CS-4513-001

FALL 2023

Professor Gruenwald

Gabriel Owen

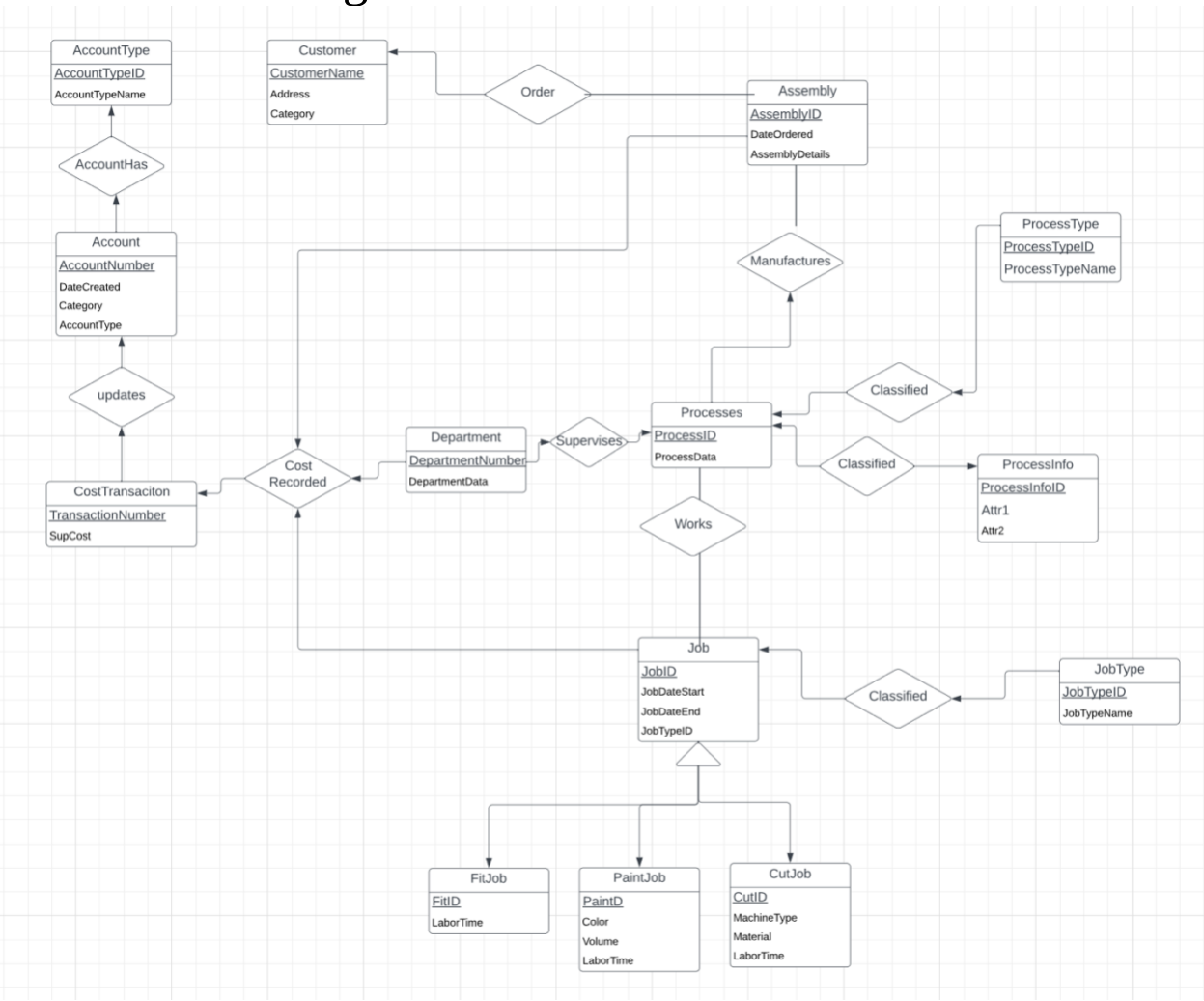
113591849

Gabriel.d.owen-1@ou.edu

DBMS Individual Project

Tasks Performed

TASK 1: ER diagram



TASK 2:

