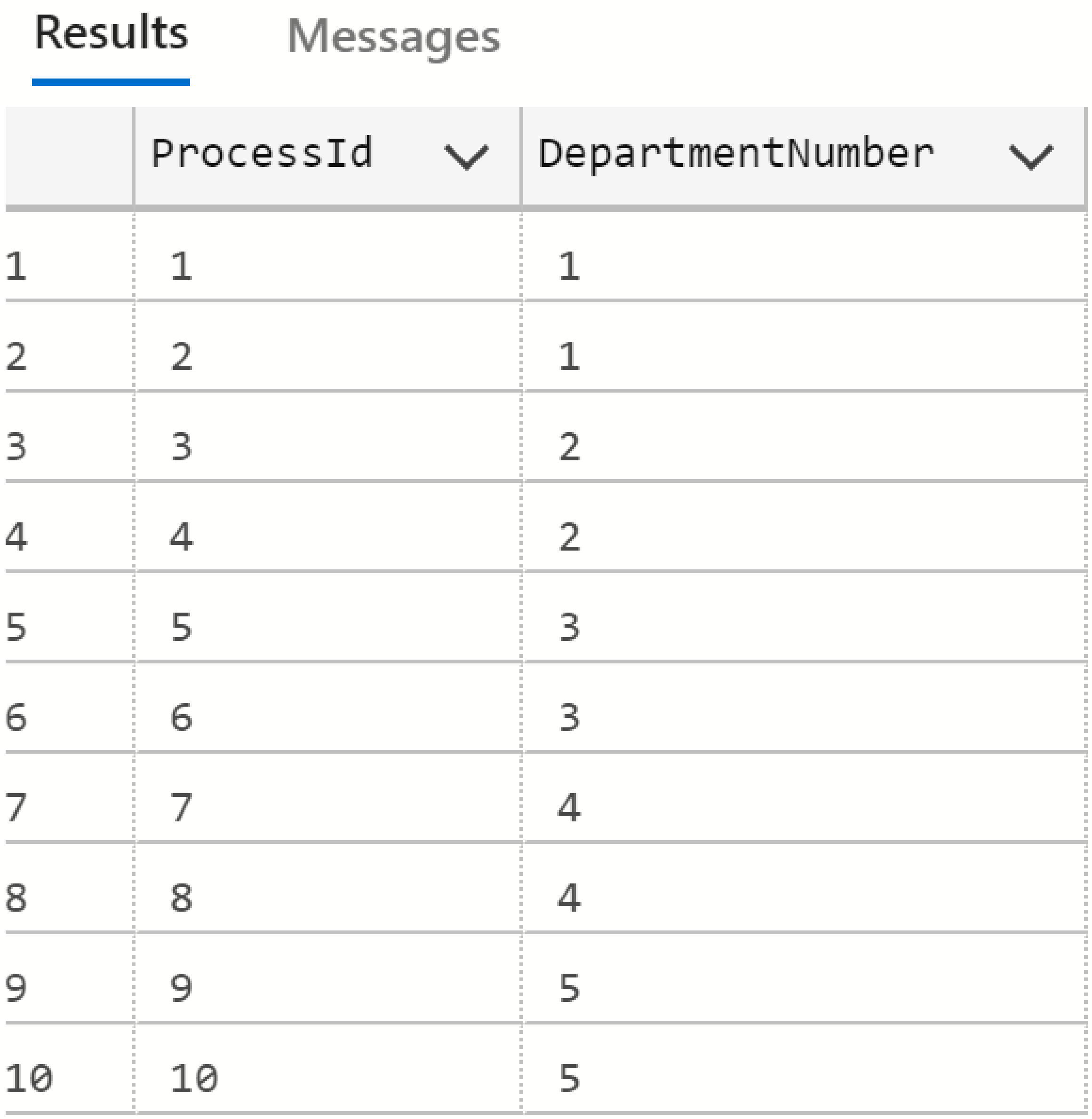
	ProcessId	PaintData	PaintType	PaintingMethod
1	3	light paint	dark paint	rolling
2	7	this processId 7 should undergo to painting	kcp paints	rolling

FIT TABLE:



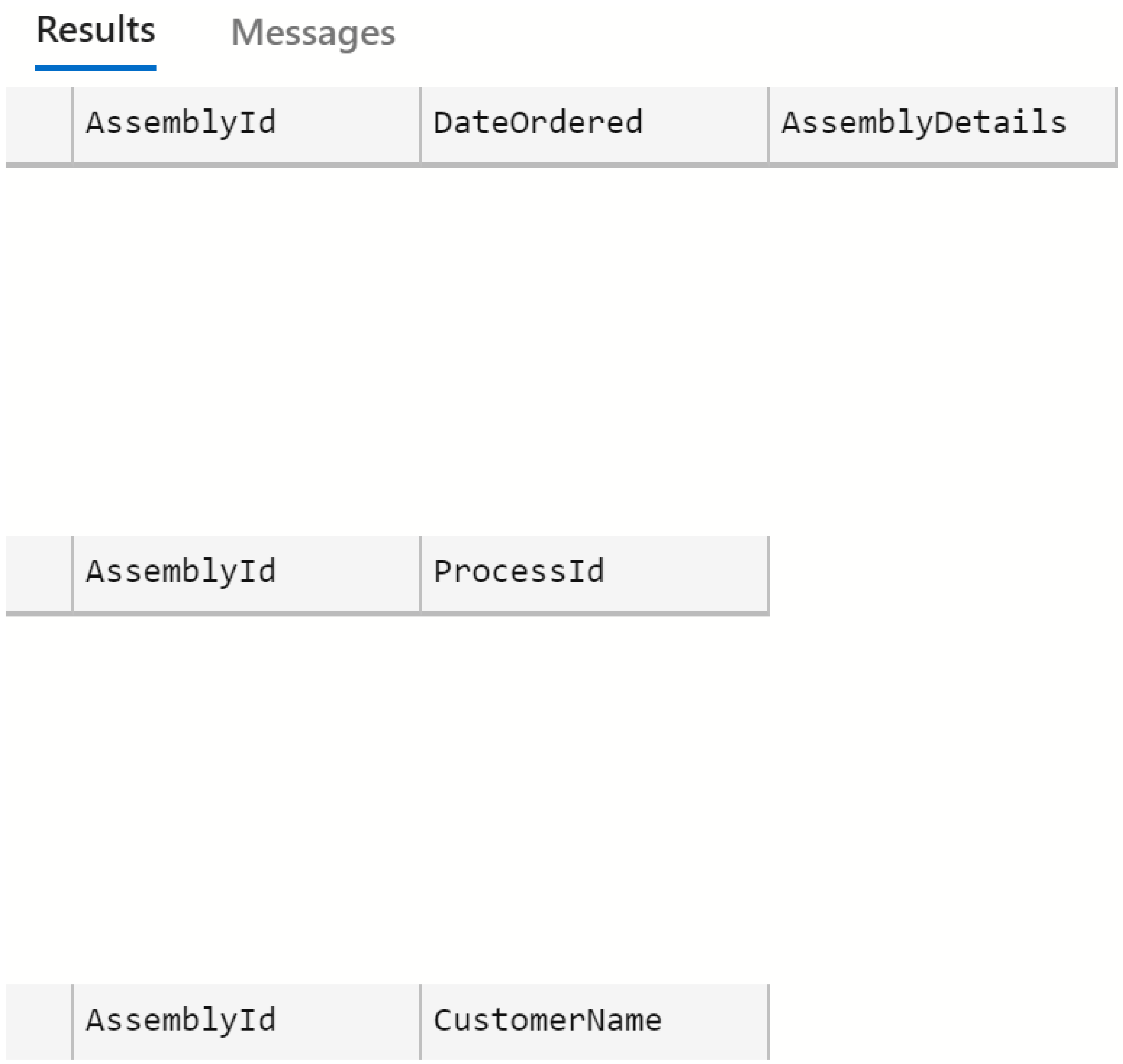
	ProcessId	FitData	FitType
1	2	this processId 2 is related to fitData	FitType
2	5	this processId 5 is related to FitData	Fitting
3	9	fitting	fit tpe

SUPERVISED TABLE:



	ProcessId	DepartmentNumber
1	1	1
2	2	1
3	3	2
4	4	2
5	5	3
6	6	3
7	7	4
8	8	4
9	9	5
10	10	5

4)BEFORE INSERTION:



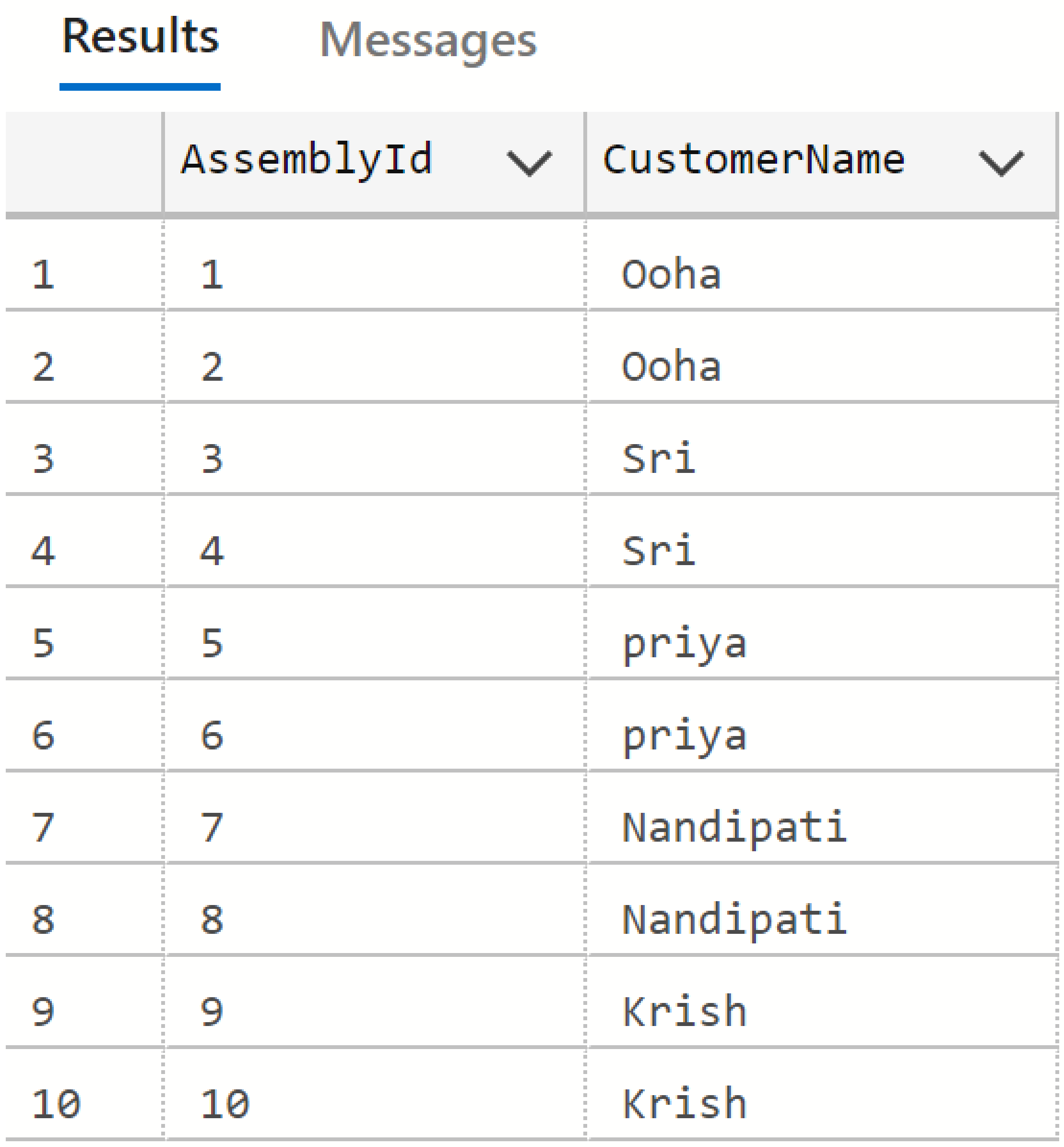
Results Messages

	AssemblyId	DateOrdered	AssemblyDetails

	AssemblyId	ProcessId

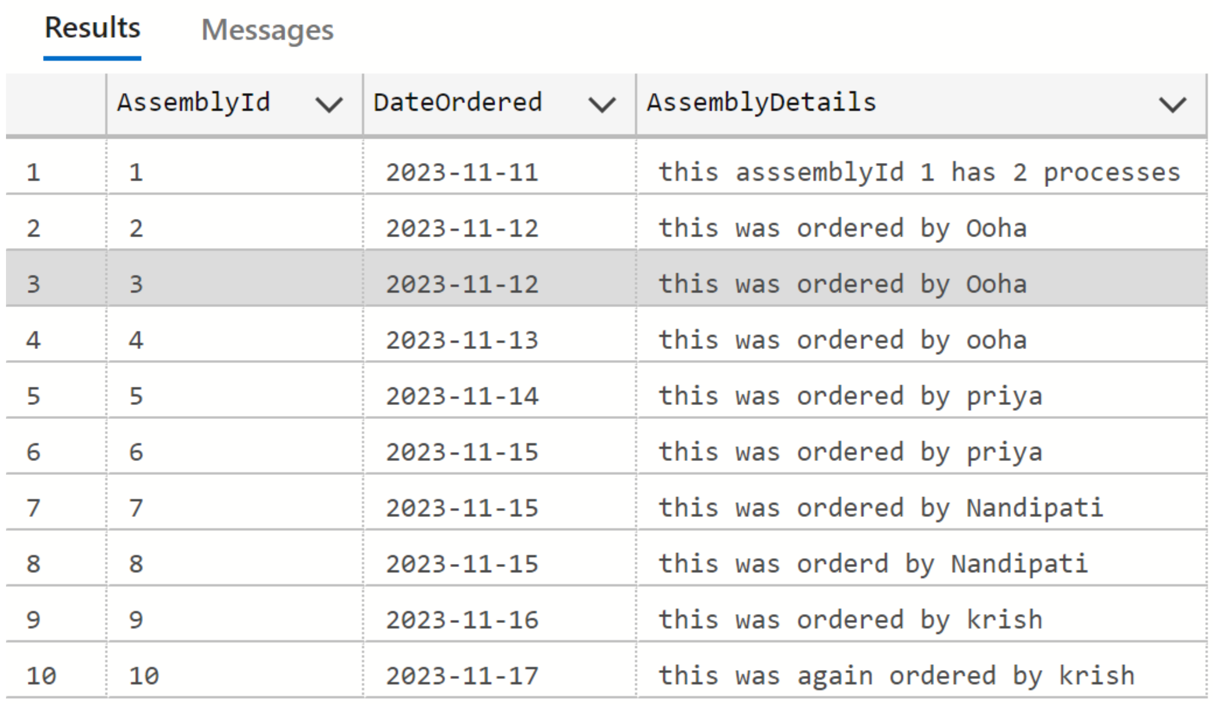
	AssemblyId	CustomerName

AFTER INSERTING:
ORDERS TABLE:



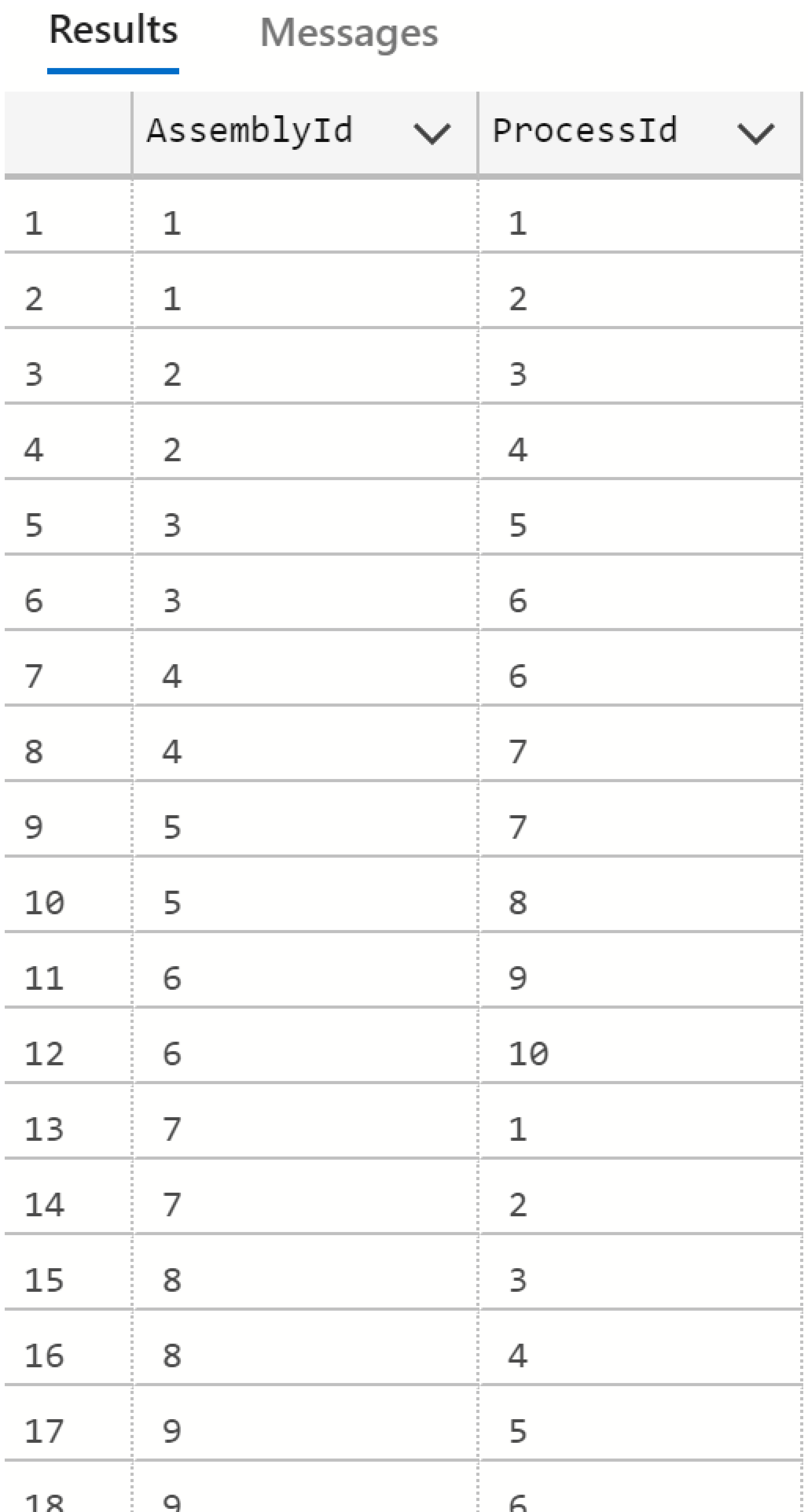
	AssemblyId	CustomerName
1	1	Ooha
2	2	Ooha
3	3	Sri
4	4	Sri
5	5	priya
6	6	priya
7	7	Nandipati
8	8	Nandipati
9	9	Krish
10	10	Krish

ASSEMBLIES TABLE:


Results Messages

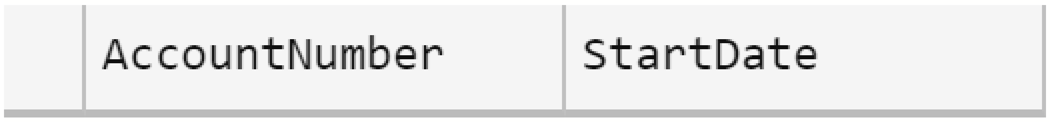
	AssemblyId	DateOrdered	AssemblyDetails
1	1	2023-11-11	this asssemblyId 1 has 2 processes
2	2	2023-11-12	this was ordered by Ooha
3	3	2023-11-12	this was ordered by Ooha
4	4	2023-11-13	this was ordered by ooha
5	5	2023-11-14	this was ordered by priya
6	6	2023-11-15	this was ordered by priya
7	7	2023-11-15	this was ordered by Nandipati
8	8	2023-11-15	this was orderd by Nandipati
9	9	2023-11-16	this was ordered by krish
10	10	2023-11-17	this was again ordered by krish

MANUFACTURED TABLE:

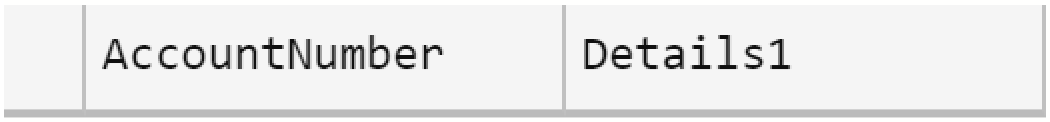


	AssemblyId	ProcessId
1	1	1
2	1	2
3	2	3
4	2	4
5	3	5
6	3	6
7	4	6
8	4	7
9	5	7
10	5	8
11	6	9
12	6	10
13	7	1
14	7	2
15	8	3
16	8	4
17	9	5
18	9	6

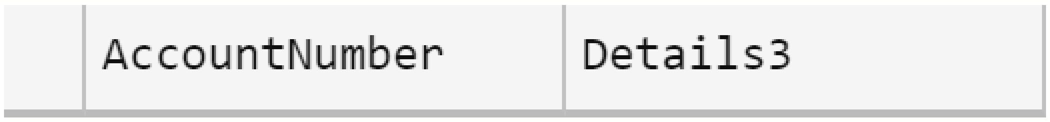
5)Before Any Insertion:



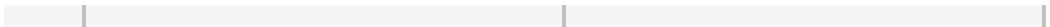
	AccountNumber	StartDate



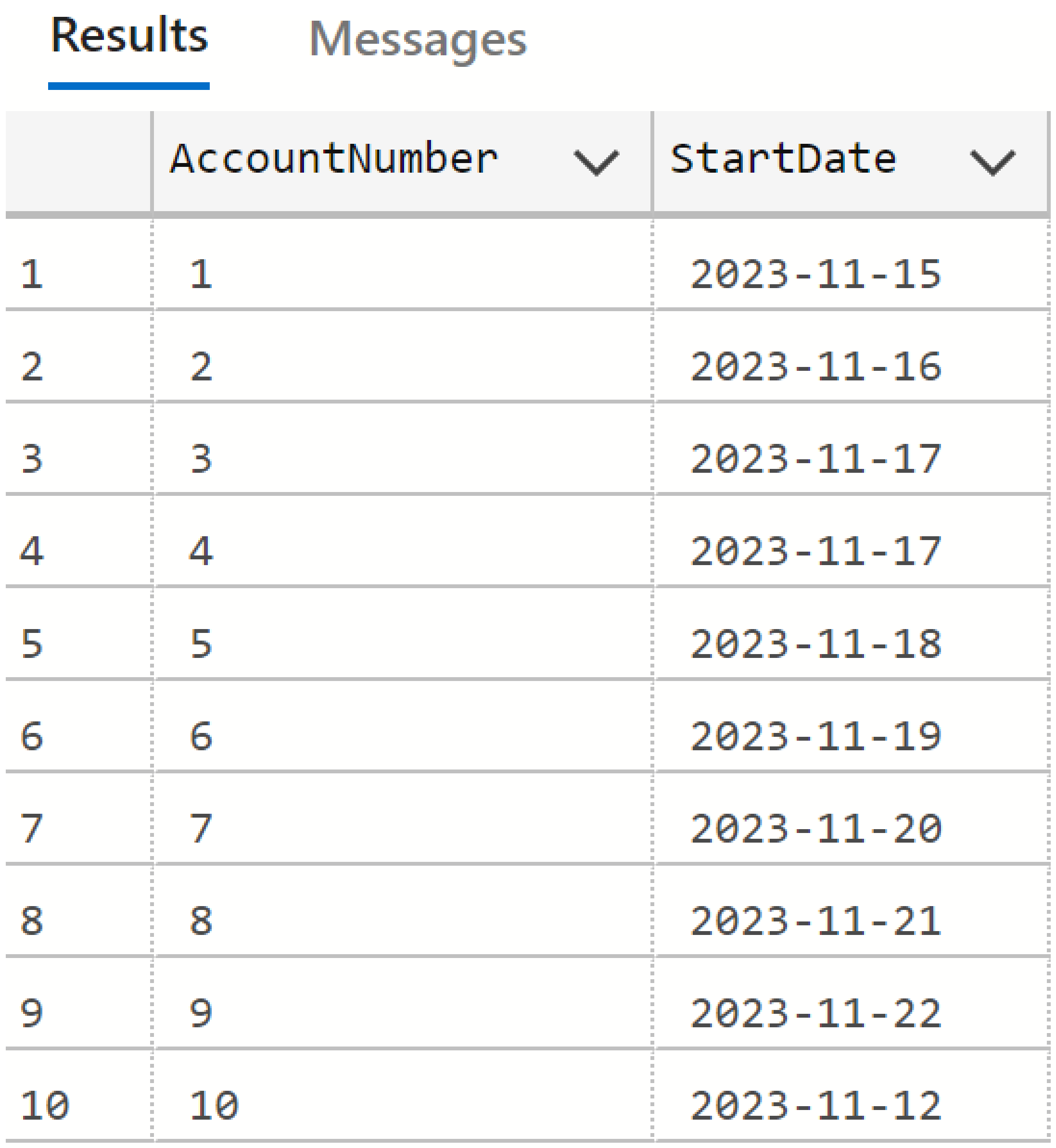
	AccountNumber	Details1



	AccountNumber	Details3

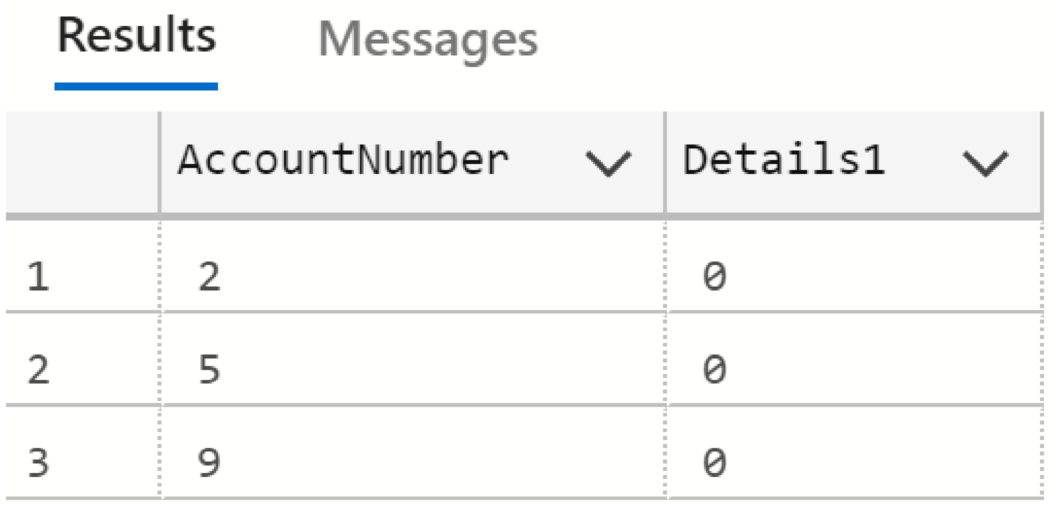


After Insertion:
Account Table:

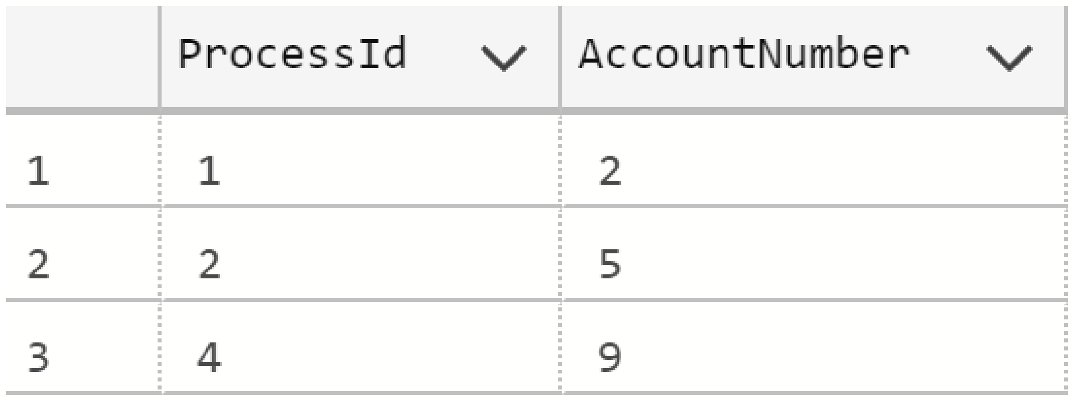


	AccountNumber	StartDate
1	1	2023-11-15
2	2	2023-11-16
3	3	2023-11-17
4	4	2023-11-17
5	5	2023-11-18
6	6	2023-11-19
7	7	2023-11-20
8	8	2023-11-21
9	9	2023-11-22
10	10	2023-11-12

ProcessAccount and ProcessMaintain Table:

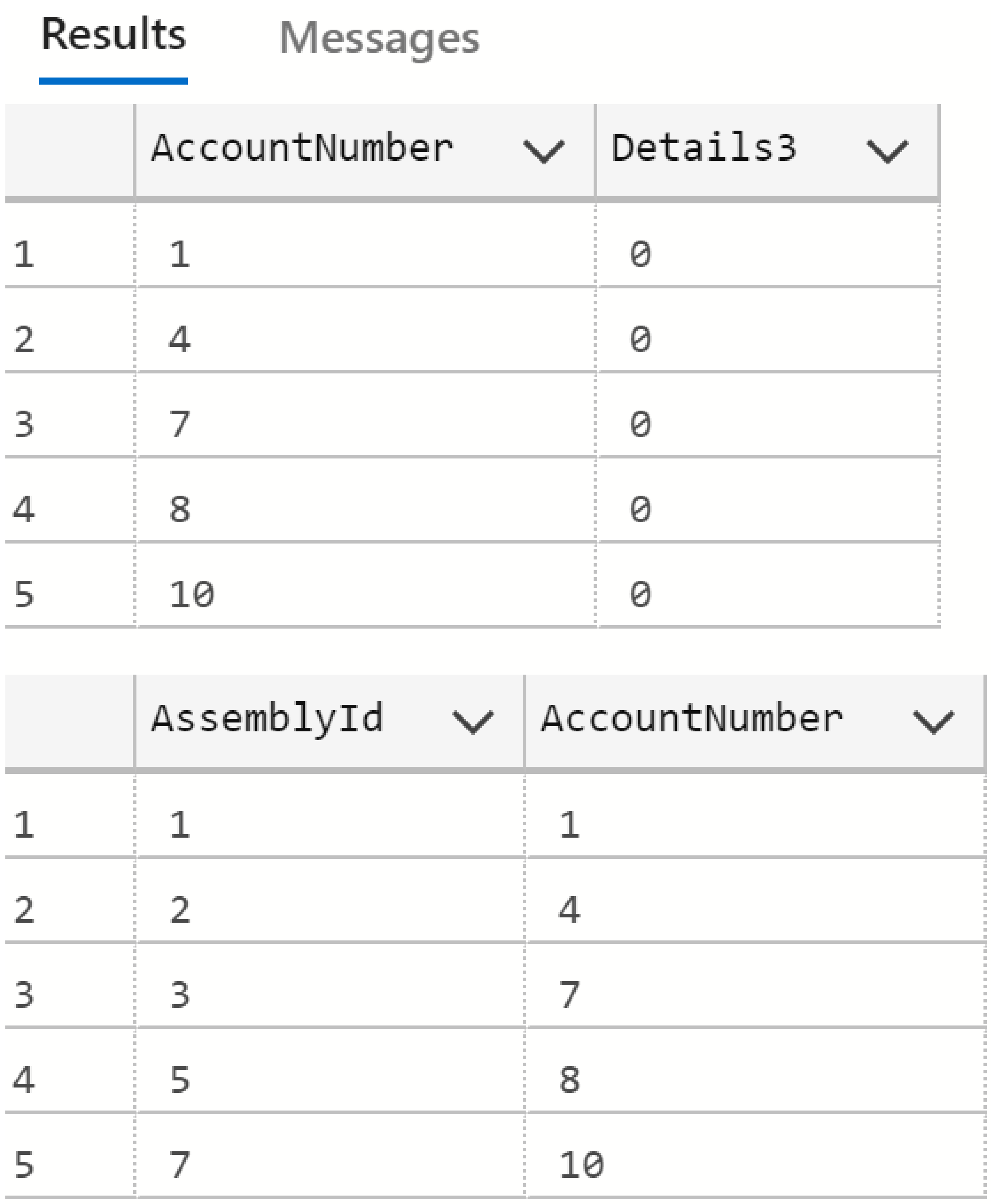


	AccountNumber	Details1
1	2	0
2	5	0
3	9	0



	ProcessId	AccountNumber
1	1	2
2	2	5
3	4	9

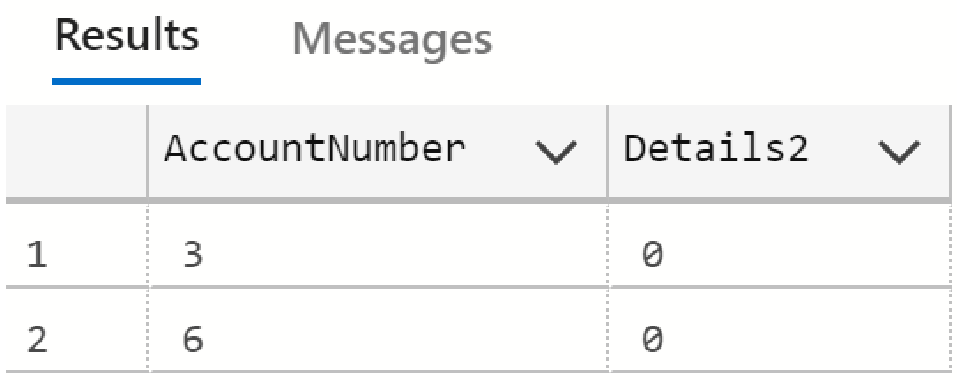
AssemblyAccount & AssemblyMaintain Table:


Results Messages

	AccountNumber	Details3
1	1	0
2	4	0
3	7	0
4	8	0
5	10	0

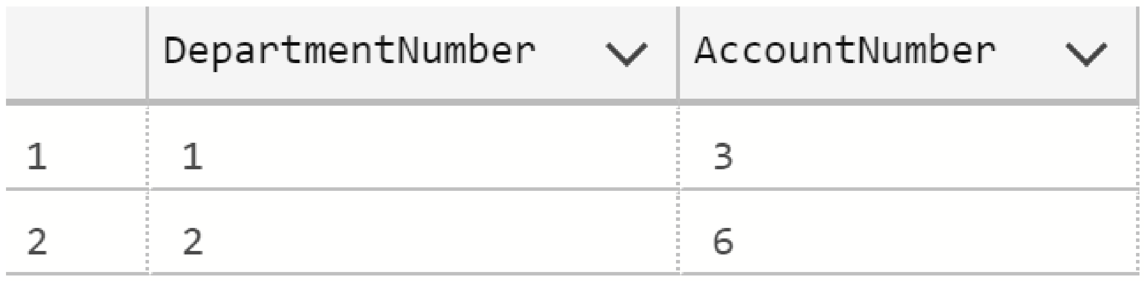
	AssemblyId	AccountNumber
1	1	1
2	2	4
3	3	7
4	5	8
5	7	10

DepartmentAccount and DepartmentMaintain Table:



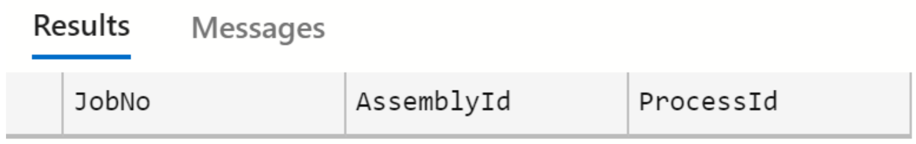
Results Messages

	AccountNumber	Details2
1	3	0
2	6	0



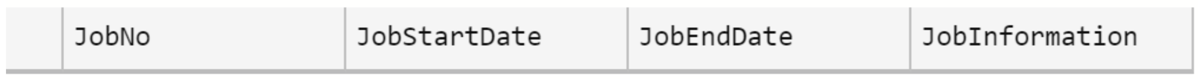
	DepartmentNumber	AccountNumber
1	1	3
2	2	6

6)Before Inserting:



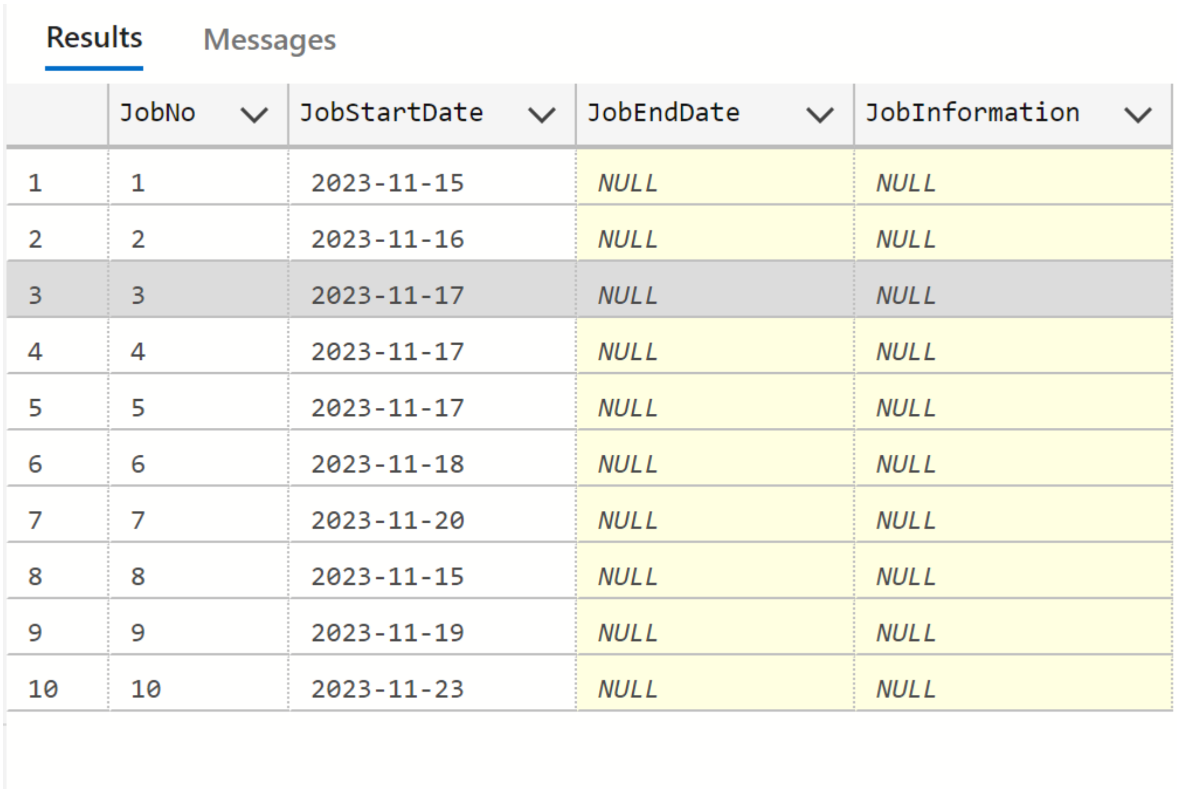
Results Messages

	JobNo	AssemblyId	ProcessId



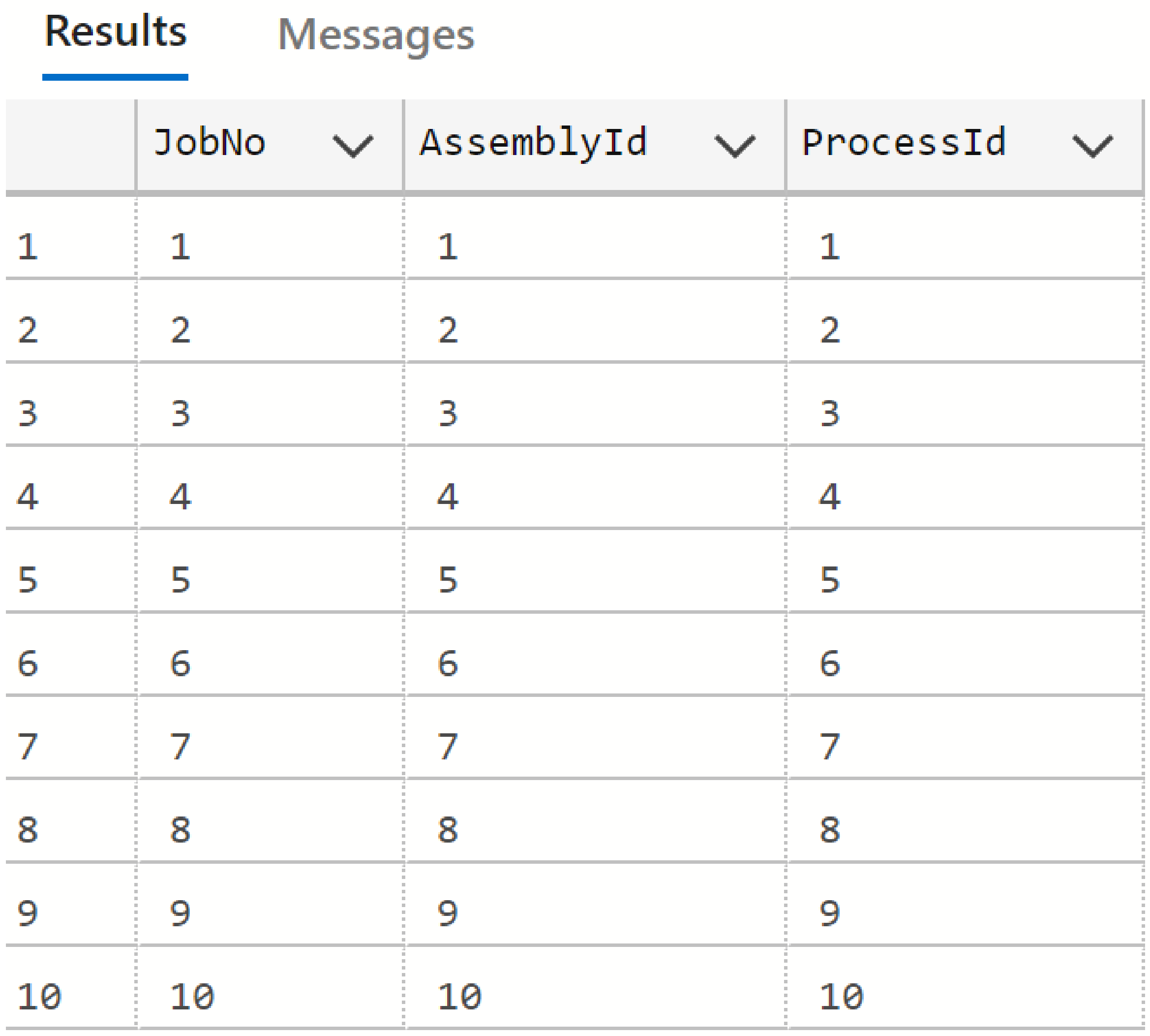
	JobNo	JobStartDate	JobEndDate	JobInformation