- Customer(CustomerName: string, Address: string, Category: int)
- Assembly(AssemblyID:int, DateOrdered: date, AssemblyDetails: string, CustomerName: String, AccountNumber: int)
- AssemblyXREF(ProcessID: int, AssemblyID: int)
- Process(ProcessID: int,ProcessData: String, AssemblyID: int, ProcessInfoID: int, ProcessTypeID: int, DepertmentNumber: int, accountNumber: int)
- ProcessType(ProcessTypeID: int, ProcessType: string)
- ProcessInfo(ProcessInfoID: int, Attr1: String, Attr2: String)
- Job(JobID: int, JobDateStart: Date, JobDateEnd: Date, JobTypeID: int, ProcessID: int, assemblyID: int, LaborTime:float)
- JobType(JobTypeID: int, JobTypeName: string)
- JobFit(JobID: int)
- JobPaint(JobID: int, Color: string, Volume: float)
- JobCut(JobID: int, MachineType: string, Material: string)
- CostTransaction(TransactionNumber: int, SupCost: float, JobID: int, ProcessAccountID: int, DepartmentAccountID: int, AssemblyAccountID: int)
- Account(AccountNumber: Int, DateCreated: Date, Details: float, AccountTypeID: int)
- AccountType(AccountTypeID: int, AccountTypeName: string)
- Department(DepartmentNumber: int, DepartmentData: string, AccountNumber: int)

TASK 3: Storage Structures

Table Name	Query Number and Type	SearchKey	Query Frequency	Selected File Organization	Justifications
Account	5 INSERT 8 INSERT 9 JOIN	AccountID	10/Day 50/day 200/day	clustered	Access mutliple different tables
AccountType				Heap	Only need account table
Assembly	4 INSERT 5 UPDATE 6 SELECT 8 JOIN 9 JOIN 11 JOIN	AssemblyID	40/day 10/day 50/day 50/day 200/day 100/day	Heap	Relates almost soley to xref table
AssemblyXREF	4 INSERT 8 JOIN 11 JOIN	ProcessID, AssemblyID	40/day 50/day 100/day	Clustered	Will be used a lot to join process and assembly
CostTrasaction	8 INSERT 13 DELETE		50/day 1/month		
Customer	1 INSERT 4 SELECT 12 SELECT	CustomerName	30/Day 40/day 100/day	heap	Most frequenlty used just for select
Department	2 INSERT 3 SELECT 5 UPDATE 8 JOIN 10 JOIN	DepartmentNumber	Infrequent Infrequent 10/day 50/day 20/day	heap	Not as many references as other table and usually only selecting one row.
Job	6 INSERT 7 UPDATE 8 JOIN 10 JOIN 11 JOIN 13 DELETE	JobID	50/day 50/day 50/day 20/day 100/day 1/month	clustered	jobID may be the most used and looked up individual attribute
JobCut	7 INSERT 13 DELETE	JobID	17/day 1/month	Heap	Goes only to job table
JobFit	7 INSERT	JobID	17/day	Heap	Goes only to job table
JobPaint	7 INSERT 14 UPDATE	JobID	17/day 1/week	heap	Goes only to job table
Process	3 INSERT 5 UPDATE 6 SELECT 8 JOIN 10 JOIN 11 JOIN	ProcessID	Infrequent 10/day 50/day 50/day 20/day 100/day	heap	Relates almost soley to xref table
ProcessInfo	3 INSERT	ProcessInfoID	Infrequent	Heap	Only access process table
ProcessType	3 INSERT	ProcessTypeID	Infrequent	heap	Only access process table

TASK 4: Create Tables

```sql
-- Create AccountType Table
CREATE TABLE AccountType (
    AccountTypeID INT PRIMARY KEY,
    AccountTypeName VARCHAR(255)
);

-- Insert the types
INSERT INTO AccountType(AccountTypeID, AccountTypeName)
VALUES (1, 'Assembly'), (2, 'Department'), (3, 'Process')
```

Messages

10:27:23 PM Started executing query at Line 20
(3 rows affected)
Total execution time: 00:00:00.067

```sql
-- Create Account Table
CREATE TABLE Account (
    AccountNumber INT PRIMARY KEY,
    DateCreated DATE,
    Details FLOAT,
    AccountTypeID INT,
    FOREIGN KEY (AccountTypeID) REFERENCES AccountType(AccountTypeID)
);
```

Messages

10:27:42 PM Started executing query at Line 30
Commands completed successfully.
Total execution time: 00:00:00.063

```sql
-- Create Customer Table
CREATE TABLE Customer (
    CustomerName VARCHAR(255) PRIMARY KEY,
    [Address] VARCHAR(255),
    Category INT CHECK (Category BETWEEN 1 AND 10),
);
```

Messages

10:27:58 PM Started executing query at Line 39
Commands completed successfully.
Total execution time: 00:00:00.054

```sql
-- Create Assembly Table
CREATE TABLE Assembly (
    AssemblyID INT PRIMARY KEY,
    DateOrdered DATE,
    AssemblyDetails VARCHAR(255),
    CustomerName VARCHAR(255),
    AccountNumber INT,
    FOREIGN KEY (CustomerName) REFERENCES Customer(CustomerName),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);
```

Messages

10:28:17 PM Started executing query at Line 46
Commands completed successfully.
Total execution time: 00:00:00.066

```sql
-- Create Department Table
CREATE TABLE Department (
    DepartmentNumber INT PRIMARY KEY,
    DepartmentData VARCHAR(255),
    AccountNumber INT,
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);
```

Messages

10:28:38 PM Started executing query at Line 57
Commands completed successfully.
Total execution time: 00:00:00.054

```sql
CREATE TABLE ProcessType(
    ProcessTypeID INT PRIMARY KEY,
    ProcessType VARCHAR(20)
)

-- Insert the types
INSERT INTO ProcessType(ProcessTypeID, ProcessType)
VALUES (1, 'Fit'), (2, 'Cut'), (3, 'Paint')
```

Messages

10:28:57 PM Started executing query at Line 65
(3 rows affected)
Total execution time: 00:00:00.078

```sql
CREATE TABLE ProcessInfo (
    ProcessInfoID INT PRIMARY KEY,
    Attr1 VARCHAR(255),
    Attr2 VARCHAR(255)
);
```

Messages

10:29:27 PM Started executing query at Line 74
Commands completed successfully.
Total execution time: 00:00:00.055

```sql
-- Create Process Table
CREATE TABLE Process (
    ProcessID INT PRIMARY KEY,
    AssemblyID INT,
    ProcessData VARCHAR(255),
    ProcessInfoID INT,
    ProcessTypeID INT,
    DepartmentNumber INT,
    AccountNumber INT,
    FOREIGN KEY (AssemblyID) REFERENCES [Assembly](AssemblyID),
    FOREIGN KEY (DepartmentNumber) REFERENCES Department(DepartmentNumber),
    FOREIGN KEY (ProcessTypeID) REFERENCES ProcessType(ProcessTypeID),
    FOREIGN KEY (ProcessInfoID) REFERENCES ProcessInfo(ProcessInfoID),
    FOREIGN KEY (AccountNumber) REFERENCES Account(AccountNumber)
);
```

Messages

10:29:41 PM Started executing query at Line 80
Commands completed successfully.
Total execution time: 00:00:00.056

```sql
CREATE TABLE AssemblyXREF (
    ProcessID INT,
    AssemblyID INT,
    PRIMARY KEY (ProcessID, AssemblyID),
    FOREIGN KEY (ProcessID) REFERENCES Process(ProcessID),
    FOREIGN KEY (AssemblyID) REFERENCES Assembly(AssemblyID)

);
```

Messages

10:29:55 PM Started executing query at Line 95
Commands completed successfully.
Total execution time: 00:00:00.061

```sql
CREATE TABLE JobType(
    JobTypeID   INT PRIMARY KEY,
    JobTypeName VARCHAR(50)

)
-- Insert the types
INSERT INTO JobType(JobTypeID, JobTypeName)
VALUES (1, 'Fit'), (2, 'Paint'), (3, 'Cut')
```