After Inserting:
Job Table:


Results Messages

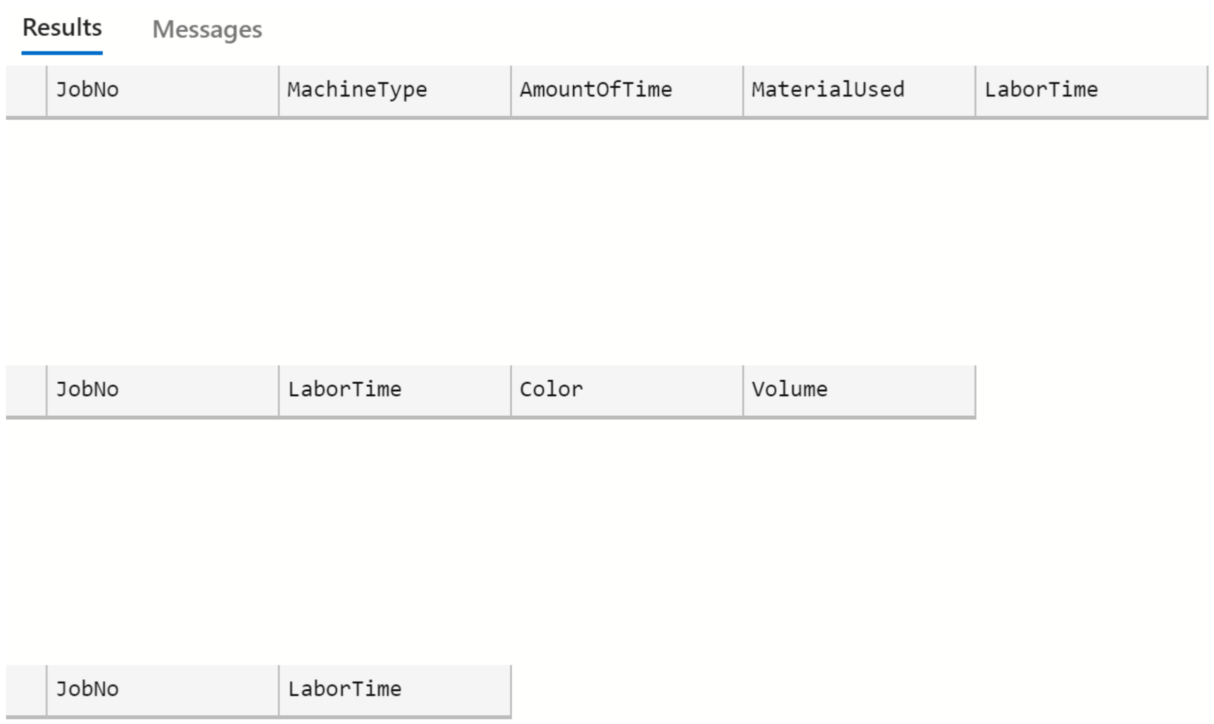
	JobNo	JobStartDate	JobEndDate	JobInformation
1	1	2023-11-15	NULL	NULL
2	2	2023-11-16	NULL	NULL
3	3	2023-11-17	NULL	NULL
4	4	2023-11-17	NULL	NULL
5	5	2023-11-17	NULL	NULL
6	6	2023-11-18	NULL	NULL
7	7	2023-11-20	NULL	NULL
8	8	2023-11-15	NULL	NULL
9	9	2023-11-19	NULL	NULL
10	10	2023-11-23	NULL	NULL

Assign Table:



	JobNo	AssemblyId	ProcessId
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	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

7)Before Inserting:



Results Messages

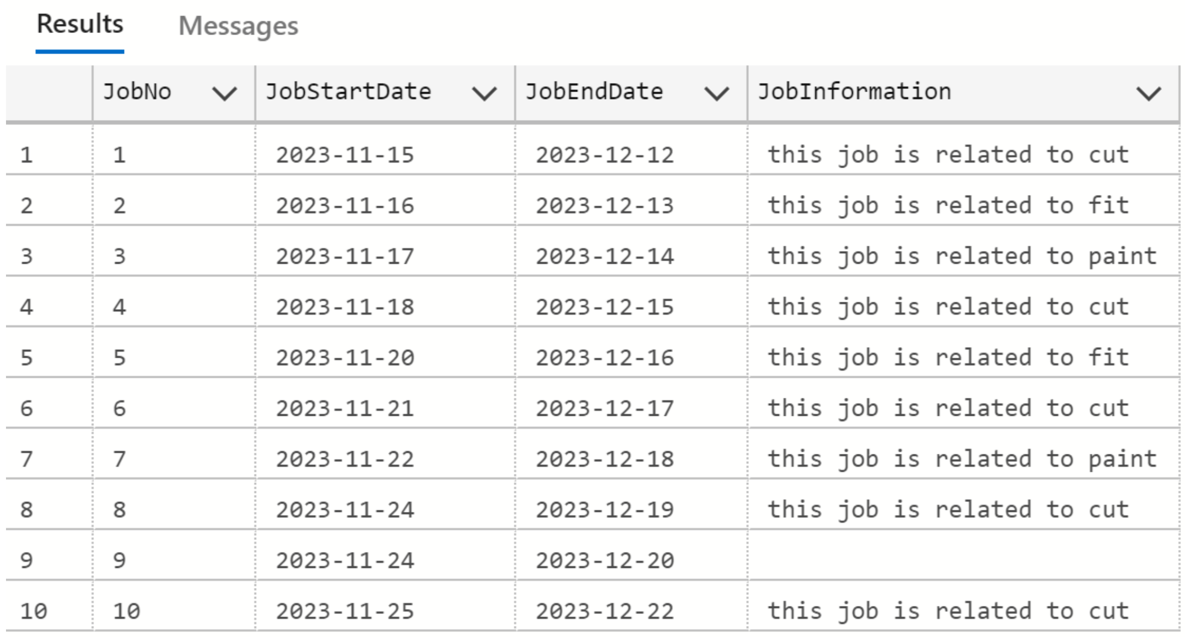
JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime

JobNo	LaborTime	Color	Volume

JobNo	LaborTime

After Inserting:

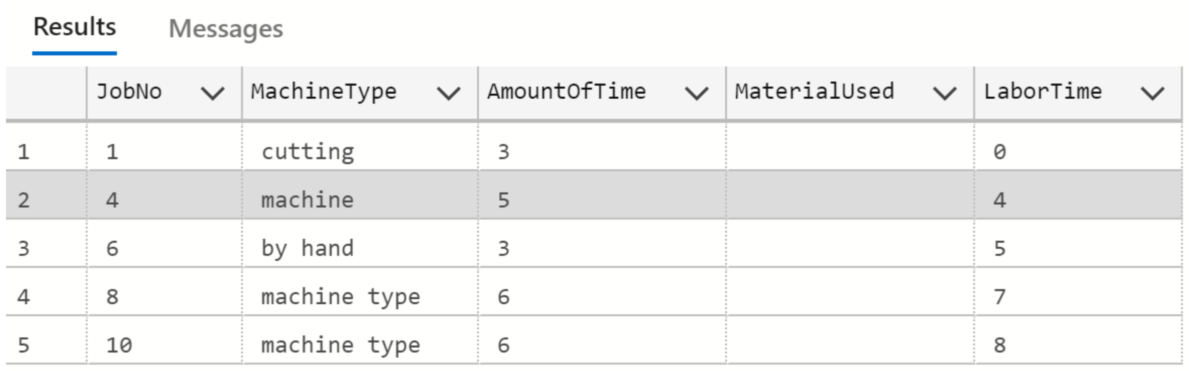
Job Table:



Results Messages

	JobNo	JobStartDate	JobEndDate	JobInformation
1	1	2023-11-15	2023-12-12	this job is related to cut
2	2	2023-11-16	2023-12-13	this job is related to fit
3	3	2023-11-17	2023-12-14	this job is related to paint
4	4	2023-11-18	2023-12-15	this job is related to cut
5	5	2023-11-20	2023-12-16	this job is related to fit
6	6	2023-11-21	2023-12-17	this job is related to cut
7	7	2023-11-22	2023-12-18	this job is related to paint
8	8	2023-11-24	2023-12-19	this job is related to cut
9	9	2023-11-24	2023-12-20	
10	10	2023-11-25	2023-12-22	this job is related to cut

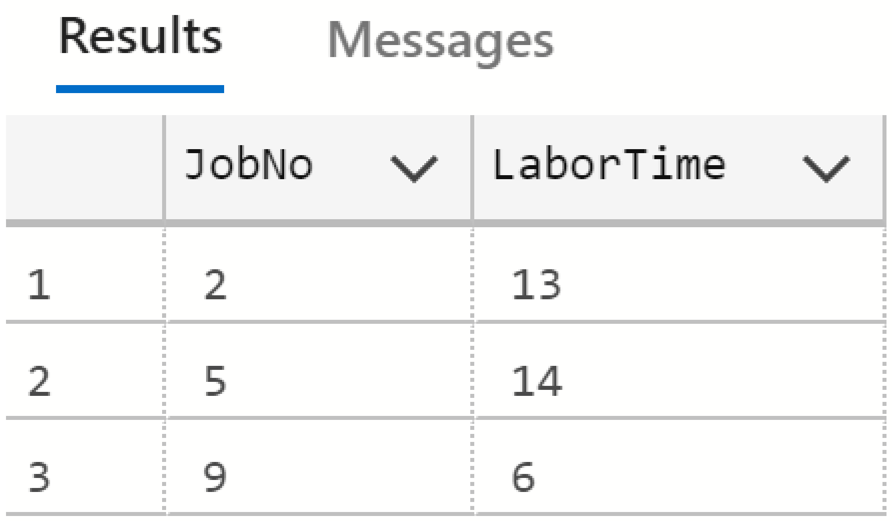
CutJob Table:



Results Messages

	JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime
1	1	cutting	3		0
2	4	machine	5		4
3	6	by hand	3		5
4	8	machine type	6		7
5	10	machine type	6		8

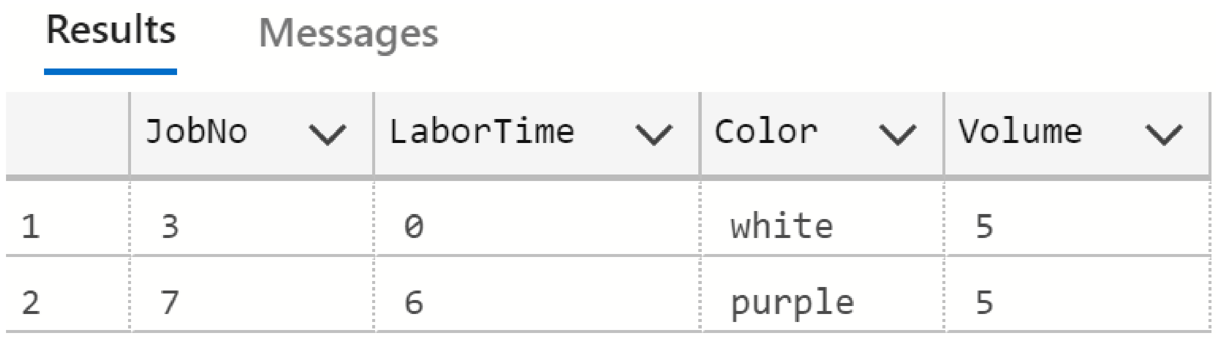
FitJob:



Results Messages

	JobNo	LaborTime
1	2	13
2	5	14
3	9	6

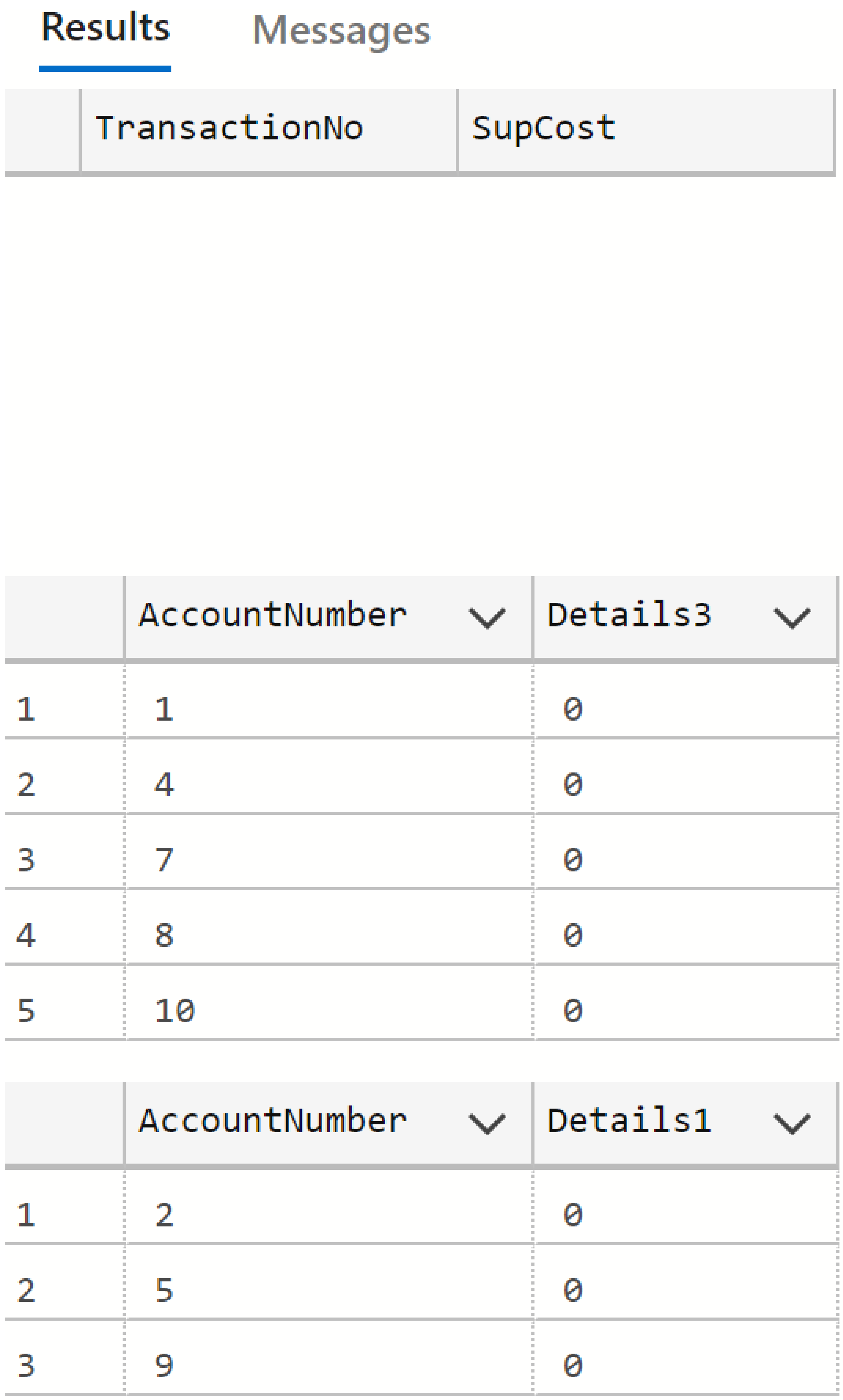
PaintJob:



Results Messages

	JobNo	LaborTime	Color	Volume
1	3	0	white	5
2	7	6	purple	5

8)Before Insertion: Transactions&Assemblies&Processes acc

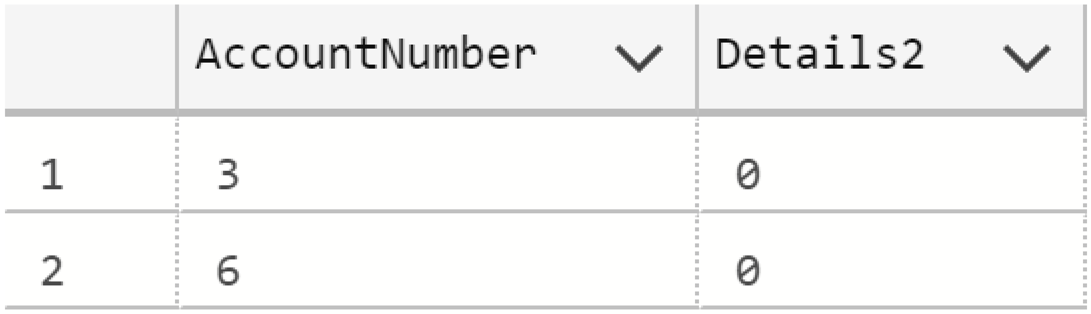


	TransactionNo	SupCost

	AccountNumber	Details3
1	1	0
2	4	0
3	7	0
4	8	0
5	10	0

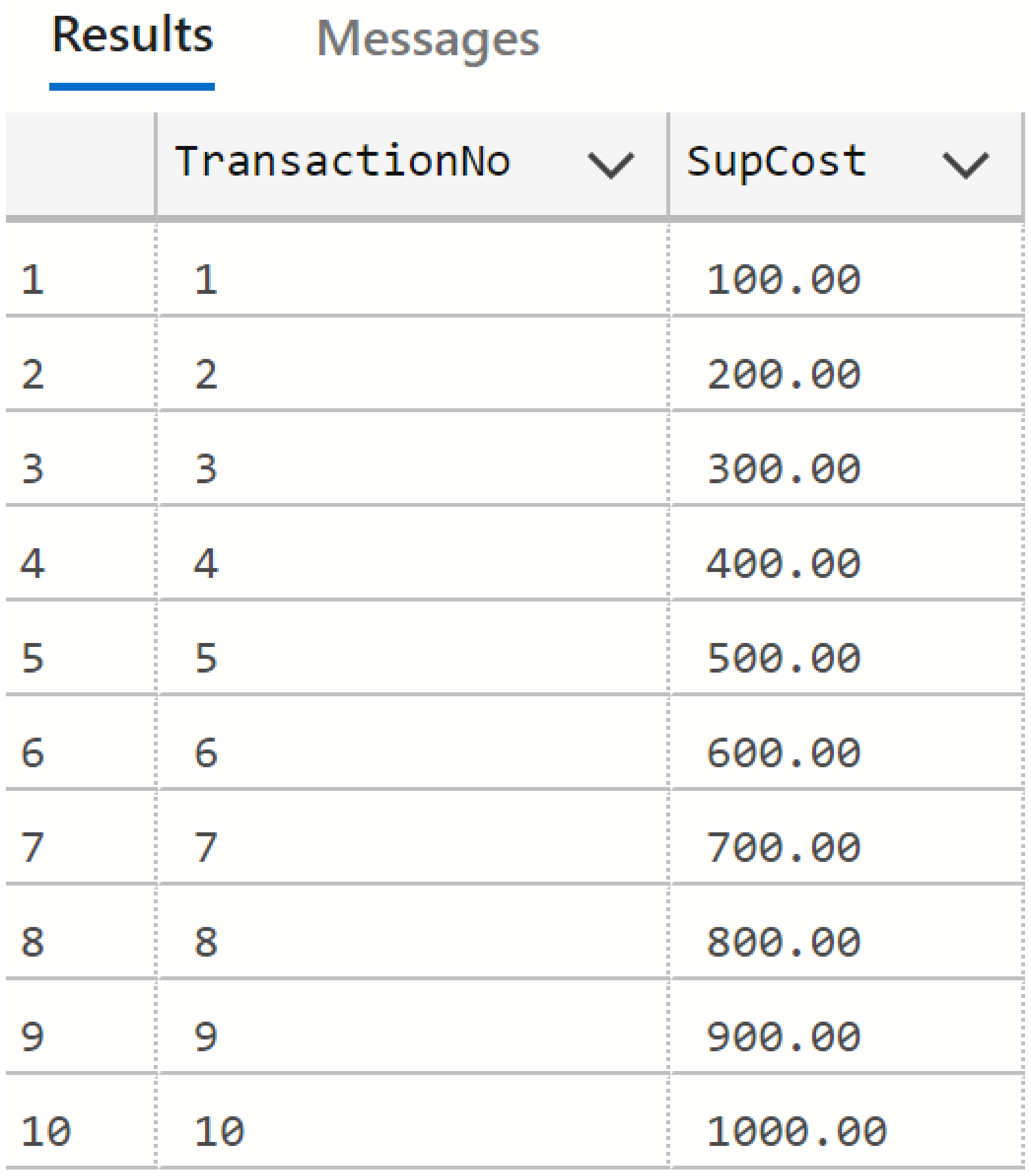
	AccountNumber	Details1
1	2	0
2	5	0
3	9	0

DepartmentAccount:



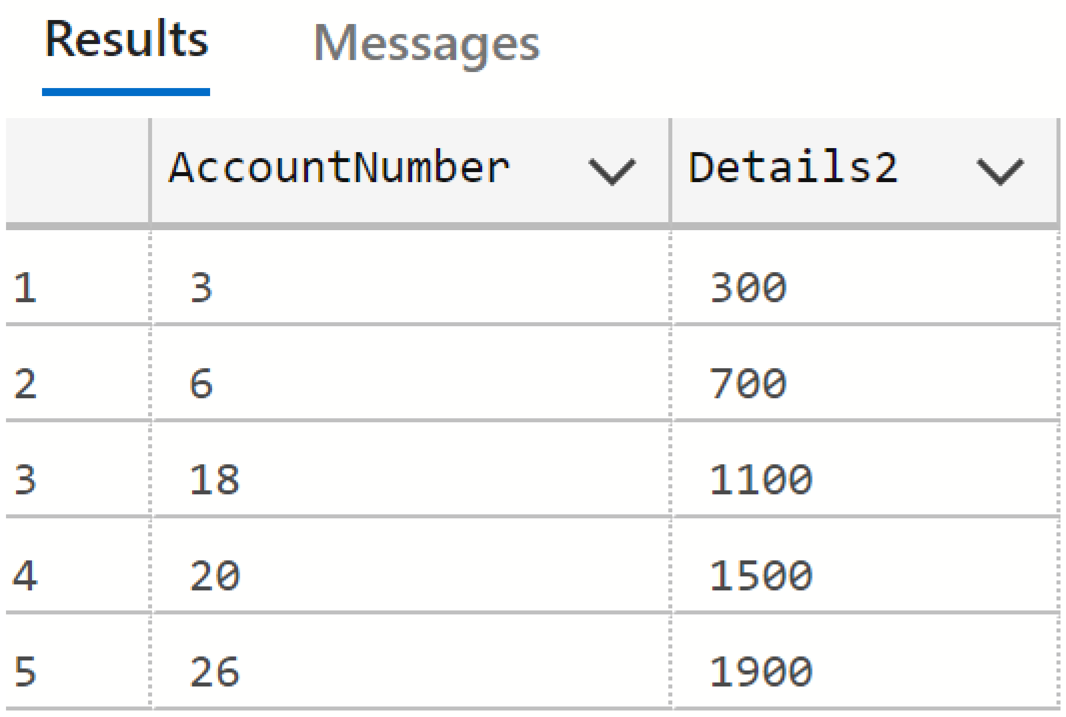
	AccountNumber	Details2
1	3	0
2	6	0

AFTER INSERTION:
Transaction Table:



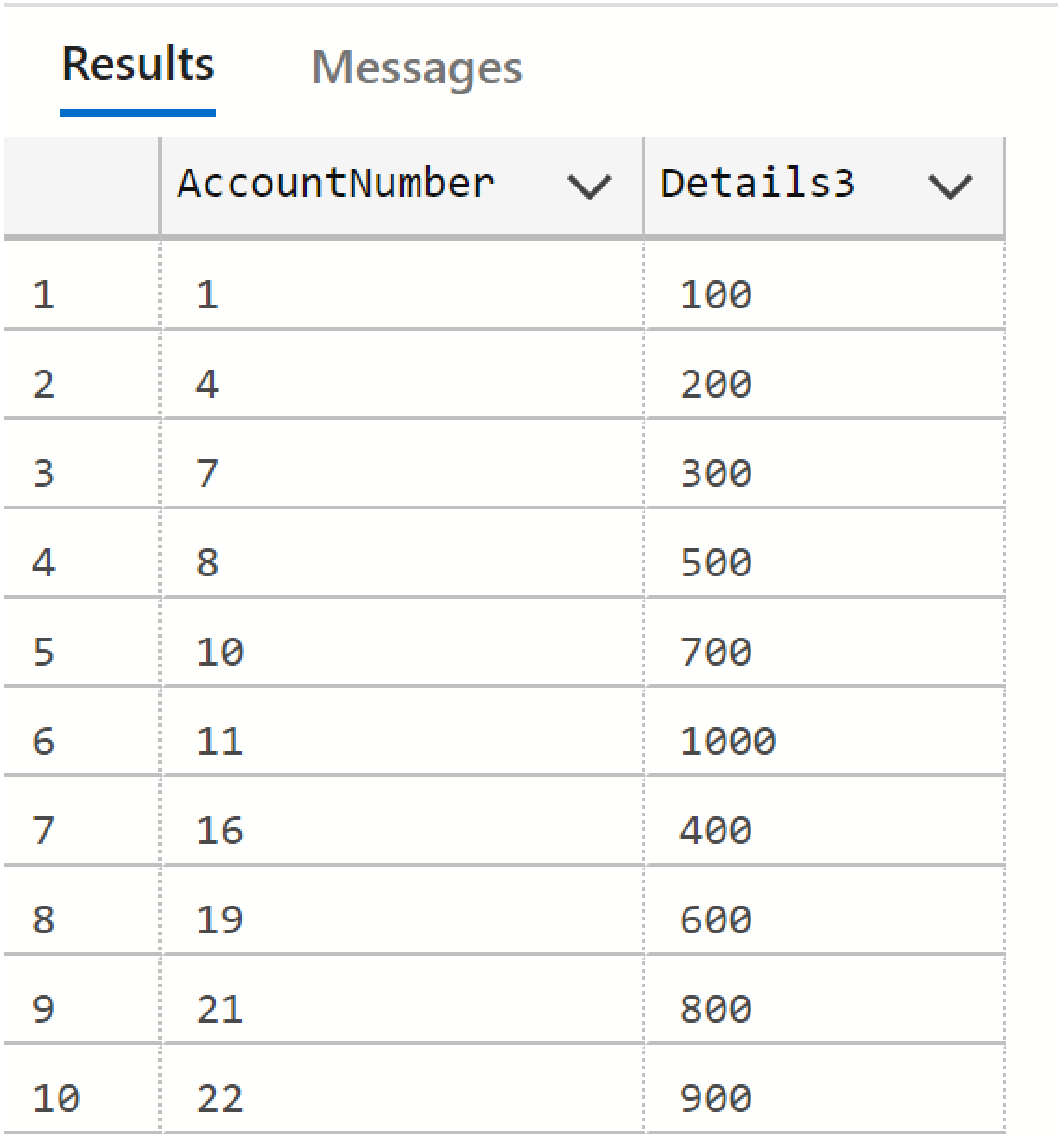
	TransactionNo	SupCost
1	1	100.00
2	2	200.00
3	3	300.00
4	4	400.00
5	5	500.00
6	6	600.00
7	7	700.00
8	8	800.00
9	9	900.00
10	10	1000.00

DepartmentAccount Table:


Results Messages

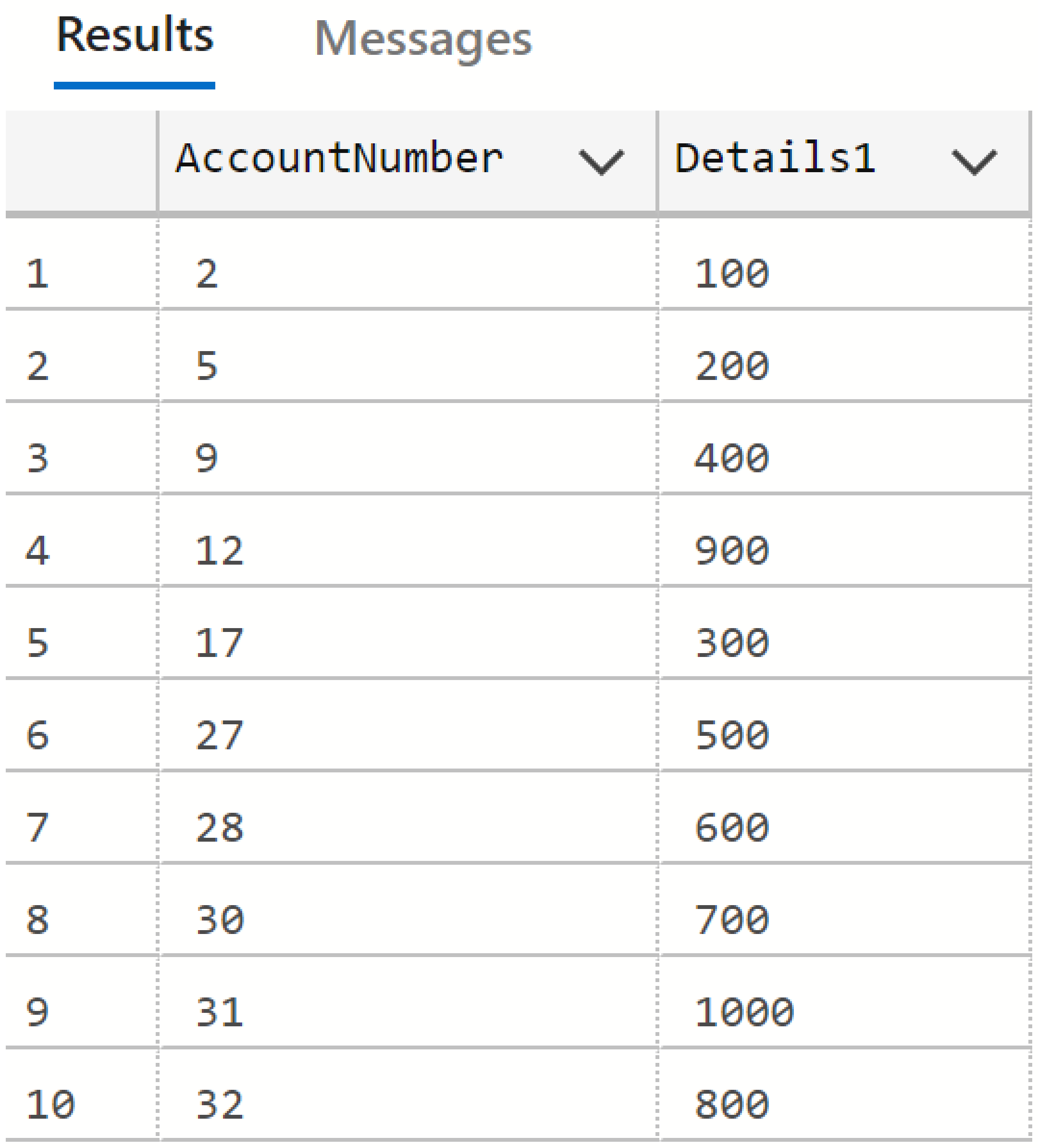
	AccountNumber	Details2
1	3	300
2	6	700
3	18	1100
4	20	1500
5	26	1900

AssembliesAccount Table:



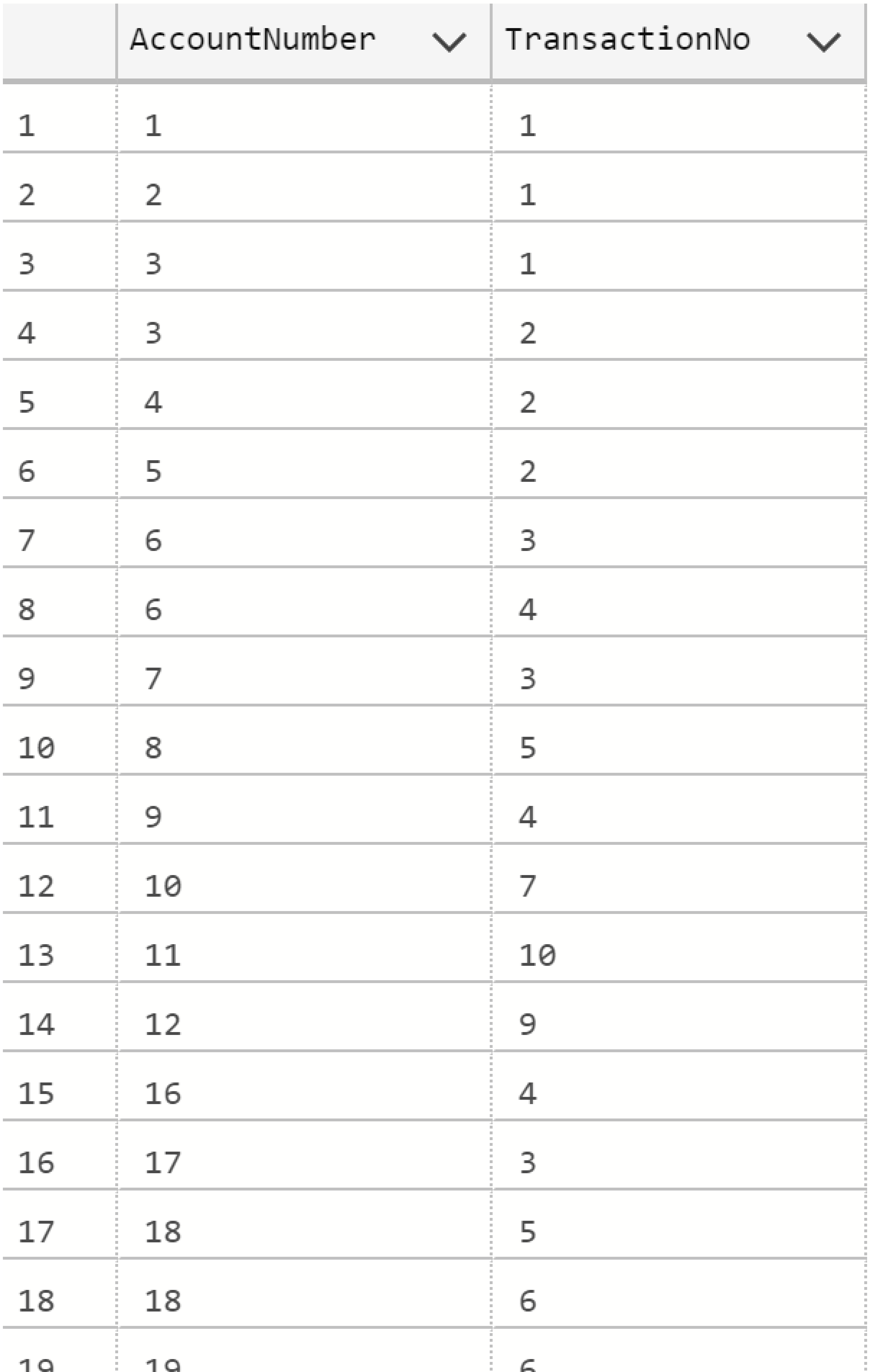
	AccountNumber	Details3
1	1	100
2	4	200
3	7	300
4	8	500
5	10	700
6	11	1000
7	16	400
8	19	600
9	21	800
10	22	900

ProcessAccount Table:

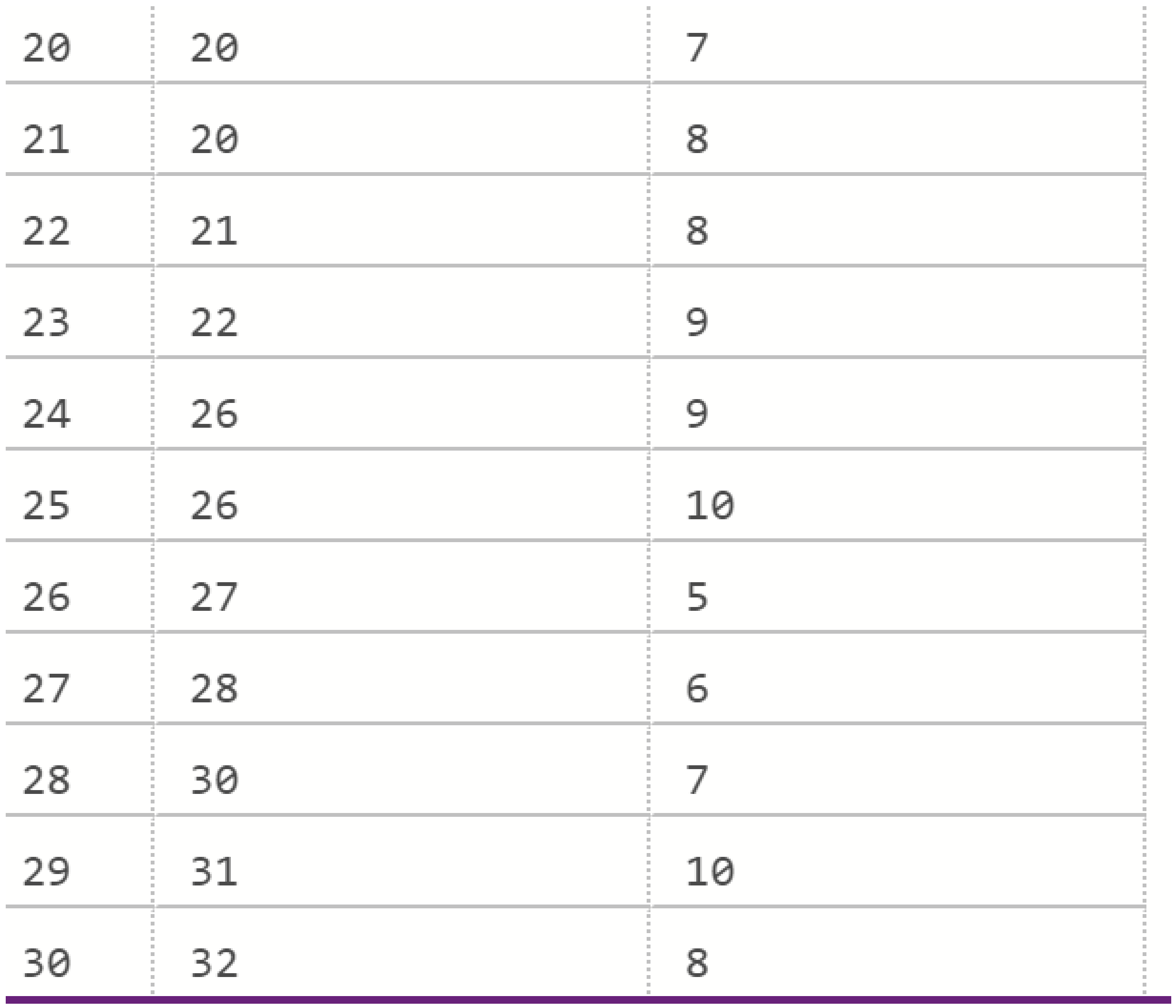


	AccountNumber	Details1
1	2	100
2	5	200
3	9	400
4	12	900
5	17	300
6	27	500
7	28	600
8	30	700
9	31	1000
10	32	800

Updates Table:



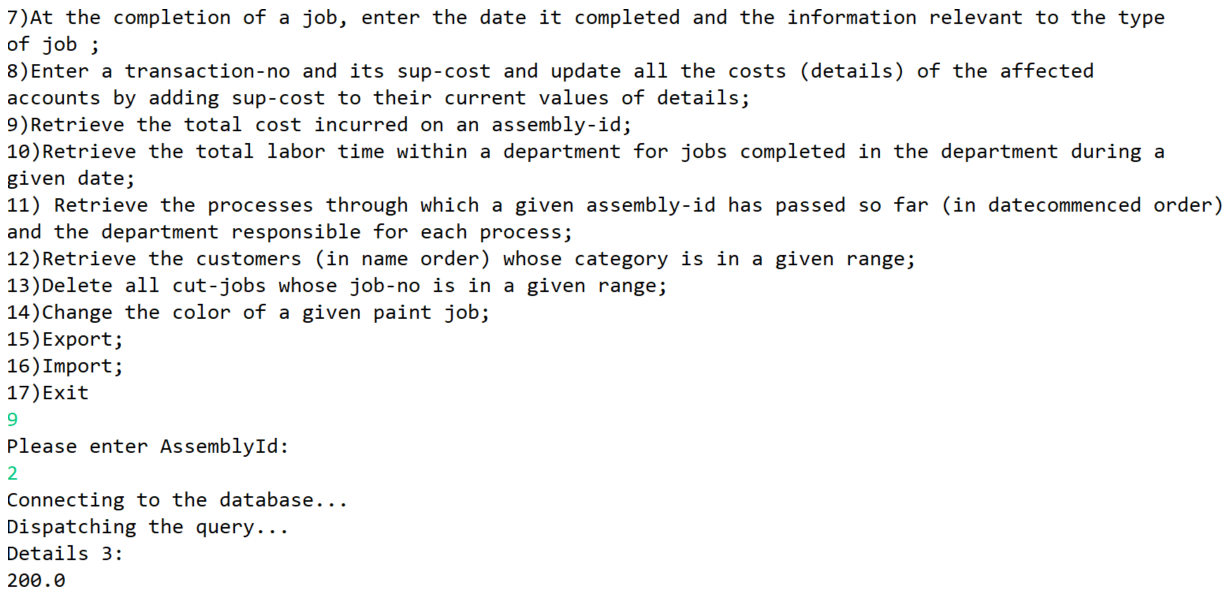
	AccountNumber	TransactionNo
1	1	1
2	2	1
3	3	1
4	3	2
5	4	2
6	5	2
7	6	3
8	6	4
9	7	3
10	8	5
11	9	4
12	10	7
13	11	10
14	12	9
15	16	4
16	17	3
17	18	5
18	18	6
19	19	6



20	20	7
21	20	8
22	21	8
23	22	9
24	26	9
25	26	10
26	27	5
27	28	6
28	30	7
29	31	10
30	32	8

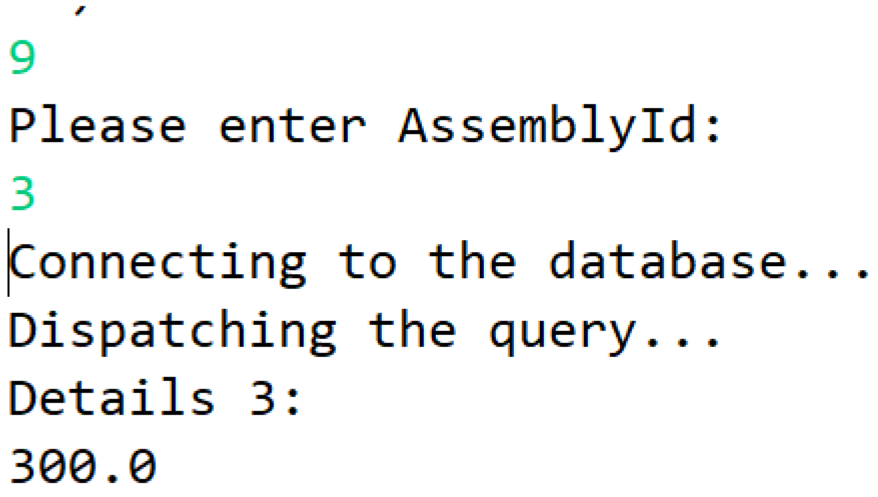
To show database access is possible, perform 3 queries for each type (9, 10, 11, 12, 13, 14).