Messages

10:30:10 PM Started executing query at Line 105
(3 rows affected)
Total execution time: 00:00:00.062

```
115  CREATE TABLE Job (
116      JobID INT PRIMARY KEY,
117      JobDateStart DATE,
118      JobDateEnd DATE,
119      JobTypeID INT,
120      ProcessID INT,
121      AssemblyID INT,
122      LaborTime FLOAT,
123      FOREIGN KEY (ProcessID) REFERENCES Process(ProcessID),
124      FOREIGN KEY (AssemblyID) REFERENCES Assembly(AssemblyID),
125      CONSTRAINT fkJobType FOREIGN KEY (JobTypeID) REFERENCES JobType(JobTypeID)
126  );
127
```

Messages

10:30:23 PM Started executing query at Line 115
Commands completed successfully.
Total execution time: 00:00:00.057

```
129  -- Create JobFit Table
130  CREATE TABLE JobFit (
131      JobID INT PRIMARY KEY,
132      CONSTRAINT fkFitJob FOREIGN KEY (JobID) REFERENCES Job(JobID)
133  );
134
```

Messages

10:30:40 PM Started executing query at Line 130
Commands completed successfully.
Total execution time: 00:00:00.053

```sql
-- Create JobPaint Table
CREATE TABLE JobPaint (
    JobID INT PRIMARY KEY,
    Color VARCHAR(255),
    Volume FLOAT,

    --FOREIGN KEY (JobID) REFERENCES Job(JobID)
    CONSTRAINT fkPaintJob FOREIGN KEY (JobID) REFERENCES Job(JobID)
);

-- Create JobCut Table
```

Messages

10:30:50 PM Started executing query at Line 135
Commands completed successfully.
Total execution time: 00:00:00.054

```sql
-- Create JobCut Table
CREATE TABLE JobCut (
    JobID INT PRIMARY KEY,
    MachineType VARCHAR(255),
    Material VARCHAR(255),
    MachineTime FLOAT,
    --FOREIGN KEY (JobID) REFERENCES Job(JobID)
    CONSTRAINT fkCutJob FOREIGN KEY (JobID) REFERENCES Job(JobID)
);
```

Messages

10:31:02 PM Started executing query at Line 146
Commands completed successfully.
Total execution time: 00:00:00.053

```
155    -- Create Transaction Table
156    CREATE TABLE CostTransaction (
157        TransactionNumber INT PRIMARY KEY,
158        SupCost FLOAT,
159        JobID INT,
160        ProcessAccountID    INT,
161        DepartmentAccountID    INT,
162        AssemblyAccountID    INT,
163        FOREIGN KEY (JobID) REFERENCES Job(JobID)
164    );
165
```

Messages

10:31:12 PM Started executing query at Line 156
Commands completed successfully.
Total execution time: 00:00:00.054

Task 5: queries

5.1: Stored Proc

Query1

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new customer
--  NewCustomer 'Gabe', 'home', 11
CREATE PROCEDURE [dbo].[usp_InsertCustomer]
    @CustomerName VARCHAR(255),
    @Address VARCHAR(255),
    @Category INT
AS
BEGIN
    -- Check if the customer already exists
```

```
    IF @Category >= 1 AND @Category <= 10
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM Customer WHERE CustomerName = @CustomerName)
        BEGIN
            -- Insert the new customer
            INSERT INTO Customer (CustomerName, [Address], Category)
            VALUES (@CustomerName, @Address, @Category);
        END
    END
END;

GO
```

Query2

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new department
-- usp_CreateNewDepartment 1, 'Datatatatatatatta'
CREATE PROCEDURE [dbo].[usp_CreateNewDepartment]
(
    @DepartmentNumber INT,
    @DepartmentData VARCHAR(255)
)

AS
BEGIN
    -- Check if the department number already exists
    IF NOT EXISTS (SELECT 1 FROM Department WHERE DepartmentNumber = @DepartmentNumber)
    BEGIN
        -- Insert the new department
        INSERT INTO Department (
            DepartmentNumber,
            DepartmentData)
        VALUES (
```

```
        @DepartmentNumber,
        @DepartmentData);

  END
END;
GO
```

Query3

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
-- usp_CreateNewDepartment 1, 'Datatatatatatatta'
-- InsertProcessWithType 3, 1, 1, 2, 'THE FIT'
CREATE PROCEDURE [dbo].[usp_CreateProcess]
(
  @ProcessID        INT,
  @ProcessTypeID    INT,
  @DepartmentNumber  INT,
  @ProcessInfoID    INT,
  @Attr1            VARCHAR(255),
  @Attr2            VARCHAR(255)   =  NULL
)

AS
BEGIN
  IF EXISTS (
    SELECT 1 FROM Department
    WHERE @DepartmentNumber = DepartmentNumber
  )
  BEGIN
    IF NOT EXISTS (SELECT 1 FROM Process WHERE ProcessID = @ProcessID)
      BEGIN
```

```
        -- insert process info
        INSERT INTO ProcessInfo(ProcessInfoID, Attr1, Attr2)
        VALUES (@ProcessInfoID, @Attr1, @Attr2)

        -- Insert into Process table
        INSERT INTO Process (ProcessID, ProcessTypeID, DepartmentNumber, ProcessInfoID)
        VALUES (@ProcessID, @ProcessTypeID, @DepartmentNumber,@ProcessInfoID);

        PRINT 'Process with type information inserted successfully.';
    END
  END
END;
GO
```

Query4

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- usp_InsertCustomer 'Gabe', 'Norman', '1'
-- usp_CreateNewDepartment 1, 'Datatatatatatatta'
-- usp_CreateProcess 2, 1, 1, 2, 'Another type'
-- usp_CreateAssembly 1,  '2023-11-12', 'Details', 'Gabe', 2
CREATE PROCEDURE [dbo].[usp_CreateAssembly]
(
  @AssemblyID       INT,
  @DateOrdered      DATE,
  @AssemblyDetails  VARCHAR(255),
  @CustomerName     VARCHAR(255),
  @ProcessID        INT

)
AS
BEGIN
```

```
    IF EXISTS (SELECT 1 FROM Customer WHERE CustomerName = @CustomerName)
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM [Assembly] WHERE [AssemblyID] = @AssemblyID)
        BEGIN
            INSERT INTO [Assembly] (AssemblyID, CustomerName, DateOrdered, AssemblyDetails)
            VALUES (@AssemblyID, @CustomerName, @DateOrdered, @AssemblyDetails)

            INSERT INTO [AssemblyXREF] (AssemblyID, ProcessID)
                VALUES(@AssemblyID, @ProcessID)
        END
        ELSE
        BEGIN
            INSERT INTO [AssemblyXREF] (AssemblyID, ProcessID)
                    VALUES(@AssemblyID, @ProcessID)
        END
    END


END;
GO
```

Query5

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
/*


EXEC usp_InsertCustomer 'Gabe', 'Norman', '1'
EXEC usp_CreateNewDepartment 1, 'Datatatatatatatta'
EXEC usp_CreateProcess 3, 1, 1, 2, 'Another type'
EXEC usp_CreateAssembly 2,  '2023-11-12', 'Details', 'Gabe', 3

-- Associate with a assembly
EXEC [dbo].[usp_CreateAccount] 1, 2, '2023-11-12', 0, 1
```

```
-- Associate with a department
EXEC [dbo].[usp_CreateAccount] 2, 1, '2023-11-12', 0, 2

-- Associate with a process
EXEC [dbo].[usp_CreateAccount] 3, 3, '2023-11-12', 0, 3
*/
--
--
CREATE PROCEDURE [dbo].[usp_CreateAccount]
(
   @AccountNumber      INT,
   @AssociationID      INT,
   @DateCreated        DATE,
   --@Details            FLOAT,
   @AccountTypeID      INT
)