9)



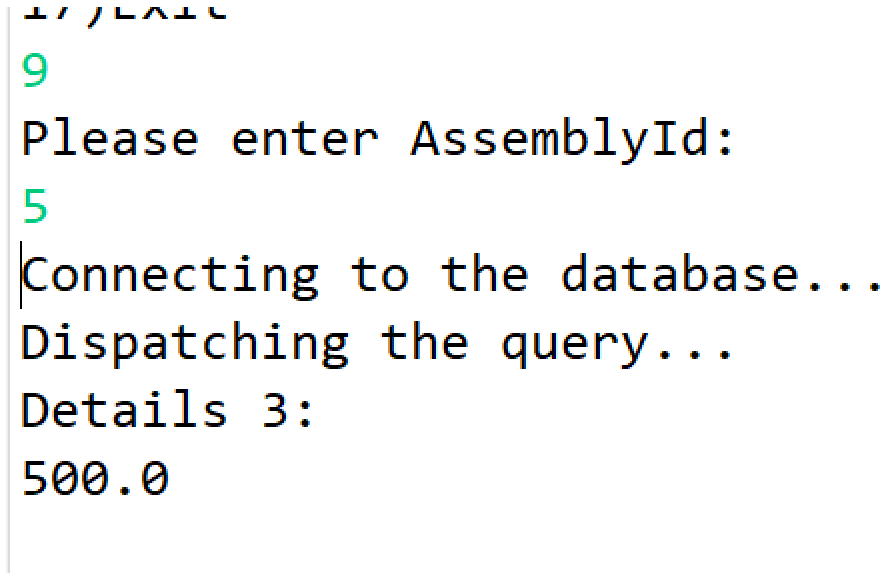
```
7)At the completion of a job, enter the date it completed and the information relevant to the type
of job ;
8)Enter a transaction-no and its sup-cost and update all the costs (details) of the affected
accounts by adding sup-cost to their current values of details;
9)Retrieve the total cost incurred on an assembly-id;
10)Retrieve the total labor time within a department for jobs completed in the department during a
given date;
11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order)
and the department responsible for each process;
12)Retrieve the customers (in name order) whose category is in a given range;
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
9
Please enter AssemblyId:
2
Connecting to the database...
Dispatching the query...
Details 3:
200.0
```

Second Retrieval:



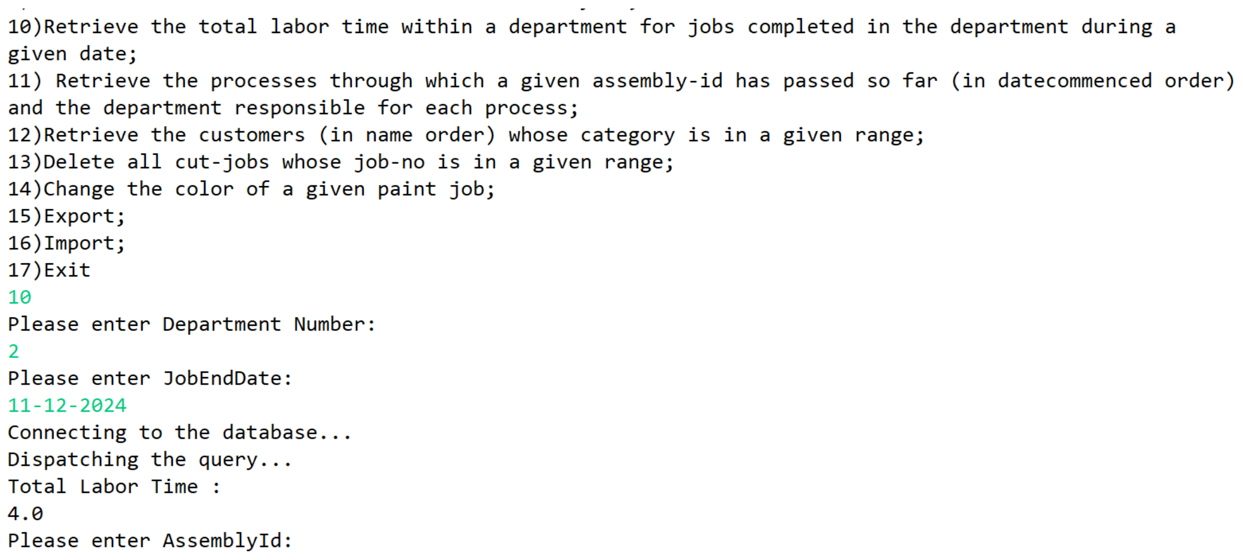
```
9
Please enter AssemblyId:
3
Connecting to the database...
Dispatching the query...
Details 3:
300.0
```

Third Retrieval:



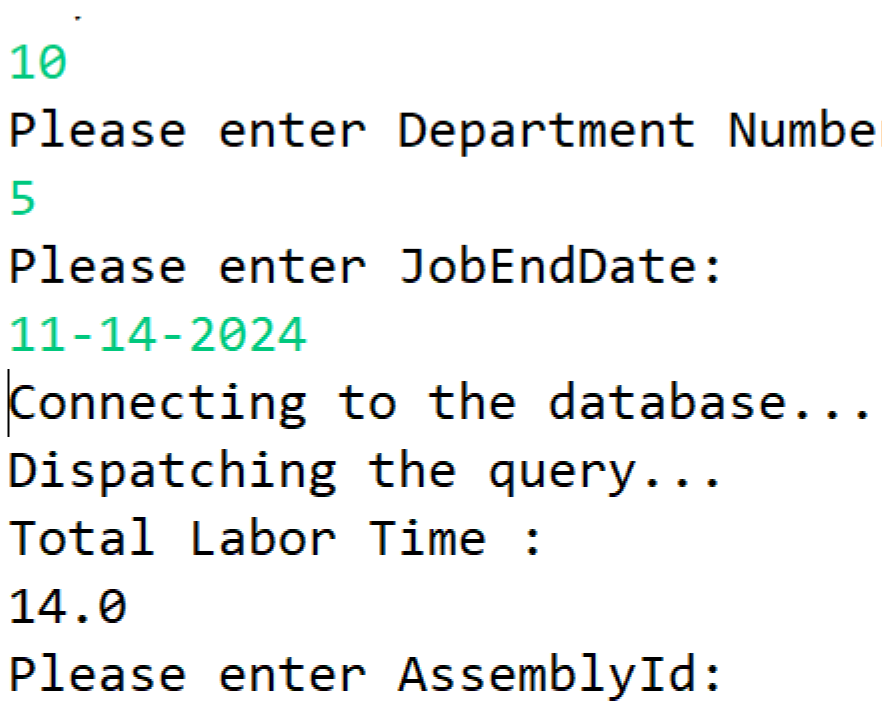
```
17)Exit
9
Please enter AssemblyId:
5
Connecting to the database...
Dispatching the query...
Details 3:
500.0
```

10)First Retrieval:



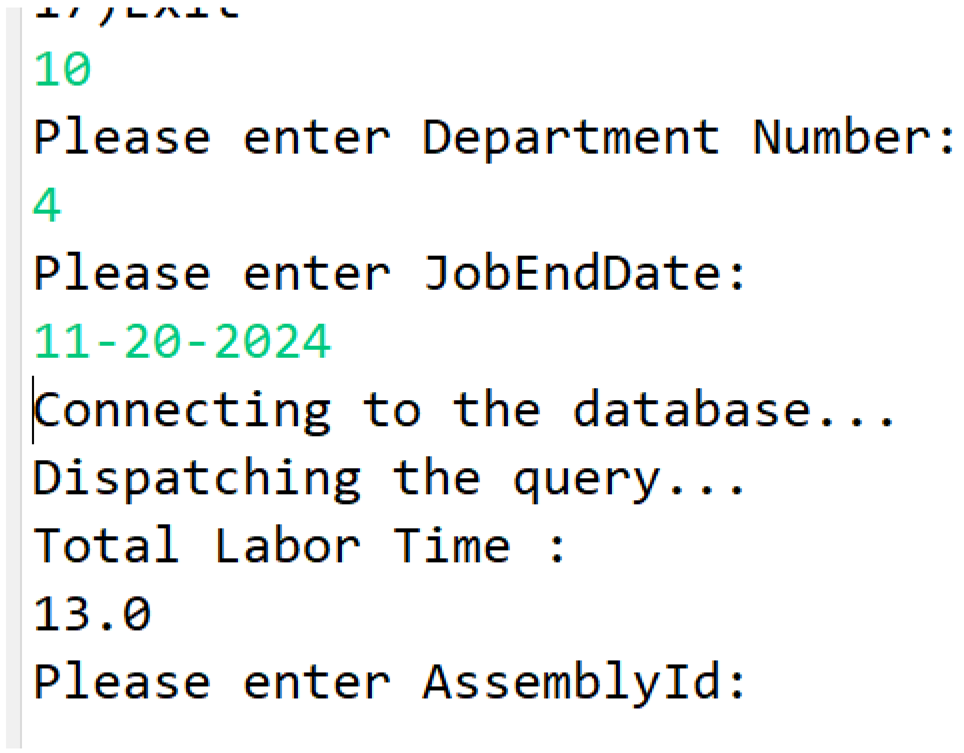
```
10)Retrieve the total labor time within a department for jobs completed in the department during a
given date;
11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order)
and the department responsible for each process;
12)Retrieve the customers (in name order) whose category is in a given range;
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
10
Please enter Department Number:
2
Please enter JobEndDate:
11-12-2024
Connecting to the database...
Dispatching the query...
Total Labor Time :
4.0
Please enter AssemblyId:
```

Second Retrieval:



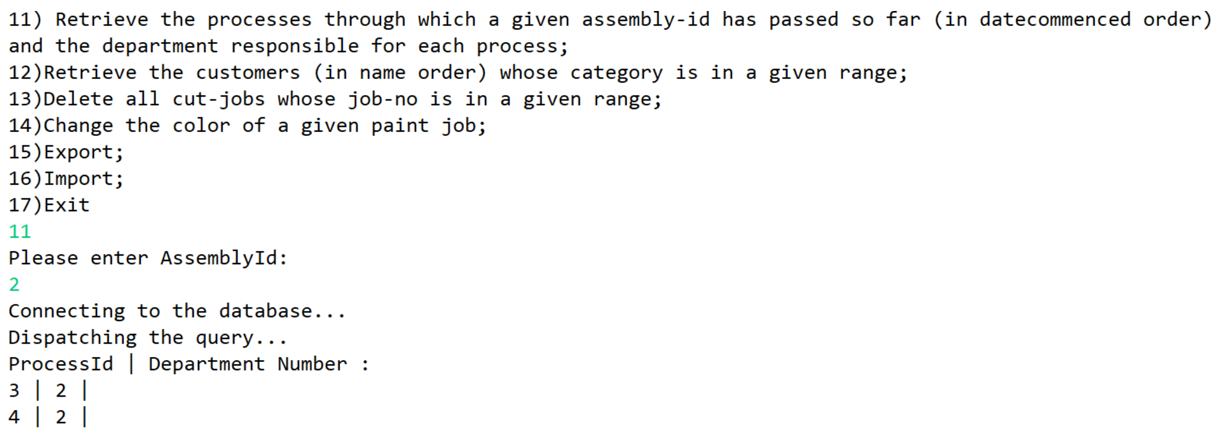
```
10
Please enter Department Number:
5
Please enter JobEndDate:
11-14-2024
Connecting to the database...
Dispatching the query...
Total Labor Time :
14.0
Please enter AssemblyId:
```

Third Retrieval:



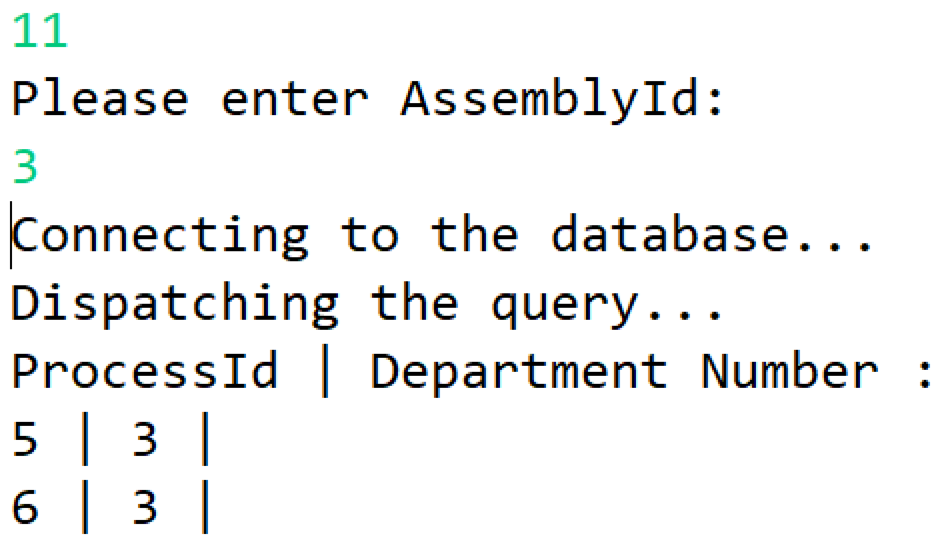
```
10
Please enter Department Number:
4
Please enter JobEndDate:
11-20-2024
Connecting to the database...
Dispatching the query...
Total Labor Time :
13.0
Please enter AssemblyId:
```

11)First Retrieval:



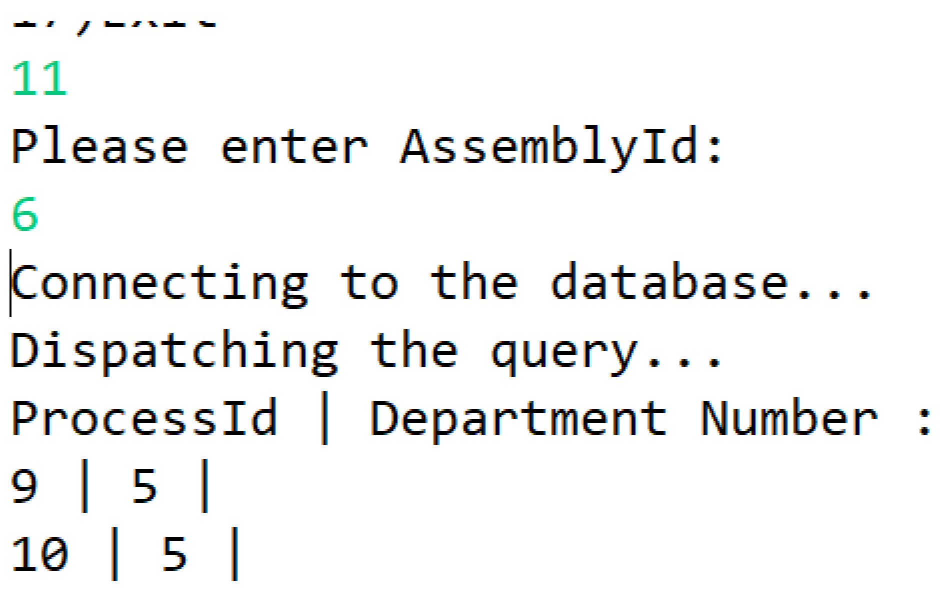
```
11) Retrieve the processes through which a given assembly-id has passed so far (in datecommenced order)
and the department responsible for each process;
12)Retrieve the customers (in name order) whose category is in a given range;
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
11
Please enter AssemblyId:
2
Connecting to the database...
Dispatching the query...
ProcessId | Department Number :
3 | 2 |
4 | 2 |
```

Second Retrieval:



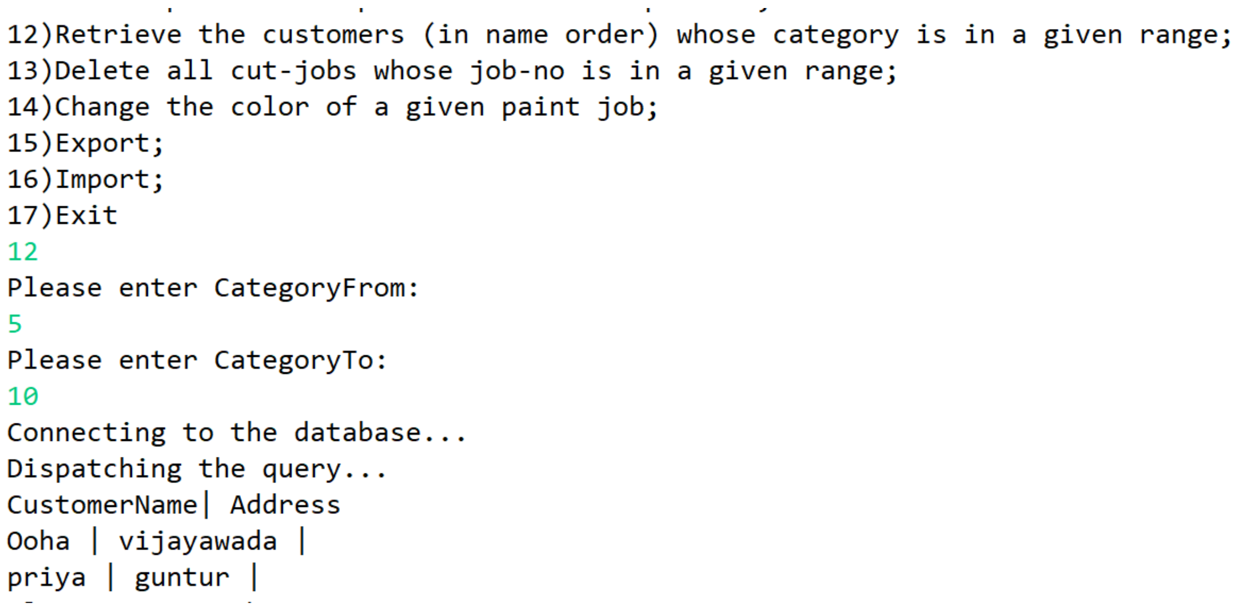
```
11
Please enter AssemblyId:
3
Connecting to the database...
Dispatching the query...
ProcessId | Department Number :
5 | 3 |
6 | 3 |
```

Third Retrieval:



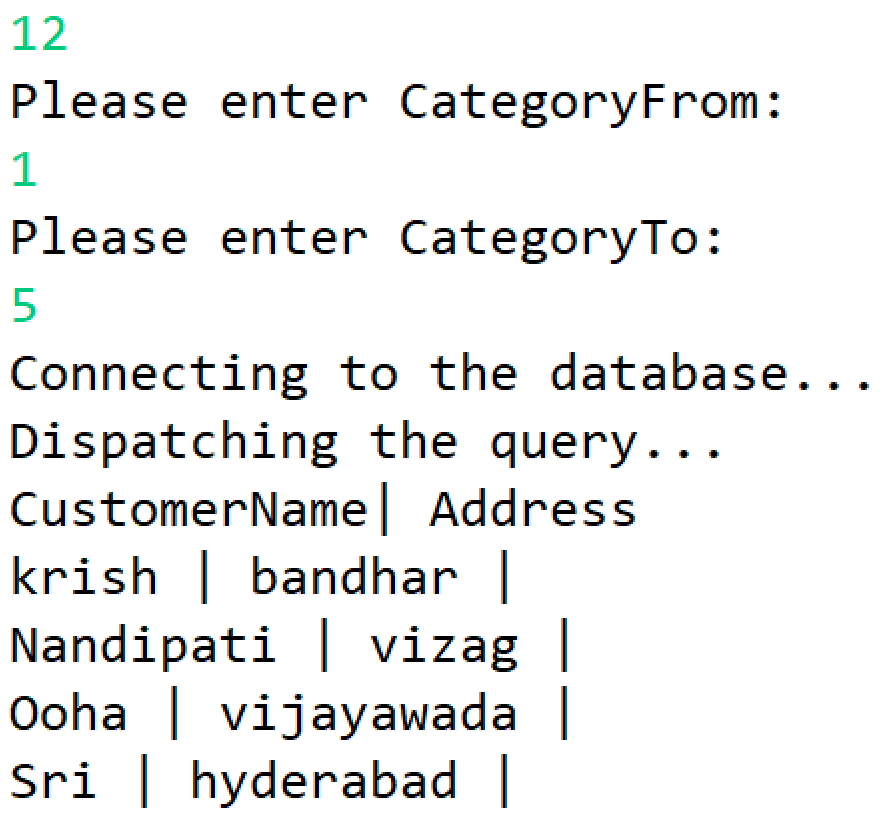
```
11
Please enter AssemblyId:
6
Connecting to the database...
Dispatching the query...
ProcessId | Department Number :
9 | 5 |
10 | 5 |
```