AS
BEGIN
   IF EXISTS (
      SELECT 1 FROM AccountType
      WHERE @AccountTypeID = AccountTypeID
   )
   BEGIN
      IF( @AccountTypeID = 1)    -- Assembly Account type
      BEGIN
         IF NOT EXISTS (SELECT 1 FROM Account WHERE AccountNumber = @AccountNumber)
         BEGIN
            INSERT INTO Account(AccountNumber, DateCreated, Details, AccountTypeID)
            VALUES (@AccountNumber, @DateCreated, 0, @AccountTypeID)

            UPDATE [dbo].[Assembly]
            SET AccountNumber = @AccountNumber
            WHERE [AssemblyID] = @AssociationID
```

```
    END
    ELSE
    BEGIN
        UPDATE [dbo].[Assembly]
        SET AccountNumber = @AccountNumber
        WHERE [AssemblyID] = @AssociationID
    END
END
IF( @AccountTypeID = 2) -- Department
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Account WHERE AccountNumber = @AccountNumber)
    BEGIN
        INSERT INTO Account(AccountNumber, DateCreated, Details, AccountTypeID)
        VALUES (@AccountNumber, @DateCreated, 0, @AccountTypeID)

        UPDATE Department
        SET AccountNumber = @AccountNumber
        WHERE DepartmentNumber = @AssociationID


    END
    BEGIN
        UPDATE Department
        SET AccountNumber = @AccountNumber
        WHERE DepartmentNumber = @AssociationID

    END
END
IF( @AccountTypeID = 3) -- Process
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Account WHERE AccountNumber = @AccountNumber)
    BEGIN
        INSERT INTO Account(AccountNumber, DateCreated, Details, AccountTypeID)
        VALUES (@AccountNumber, @DateCreated, 0, @AccountTypeID)

        UPDATE Process
        SET AccountNumber = @AccountNumber
```

```
            WHERE ProcessID = @AssociationID

        END
        BEGIN
            UPDATE Process
            SET AccountNumber = @AccountNumber
            WHERE ProcessID = @AssociationID
        END
    END
  END
END;
GO
```

Query6

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
/*


EXEC usp_InsertCustomer 'Gabe', 'Norman', '1'
EXEC usp_CreateNewDepartment 1, 'Datatatatatatatta'
EXEC usp_CreateProcess 3, 1, 1, 2, 'Another type'
EXEC usp_CreateAssembly 2,  '2023-11-12', 'Details', 'Gabe', 3

-- creates fit job
EXEC [dbo].[usp_CreateJob] 1, 2, 3, '2023-11-12'

-- creates paint job
EXEC [dbo].[usp_CreateJob] 2, 2, 3, '2023-11-12'

-- creates cut job
EXEC [dbo].[usp_CreateJob] 3, 2, 3, '2023-11-12'
```

```
*/
--
--
CREATE PROCEDURE [dbo].[usp_CreateJob]
(
   @JobID             INT,
   @AssemblyID        INT,
   @ProcessID         INT,
   @JobDateStart      DATE
)

AS
BEGIN
   IF NOT EXISTS (SELECT 1 FROM Job WHERE JobID = @JobID)
   BEGIN
      IF EXISTS (SELECT 1 FROM Process WHERE ProcessID  = @ProcessID )
      BEGIN
         IF EXISTS (SELECT 1 FROM [Assembly] WHERE AssemblyID = @AssemblyID )
         BEGIN
               INSERT INTO Job(JobID, AssemblyID, ProcessID, JobDateStart)
               VALUES (@JobID, @AssemblyID, @ProcessID, @JobDateStart)
               PRINT 'INSERTED JOB'

         END
         ELSE
         PRINT 'Assembly Doesnt Exist'
      END
      ELSE
         PRINT 'Process Doesnt Exist'
   END

   ELSE
   BEGIN
      IF EXISTS (SELECT 1 FROM Process WHERE @ProcessID = ProcessID)
      BEGIN
         IF EXISTS (SELECT 1 FROM [Assembly] WHERE @AssemblyID = AssemblyID)
         BEGIN
```

```
            UPDATE [dbo].[Job]
            SET JobID = @JobID,
               AssemblyID = @AssemblyID,
               ProcessID = @ProcessID,
               JobDateStart = @JobDateStart
            WHERE [JobID] = @JobID
         END
         ELSE
         PRINT 'Assembly Doesnt Exist'
      END
      ELSE
         PRINT 'Process Doesnt Exist'
      END

END;
GO
```

Query7

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
/*


EXEC usp_InsertCustomer 'Gabe', 'Norman', '1'
EXEC usp_CreateNewDepartment 1, 'Datatatatatatatta'
EXEC usp_CreateProcess 3, 1, 1, 2, 'Another type'
EXEC usp_CreateAssembly 2,  '2023-11-12', 'Details', 'Gabe', 3

-- creates fit job
EXEC [dbo].[usp_CompleteJob] 1, 1 , '2023-11-12', 0.69

-- creates paint job
```

```
EXEC [dbo].[usp_CompleteJob] 2, 2, '2023-11-12', 0.69, 'BLUE', 100

-- creates cut job
EXEC [dbo].[usp_CompleteJob] 3, 3, '2023-11-12', 0.69, null, null, 'CUTTER2000', 12, 'Titanium'
*/
--
--
CREATE PROCEDURE [dbo].[usp_CompleteJob]
(
   @JobID            INT,
   @JobTypeID        INT,
   @JobDateEnd       DATE,
   @LaborTime        FLOAT,
   @Color            VARCHAR(255) = NULL,
   @Volume           FLOAT = NULL,
   @MachineType      VARCHAR(255)= NULL,
   @MachineTime      FLOAT = NULL,
   @Material         VARCHAR(255) = NULL
)

AS
BEGIN
   IF EXISTS (SELECT 1 FROM Job WHERE JobID = @JobID)
   BEGIN
         UPDATE Job
         SET JobID = @JobID,
            JobDateEnd = @JobDateEnd,
            JobTypeID = @JobTypeID,
            LaborTime = @LaborTime
         WHERE JobID = @JobID
         PRINT 'UPDATE JOB'
         IF(@JobTypeID = 1)
         BEGIN
            PRINT 'Inserting job fit type'
            INSERT INTO JobFit(JobID)
            VALUES (@JobID)
         END
```

```
        IF(@JobTypeID = 2)
        BEGIN
            PRINT 'Inserting job paint type'
            INSERT INTO JobPaint(JobID, Color, Volume)
            VALUES (@JobID, @Color, @Volume)
        END
        IF(@JobTypeID = 3)
        BEGIN
            PRINT 'Inserting job cut type'
            INSERT INTO JobCut(JobID, MachineType, MachineTime, Material)
            VALUES (@JobID, @MachineType, @MachineTime, @Material)
        END
    END
    ELSE
        PRINT('No Job Exists for that number')