12)First Retrieval:



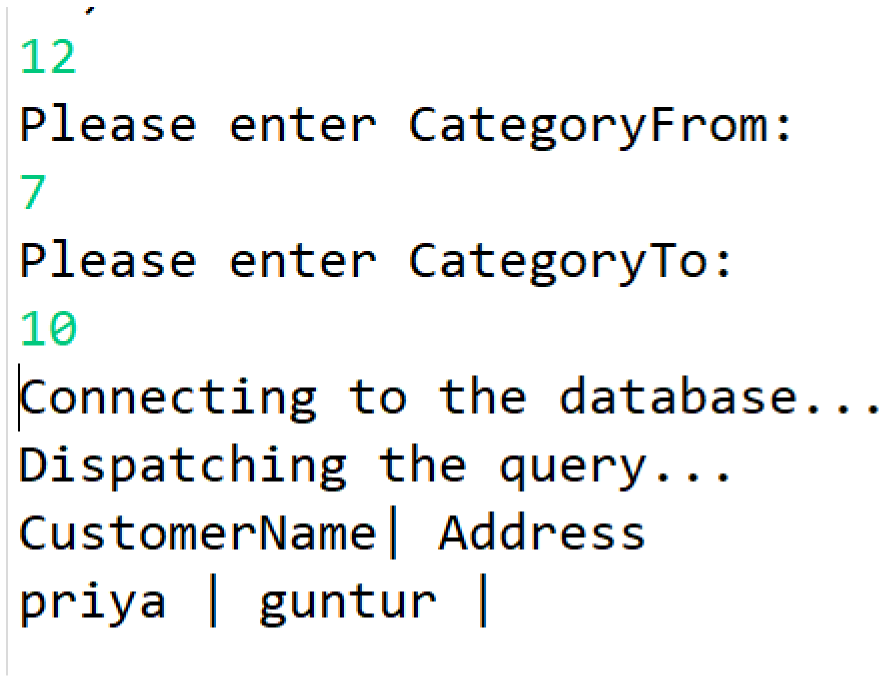
```
12)Retrieve the customers (in name order) whose category is in a given range;
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
12
Please enter CategoryFrom:
5
Please enter CategoryTo:
10
Connecting to the database...
Dispatching the query...
CustomerName| Address
Ooha | vijayawada |
priya | guntur |
```

Second Retrieval:



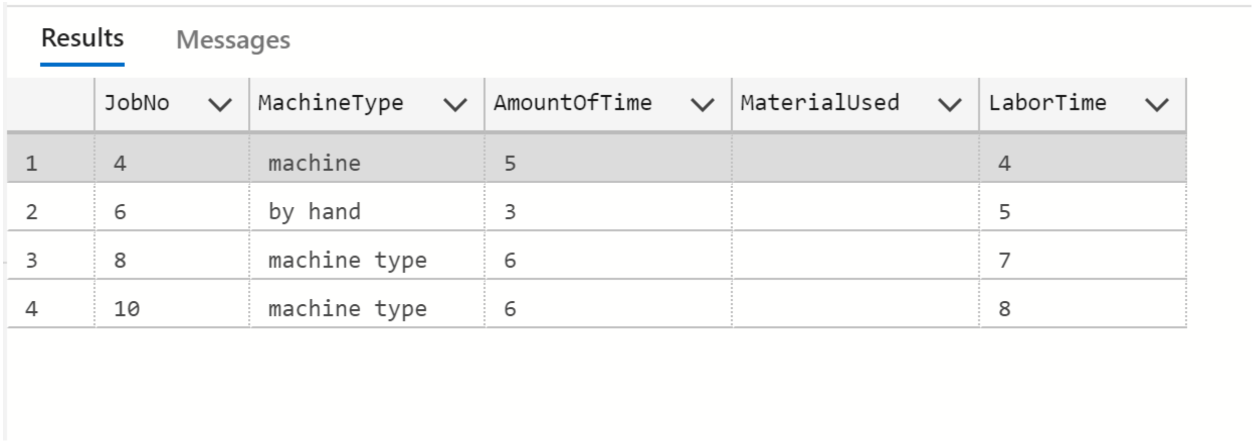
```
12
Please enter CategoryFrom:
1
Please enter CategoryTo:
5
Connecting to the database...
Dispatching the query...
CustomerName| Address
krish | bandhar |
Nandipati | vizag |
Ooha | vijayawada |
Sri | hyderabad |
```

Third Retrieval:



```
12
Please enter CategoryFrom:
7
Please enter CategoryTo:
10
Connecting to the database...
Dispatching the query...
CustomerName| Address
priya | guntur |
```

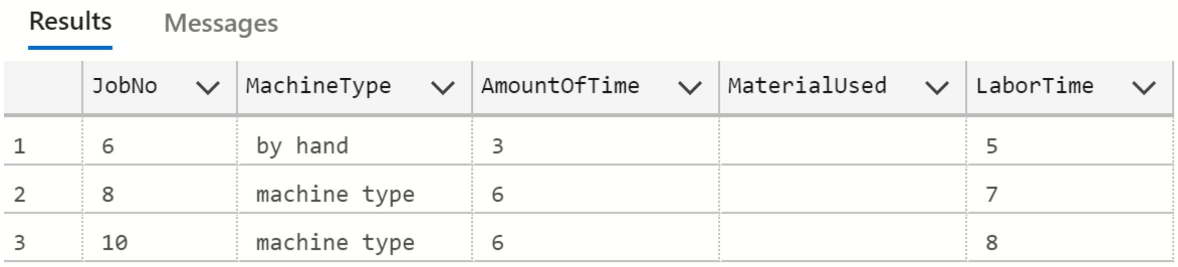
13) Before Deletion:



Results Messages

	JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime
1	4	machine	5		4
2	6	by hand	3		5
3	8	machine type	6		7
4	10	machine type	6		8

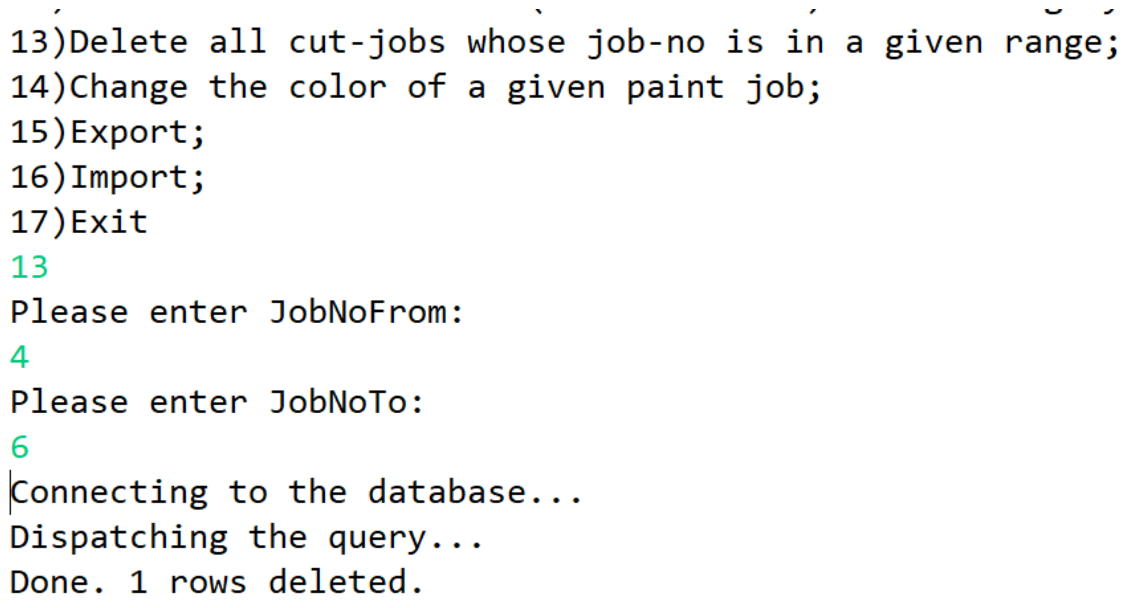
After Deletion:



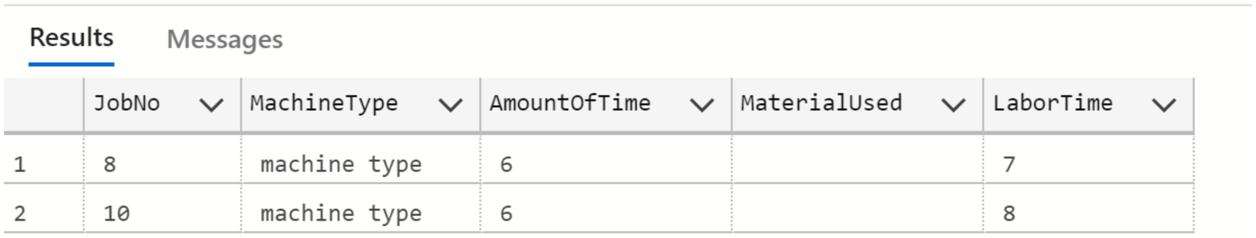
Results Messages

	JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime
1	6	by hand	3		5
2	8	machine type	6		7
3	10	machine type	6		8

Second:



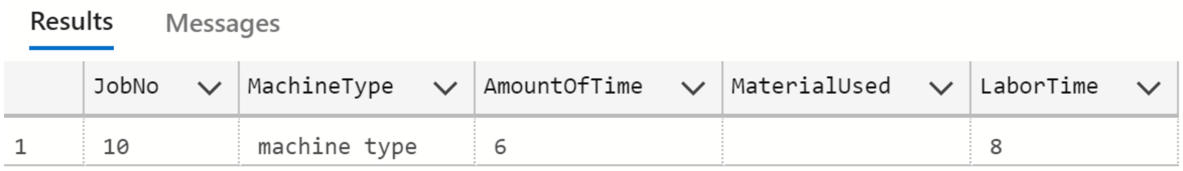
```
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
13
Please enter JobNoFrom:
4
Please enter JobNoTo:
6
Connecting to the database...
Dispatching the query...
Done. 1 rows deleted.
```



Results Messages

	JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime
1	8	machine type	6		7
2	10	machine type	6		8

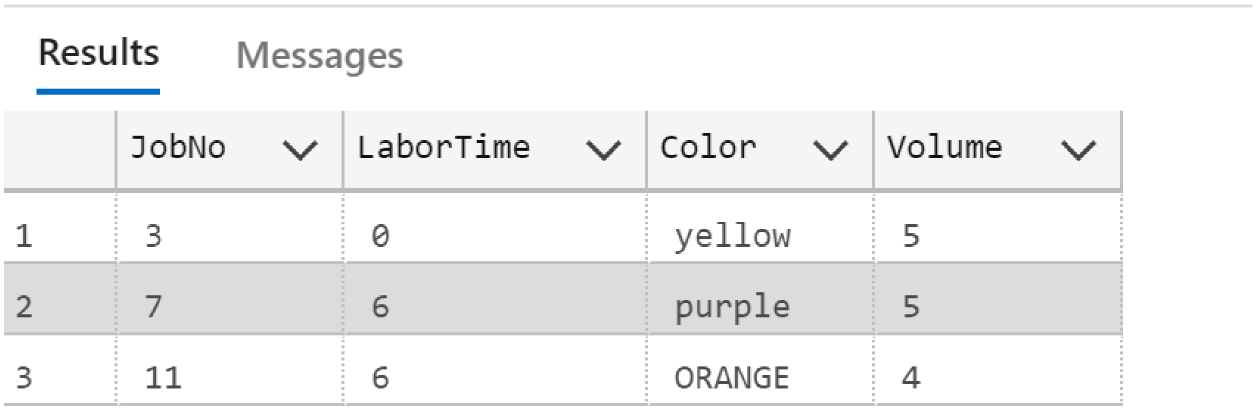
Third:



Results Messages

	JobNo	MachineType	AmountOfTime	MaterialUsed	LaborTime
1	10	machine type	6		8

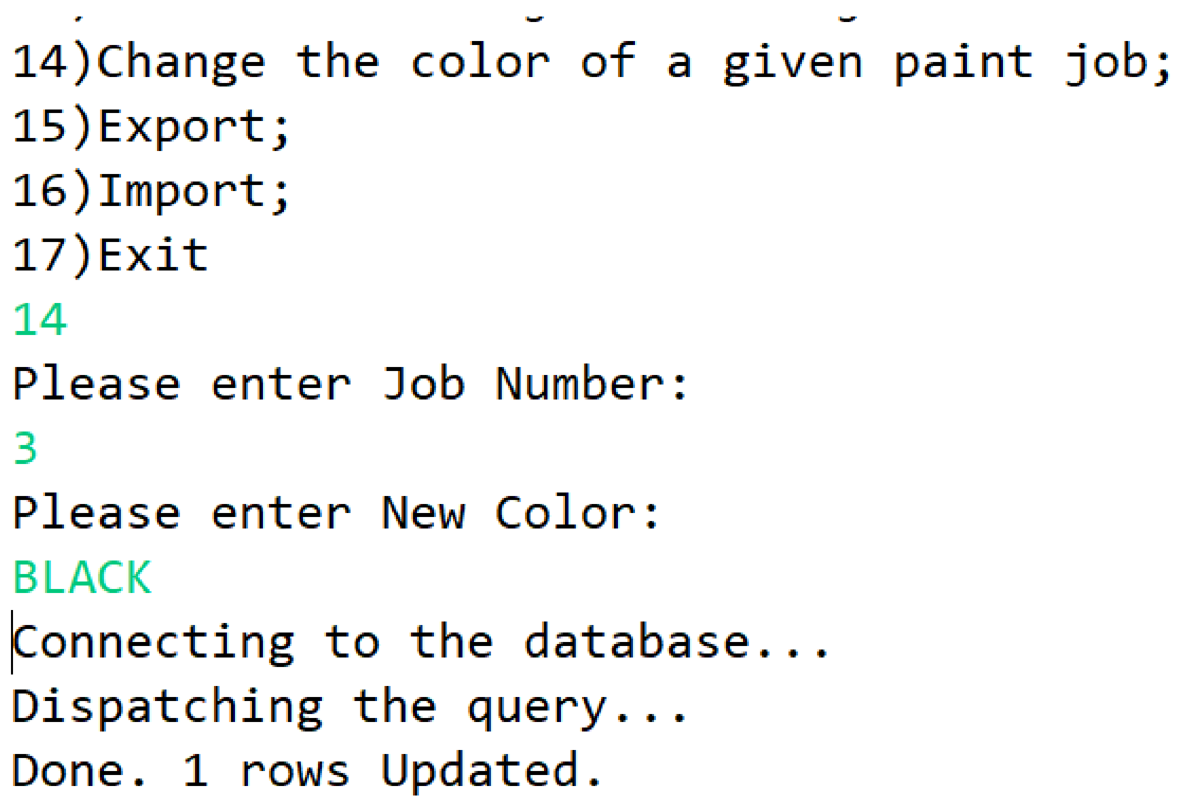
14)
Before Updation:
First Updation:



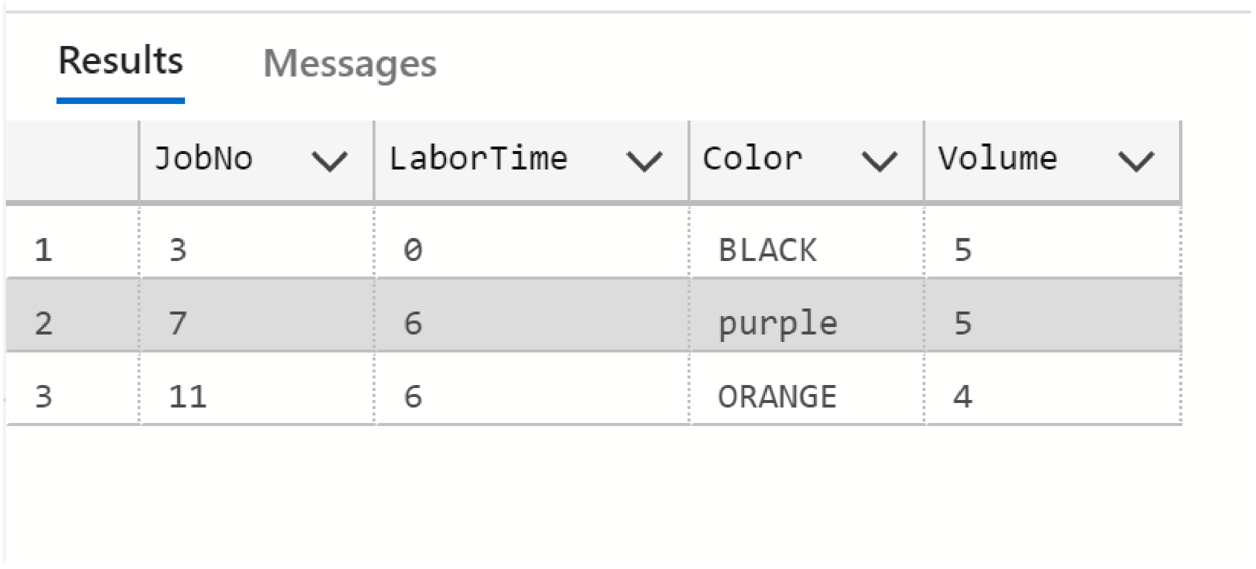
Results Messages

	JobNo	LaborTime	Color	Volume
1	3	0	yellow	5
2	7	6	purple	5
3	11	6	ORANGE	4

After Updation:

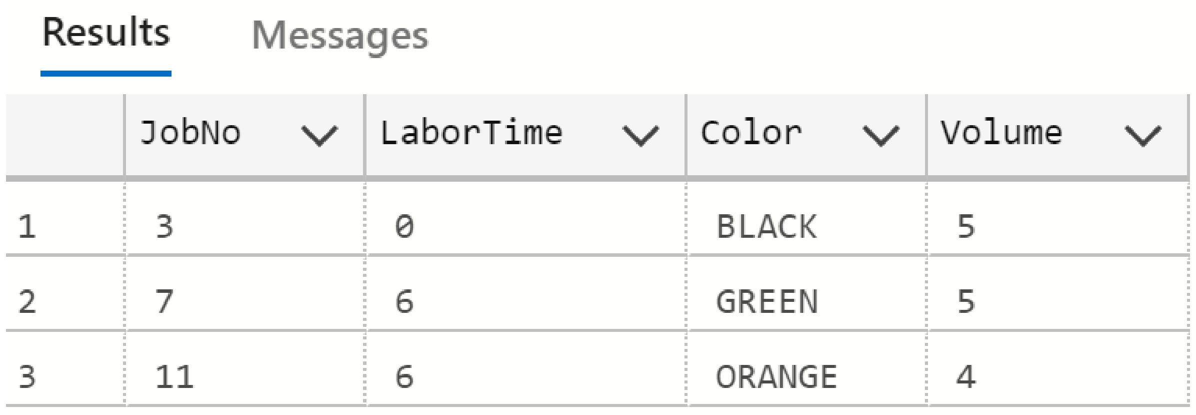


```
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
14
Please enter Job Number:
3
Please enter New Color:
BLACK
Connecting to the database...
Dispatching the query...
Done. 1 rows Updated.
```



	JobNo	LaborTime	Color	Volume
1	3	0	BLACK	5
2	7	6	purple	5
3	11	6	ORANGE	4

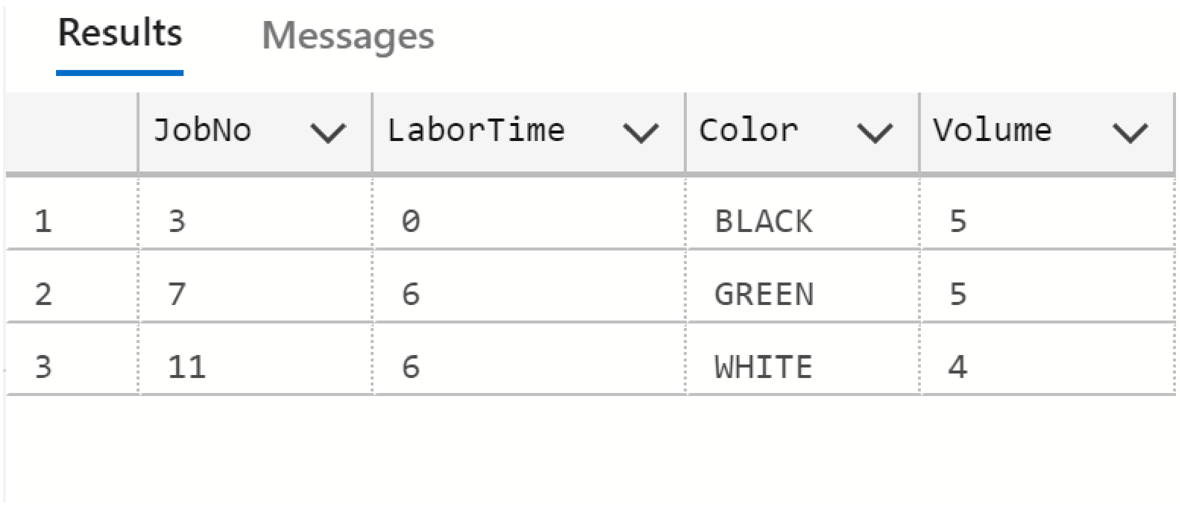
Second:



	JobNo	LaborTime	Color	Volume
1	3	0	BLACK	5
2	7	6	GREEN	5
3	11	6	ORANGE	4

THIRD:

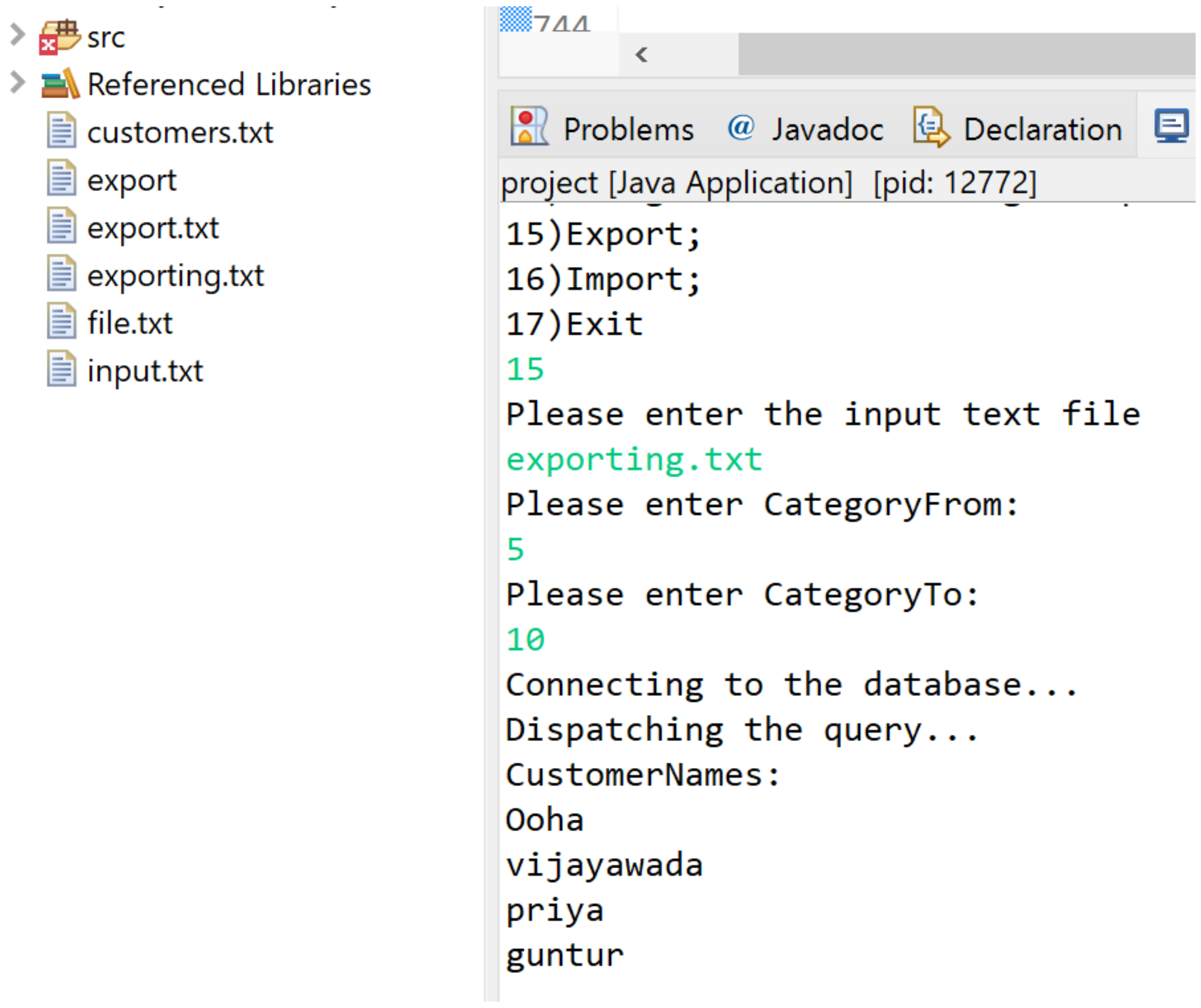
```
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
14
Please enter Job Number:
11
Please enter New Color:
WHITE
Connecting to the database...
Dispatching the query...
Done. 1 rows Updated.
```



	JobNo	LaborTime	Color	Volume
1	3	0	BLACK	5
2	7	6	GREEN	5
3	11	6	WHITE	4

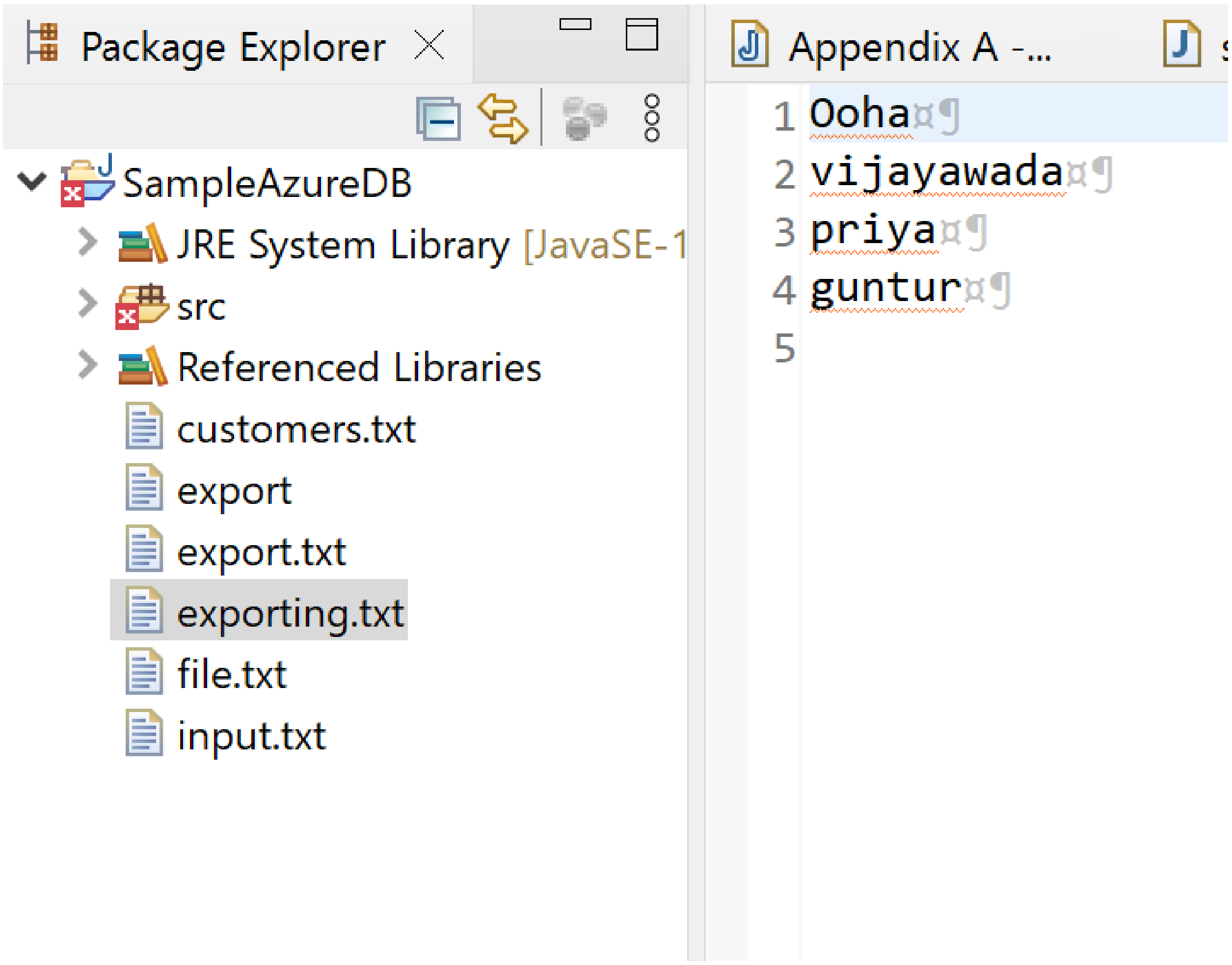
To show the import and export facilities are available, run each option (15-16) once

15)Export:



project [Java Application] [pid: 12772]

```
15)Export;
16)Import;
17)Exit
15
Please enter the input text file
exporting.txt
Please enter CategoryFrom:
5
Please enter CategoryTo:
10
Connecting to the database...
Dispatching the query...
CustomerNames:
Ooha
vijayawada
priya
guntur
```

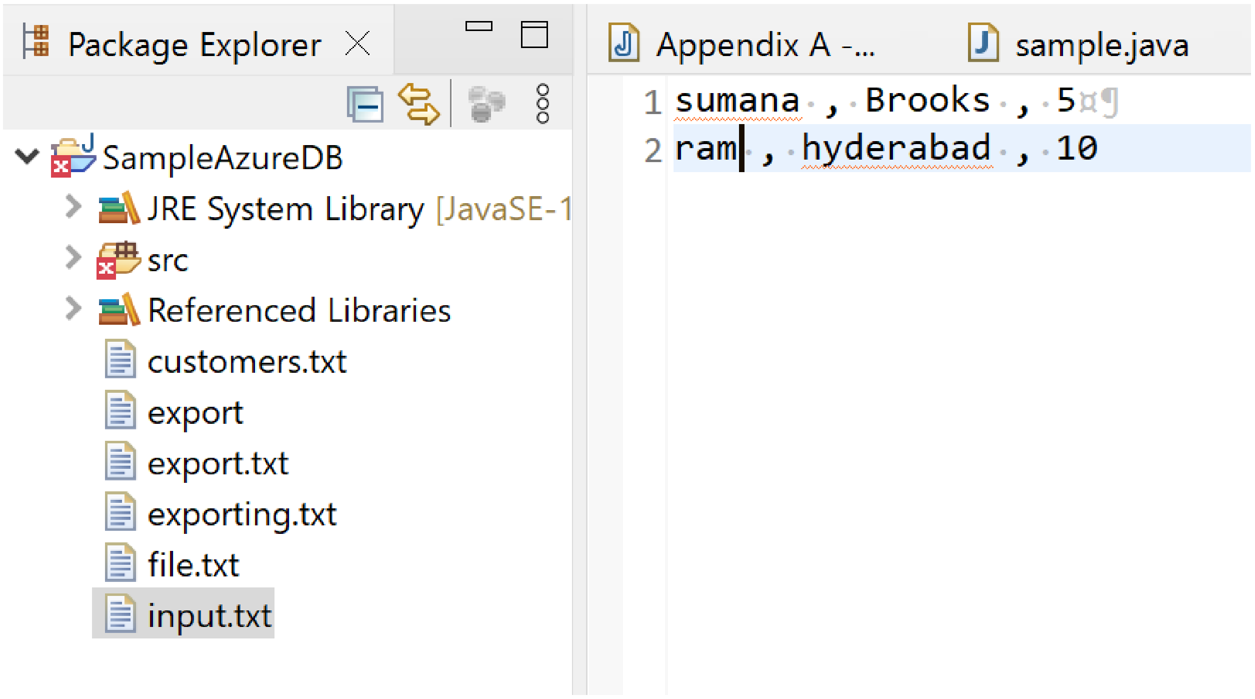


SampleAzureDB
JRE System Library [JavaSE-1
src
Referenced Libraries
customers.txt
export
export.txt
exporting.txt
file.txt
input.txt

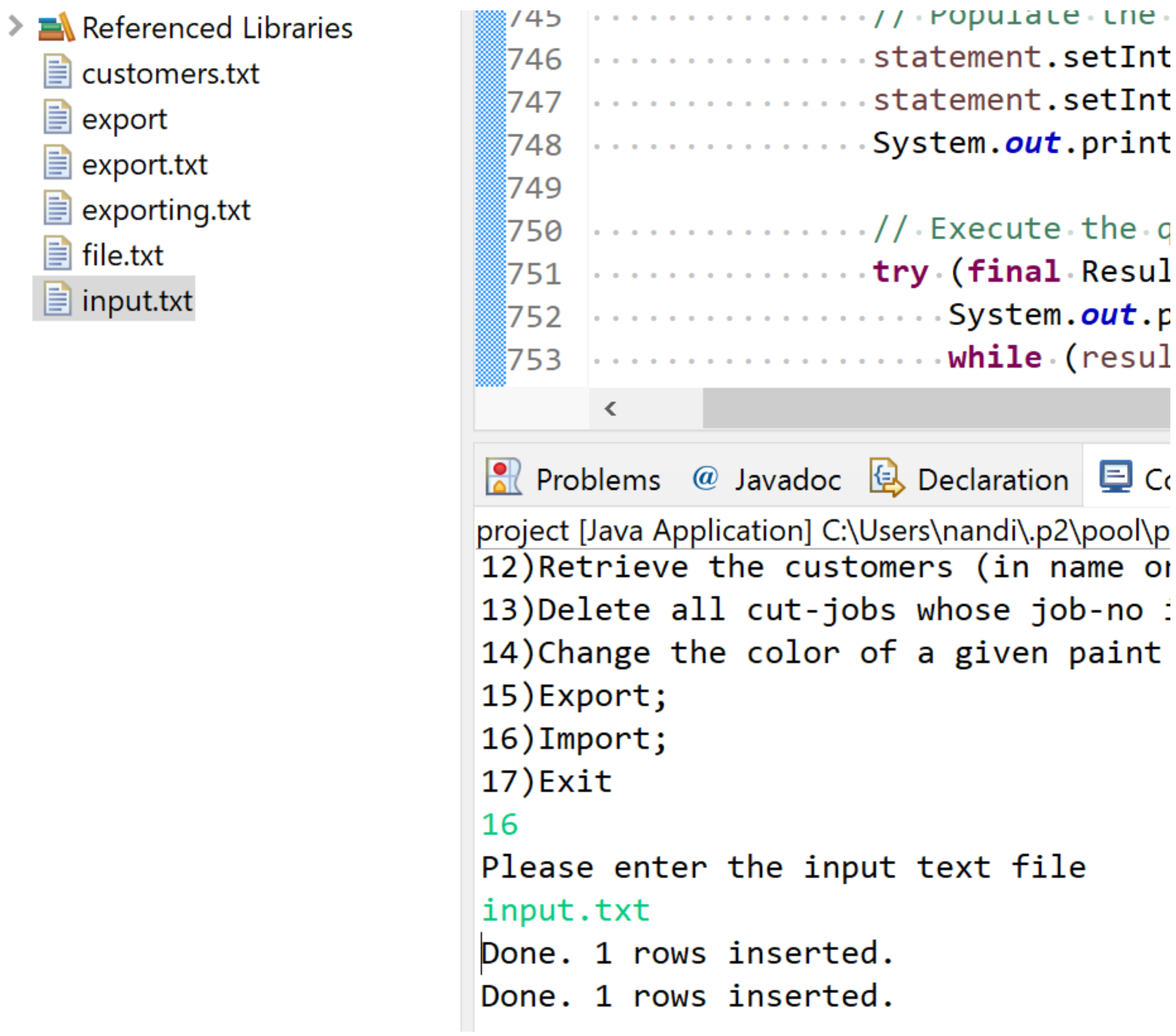
```
Ooha
vijayawada
priya
guntur
```

16)Importing:

Input:

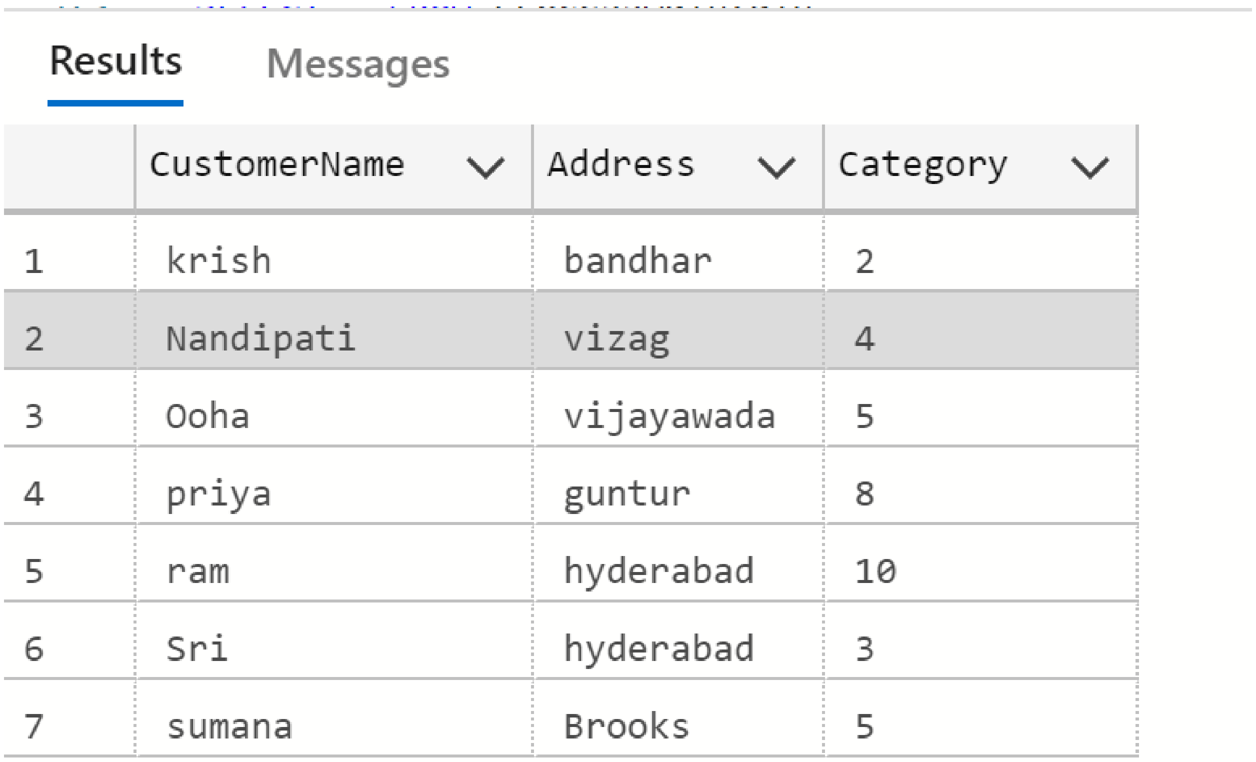


```
sumana , Brooks , 5
ram , hyderabad , 10
```



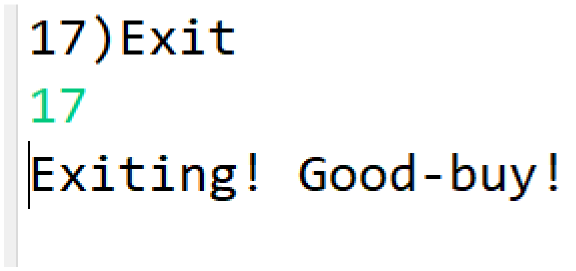
```
                // Populate the
                statement.setInt
                statement.setInt
                System.out.print

                // Execute the q
                try (final Resul
                    System.out.p
                    while (resul
```

```
project [Java Application] C:\Users\nandi\.p2\pool\p
12)Retrieve the customers (in name or
13)Delete all cut-jobs whose job-no i
14)Change the color of a given paint
15)Export;
16)Import;
17)Exit
16
Please enter the input text file
input.txt
Done. 1 rows inserted.
Done. 1 rows inserted.
```



	CustomerName	Address	Category
1	krish	bandhar	2
2	Nandipati	vizag	4
3	Ooha	vijayawada	5
4	priya	guntur	8
5	ram	hyderabad	10
6	Sri	hyderabad	3
7	sumana	Brooks	5

To show the Quit option is available, run option (17) at least once

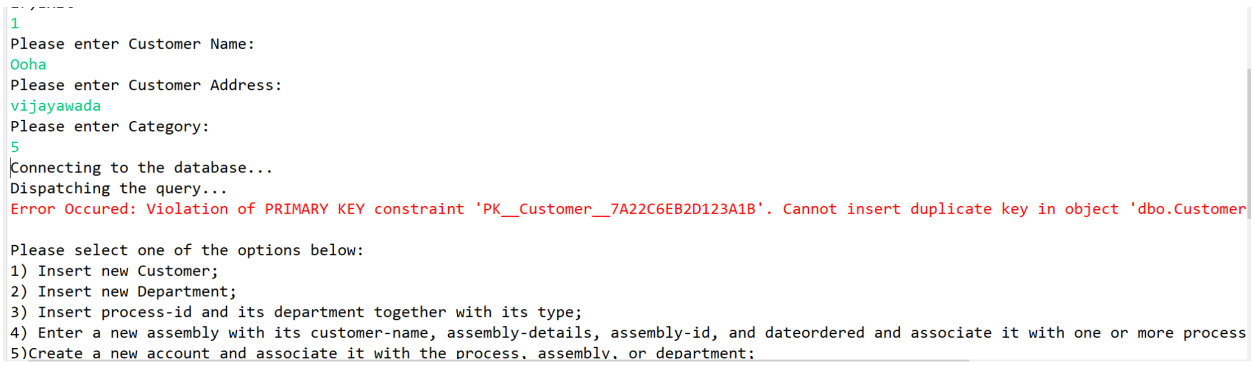


```
17)Exit
17
Exiting! Good-buy!
```

To demonstrate that Azure SQL Database can detect errors, you also need to perform 3 queries of different types that contain some errors.
Error when given value is not in the database.

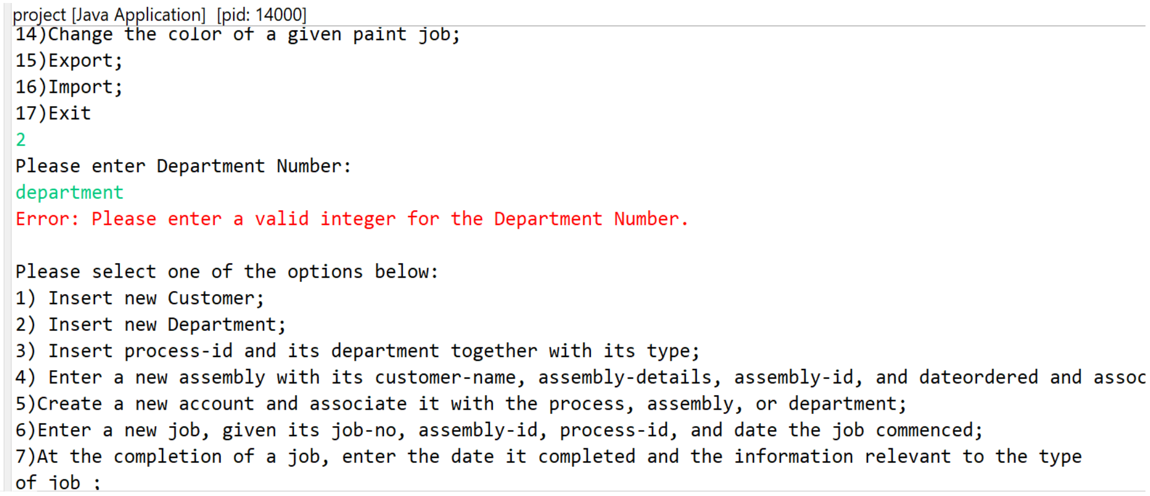
```
13)Delete all cut-jobs whose job-no is in a given range;
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
13
Please enter JobNoFrom:
11
Please enter JobNoTo:
20
Connecting to the database...
Dispatching the query...
Error Occured: No matching CutJobs found for the specified JobNo range.
```

Primary key violation error:



```
1
Please enter Customer Name:
Ooha
Please enter Customer Address:
vijayawada
Please enter Category:
5
Connecting to the database...
Dispatching the query...
Error Occured: Violation of PRIMARY KEY constraint 'PK__Customer__7A22C6EB2D123A1B'. Cannot insert duplicate key in object 'dbo.Customer

Please select one of the options below:
1) Insert new Customer;
2) Insert new Department;
3) Insert process-id and its department together with its type;
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and dateordered and associate it with one or more process
5)Create a new account and associate it with the process, assembly, or department;
```

Data Mismatch Error:



project [Java Application] [pid: 14000]

```
14)Change the color of a given paint job;
15)Export;
16)Import;
17)Exit
2
Please enter Department Number:
department
Error: Please enter a valid integer for the Department Number.

Please select one of the options below:
1) Insert new Customer;
2) Insert new Department;
3) Insert process-id and its department together with its type;
4) Enter a new assembly with its customer-name, assembly-details, assembly-id, and dateordered and assoc
5)Create a new account and associate it with the process, assembly, or department;
6)Enter a new job, given its job-no, assembly-id, process-id, and date the job commenced;
7)At the completion of a job, enter the date it completed and the information relevant to the type
of job ;
```

Task 7. (23 points): Write a Web database application using Azure SQL Database and JSP which provides the Web pages for query 1 and query 12. Since both queries take the input data from the user, there should be two Web pages for each query as follows: for query 1, one Web page to allow the user to enter the input data and one to display a message confirming the successful execution of the insertion; and for query 12, there should be one Web page to allow the user to enter the input data and one to display the retrieval results with appropriate headings. To show that your Web application works correctly, run the Web application so that queries 1 and 12 will be executed in this order: first query 12, then query 1, and then query 12 again, making sure that the results of query 1 will change the results of query 12 that follow query 1.

DATAHANDLER.JAVA

```
package jsp_azure;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class DataHandler {
  private Connection conn;
  // Azure SQL connection credentials
  private String server = "nand0019.database.windows.net";
  private String database = "cs-dsa-4513-sql-db";
  private String username = "nand0019";
  private String password = "Oohasrinandi@123";
  // Resulting connection string
  final private String url =

String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCe
rtificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;",
            server, database, username, password);
  // Initialize and save the database connection
  private void getDBConnection() throws SQLException {
    if (conn != null) {
      return;
    }
    this.conn = DriverManager.getConnection(url);
  }
  // Return the result of selecting everything from the movie_night table
  public ResultSet getAllMovies() throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM Customer;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    return stmt.executeQuery();
  }
  public ResultSet getbyCategory(int CategoryFrom ,int CategoryTo) throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM Customer where category between ? and ?;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    stmt.setInt(1, CategoryFrom);
    stmt.setInt(2, CategoryTo);

    return stmt.executeQuery();
  }
  // Inserts a record into the movie_night table with the given attribute values
  public boolean Customer(String CustomerName,String Address,int Category) throws SQLException {
    getDBConnection(); // Prepare the database connection
    // Prepare the SQL statement
final String sqlQuery =

                    "INSERT INTO Customer " + "(CustomerName, Address , Category)"+
                  "VALUES " + "(?, ?, ?)";
             /*
            "INSERT INTO movie_night " +
            "(start_time, movie_name, duration_min, guest_1, guest_2, guest_3, guest_4, guest_5) " +
            "VALUES " +
            "(?, ?, ?, ?, ?, ?, ?, ?)";
            */
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    // Replace the '?' in the above statement with the given attribute values
    stmt.setString(1, CustomerName);
    stmt.setString(2, Address);
    stmt.setInt(3, Category);
    // Execute the query, if only one record is updated, then we indicate success by returning true
    return stmt.executeUpdate() == 1;
  }
}
```

add_customer_form.jsp

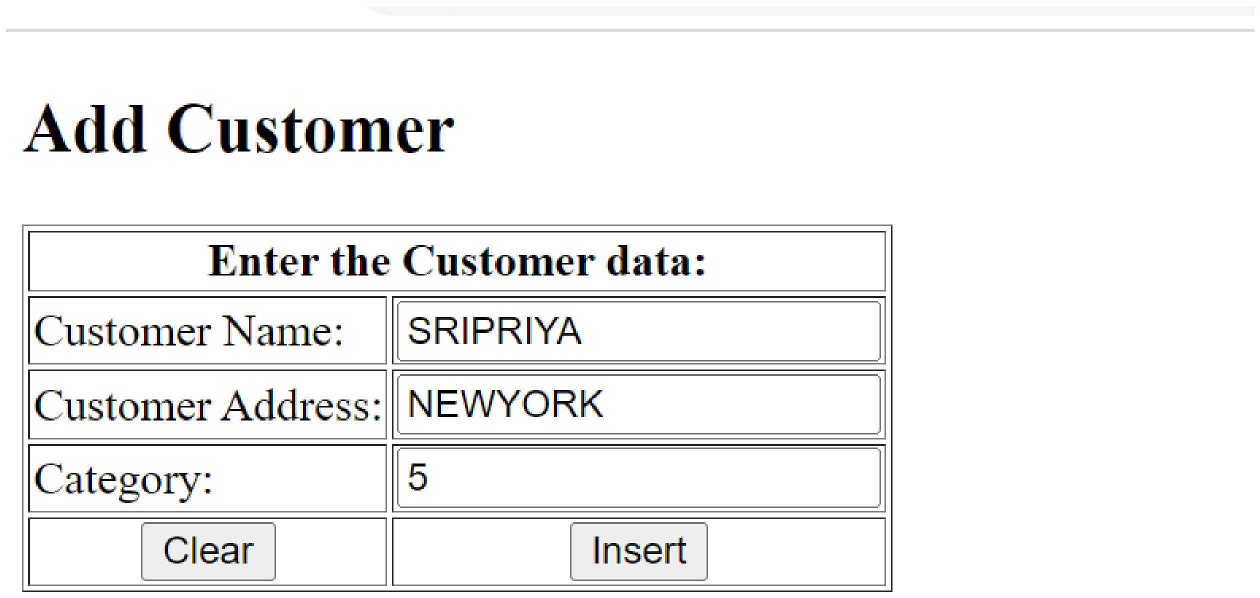
```html
<!DOCTYPE html>
<html>
  <head>
      <meta charset="UTF-8">
      <title>Add Customer</title>
  </head>
  <body>
      <h2>Add Customer</h2>
      <!--
          Form for collecting user input for the new movie_night record.
          Upon form submission, add_movie.jsp file will be invoked.
      -->
      <form action="add_movie.jsp">
          <!-- The form organized in an HTML table for better clarity. -->
          <table border=1>
              <tr>
                  <th colspan="2">Enter the Customer data:</th>
              </tr>
              <tr>
                  <td>Customer Name:</td>
                  <td><div style="text-align: center;">
                  <input type=text name=CustomerName>
                  </div></td>
              </tr>
              <tr>
                  <td>Customer Address:</td>
                  <td><div style="text-align: center;">
                  <input type=text name=Address>
                  </div></td>
              </tr>
              <tr>
                  <td>Category:</td>
                  <td><div style="text-align: center;">
                  <input type=text name=Category>
                  </div></td>
              </tr>
              <tr>

                  <td><div style="text-align: center;">
                  <input type=reset value=Clear>
                  </div></td>
                  <td><div style="text-align: center;">
                  <input type=submit value=Insert>
                  </div></td>
              </tr>
          </table>
      </form>
  </body>
</html>
```

add_customer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
  <body>
  <%@page import="jsp_azure.DataHandler"%>
  <%@page import="java.sql.ResultSet"%>
  <%@page import="java.sql.Array"%>
  <%
  // The handler is the one in charge of establishing the connection.
  DataHandler handler = new DataHandler();
  // Get the attribute values passed from the input form.
  String CustomerName = request.getParameter("CustomerName");
  String Address = request.getParameter("Address");
  int Category = Integer.parseInt(request.getParameter("Category"));
  /*
  String g1 = request.getParameter("guest_1");
  String g2 = request.getParameter("guest_2");
  String g3 = request.getParameter("guest_3");
  String g4 = request.getParameter("guest_4");
  String g5 = request.getParameter("guest_5");
  */
  /*
   * If the user hasn't filled out all the time, movie name and duration. This is very simple checking.
   */


     // Now perform the query with the data from the form.
     boolean success = handler.Customer(CustomerName, Address, Category);
     if (!success) { // Something went wrong
        %>
           <h2>There was a problem inserting the course</h2>
        <%
     } else { // Confirm success to the user
        %>
        <h2>Customer table:</h2>
        <ul>
           <li>CustomerName: <%= CustomerName%></li>
```

```
            <li>Address: <%= Address%></li>
            <li>Category: <%=Category%></li>



        </ul>
        <h2>Was successfully inserted.</h2>

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

OUTPUT:



Add Customer

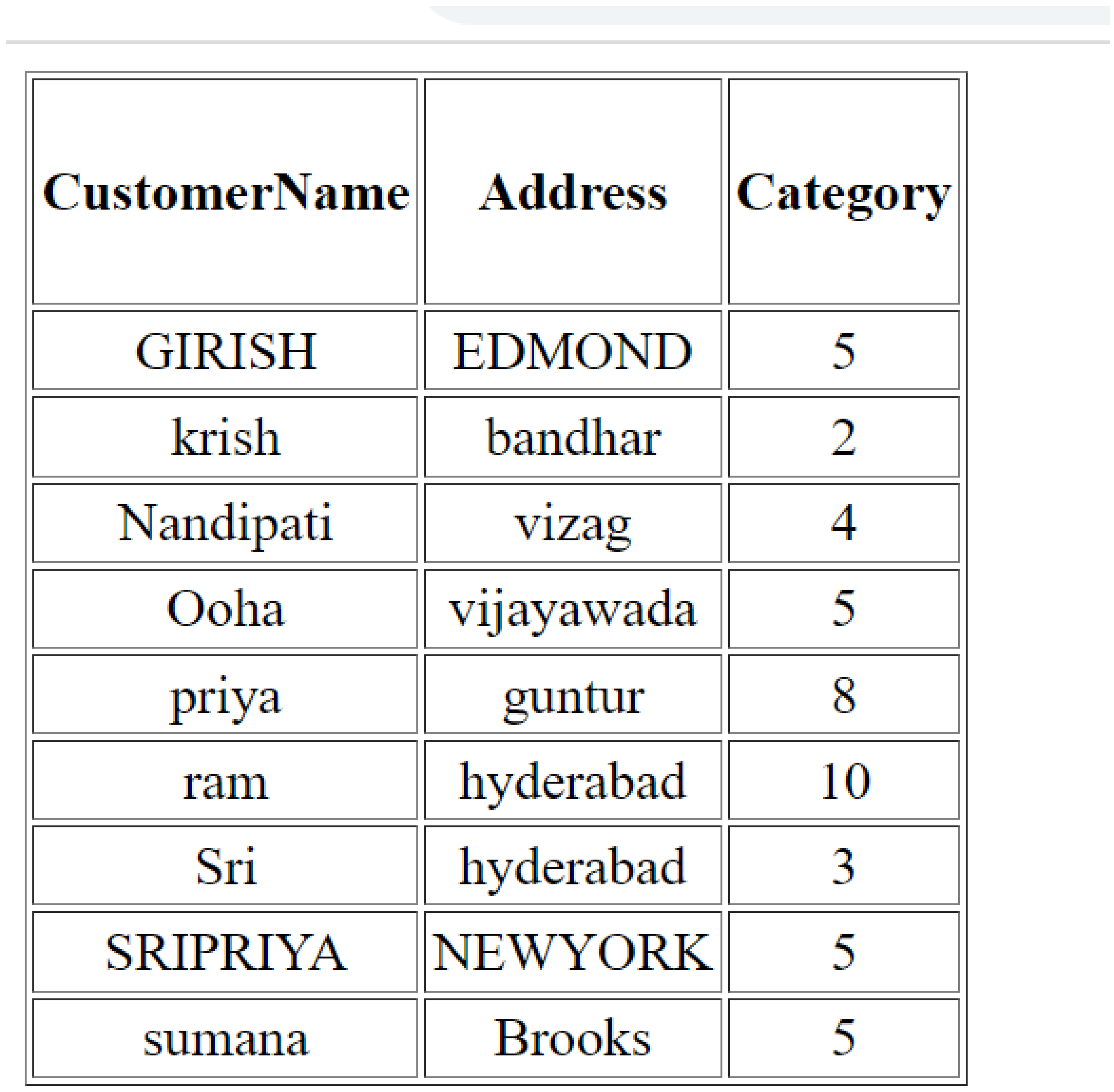
Enter the Customer data:	
Customer Name:	SRIPRIYA
Customer Address:	NEWYORK
Category:	5
Clear	Insert

Customer table:

- CustomerName: SRIPRIYA
- Address: NEWYORK
- Category: 5

Was successfully inserted.

See all Customer Names.



CustomerName	Address	Category
GIRISH	EDMOND	5
krish	bandhar	2
Nandipati	vizag	4
Ooha	vijayawada	5
priya	guntur	8
ram	hyderabad	10
Sri	hyderabad	3
SRIPRIYA	NEWYORK	5
sumana	Brooks	5

get_customer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
  <meta charset="UTF-8">
     <title>Customer Table</title>
  </head>
  <body>
```

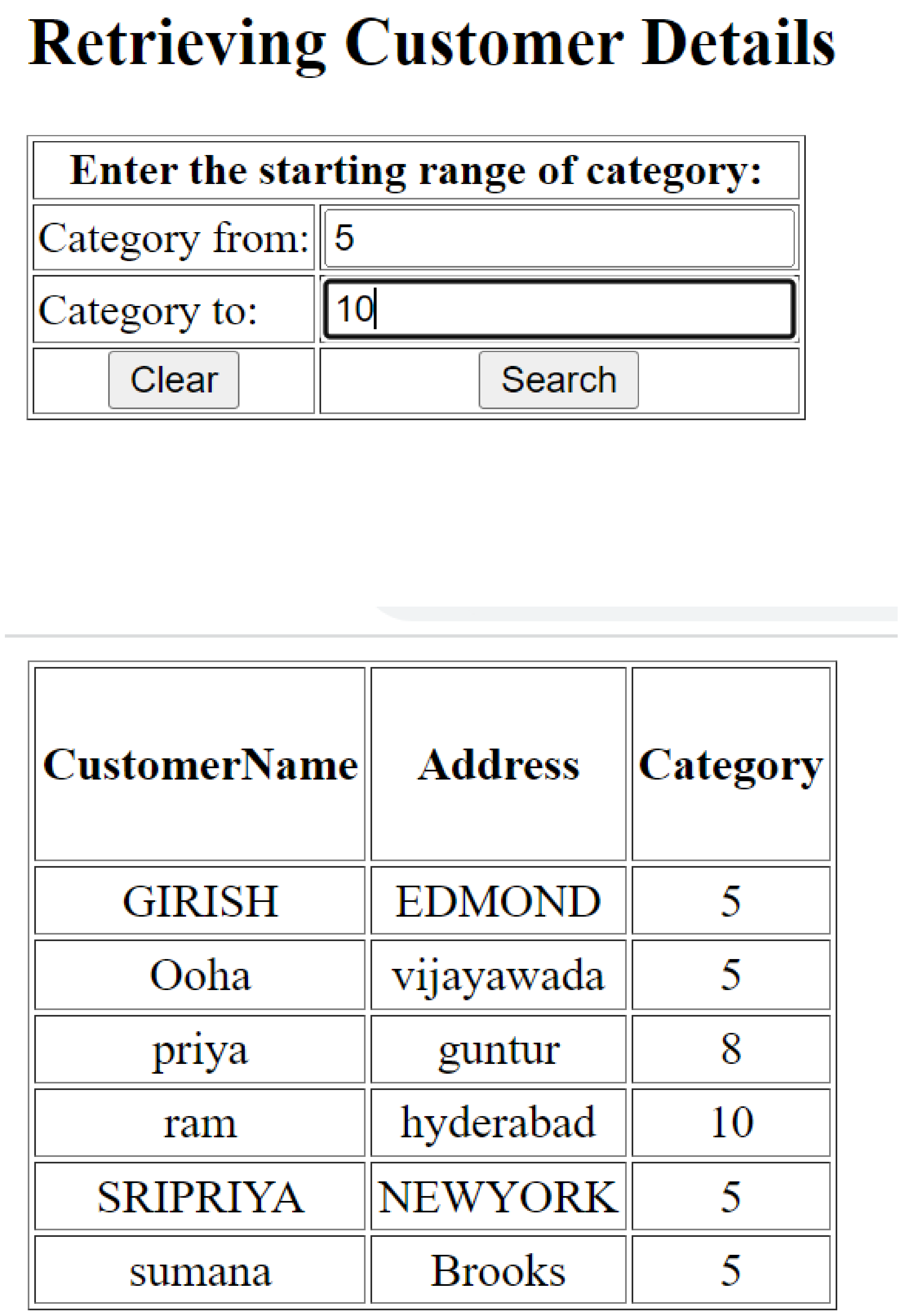
```
    <%@page import="jsp_azure.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%
       // We instantiate the data handler here, and get all the movies from the database
    final DataHandler handler = new DataHandler();
    int CategoryFrom = Integer.parseInt(request.getParameter("CategoryFrom"));
        int CategoryTo = Integer.parseInt(request.getParameter("CategoryTo"));
    final ResultSet movies = handler.getbyCategory(CategoryFrom,CategoryTo);
    %>
    <!-- The table for displaying all the movie records -->
    <table cellspacing="2" cellpadding="2" border="1">
       <tr> <!-- The table headers row -->
         <td align="center">
           <h4>CustomerName</h4>
         </td>
         <td align="center">
           <h4>Address</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 CustomerName = movies.getString("CustomerName");
             final String Address = movies.getString("Address");
             final int Category = movies.getInt("Category");

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

get_customer_form.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
```

```
        <title>Retrieve Customer from Category</title>
    </head>
    <body>
        <h2>Add Category From</h2>
        <!--
            Form for collecting user input for the new movie_night record.
            Upon form submission, add_movie.jsp file will be invoked.
        -->
        <form action="get_all_movies.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Enter the Category From:</th>
                </tr>
                <tr>
                    <td>Category From:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=CategoryFrom>
                    </div></td>
                </tr>
                <tr>
                    <td>Category To:</td>
                    <td><div style="text-align: center;">
                    <input type=text name=CategoryTo>
                    </div></td>
                </tr>
                <tr>


                    <td><div style="text-align: center;">
                    <input type=reset value=Clear>
                    </div></td>
                    <td><div style="text-align: center;">
                    <input type=submit value=Search>
                    </div></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

OUTPUT:



Retrieving Customer Details

Enter the starting range of category:	
Category from:	5
Category to:	10
Clear	Search

CustomerName	Address	Category
GIRISH	EDMOND	5
Ooha	vijayawada	5
priya	guntur	8
ram	hyderabad	10
SRIPRIYA	NEWYORK	5
sumana	Brooks	5