END;
GO
```

Query8

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
/*


EXEC usp_InsertCustomer 'Gabe', 'Norman', '1'
EXEC usp_CreateNewDepartment 1, 'Datatatatatatatta'
EXEC usp_CreateProcess 3, 1, 1, 2, 'Another type'
EXEC usp_CreateAssembly 2,  '2023-11-12', 'Details', 'Gabe', 3

-- Create accounts
EXEC [dbo].[usp_CreateAccount] 1, 2, '2023-11-12', 0, 1
```

```
EXEC [dbo].[usp_CreateAccount] 2, 1, '2023-11-12', 0, 2
EXEC [dbo].[usp_CreateAccount] 3, 3, '2023-11-12', 0, 3

EXEC [dbo].[usp_CreateJob] 1, 2, 3, '2023-11-12'
EXEC [dbo].[usp_CreateJob] 2, 2, 3, '2023-11-12'
EXEC [dbo].[usp_CreateJob] 3, 2, 3, '2023-11-12'

EXEC [dbo].[usp_CreateTransaction] 11, 9, 1
*/
--
--
CREATE PROCEDURE [dbo].[usp_CreateTransaction]
(
   @TransactionNumber     INT,
   @JobID                 INT,
   @SupCost               FLOAT
)

AS
BEGIN

  INSERT INTO CostTransaction (TransactionNumber,JobID, SupCost)
  VALUES(@TransactionNumber, @JobID, @SupCost)

  DECLARE @ProcessAccountNumber       INT;
  DECLARE @DepartmentAccountNumber    INT;
  DECLARE @AssemblyAccountNumber      INT;


  SELECT DISTINCT @ProcessAccountNumber = p.AccountNumber
  FROM Process p
  INNER JOIN Job j
  ON p.ProcessID = j.ProcessID
  INNER JOIN Account a
  ON p.AccountNumber = a.AccountNumber
  WHERE 1 = j.JobID
```

```
  SELECT DISTINCT @DepartmentAccountNumber = d.AccountNumber
  FROM Process p
  INNER JOIN Job j
  ON p.ProcessID = j.ProcessID
  INNER JOIN Department d
  ON d.DepartmentNumber = d.DepartmentNumber
  WHERE 1 = j.JobID

   SELECT DISTINCT @AssemblyAccountNumber = a.AccountNumber
   FROM Process p
   INNER JOIN Job j
      ON p.ProcessID = j.ProcessID
   INNER JOIN AssemblyXREF x
      ON x.ProcessID = p.ProcessID
   INNER JOIN [Assembly] a
      ON x.AssemblyID = a.AssemblyID
   WHERE 1 = j.JobID



  PRINT @ProcessAccountNumber
  PRINT @DepartmentAccountNumber
  PRINT @AssemblyAccountNumber

UPDATE Account
  SET Details += @SupCost
  WHERE AccountNumber = @ProcessAccountNumber

UPDATE Account
  SET Details += @SupCost
  WHERE AccountNumber = @DepartmentAccountNumber

UPDATE Account
  SET Details += @SupCost
  WHERE AccountNumber = @AssemblyAccountNumber
```

```
END;
GO
```

Query9

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- usp_InsertCustomer 'Gabe', 'Norman', '1'
-- usp_CreateNewDepartment 1, 'Datatatatatatatta'
-- usp_CreateProcess 2, 1, 1, 2, 'Another type'
-- usp_CreateAssembly 1,  '2023-11-12', 'Details', 'Gabe', 2

CREATE PROCEDURE [dbo].[usp_GetAssemblyCost]
(
   @AssemblyID        INT


)
AS
BEGIN
   SELECT ac.AccountNumber,
      a.AssemblyID,
      ac.Details
   FROM Account ac
   INNER JOIN [Assembly] a
      ON a.AccountNumber = ac.AccountNumber
   Where a.AssemblyID = @AssemblyID

END;
GO
```

Query10

```
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
-- Create a stored procedure to insert a new process with type information
/*


EXEC usp_InsertCustomer 'Gabe', 'Norman', '1'
EXEC usp_CreateNewDepartment 1, 'Datatatatatatatta'
EXEC usp_CreateProcess 3, 1, 1, 2, 'Another type'
EXEC usp_CreateAssembly 2,  '2023-11-12', 'Details', 'Gabe', 3

-- creates fit job
EXEC [dbo].[usp_CreateJob] 1, 2, 3, '2023-11-12'

-- creates paint job
EXEC [dbo].[usp_CreateJob] 2, 2, 3, '2023-11-12'

-- creates cut job
EXEC [dbo].[usp_CreateJob] 4, 2, 3, '2023-11-12'

EXEC [dbo].[usp_CompleteJob] 1, 1 , '2023-11-12', 0.69
EXEC [dbo].[usp_CompleteJob] 2, 2 , '2023-11-12', 10
EXEC [dbo].[usp_CompleteJob] 4, 3 , '2024-11-12', 200

EXEC [dbo].[usp_GetDepartmentTime] 1,  '2023-12-12'
*/
--
--
CREATE PROCEDURE [dbo].[usp_GetDepartmentTime]
(
    @DepartmentNumber   INT,
    @StartDate          DATE = NULL,
    @EndDate            DATE = NULL
)

AS
```

```
BEGIN
  SELECT SUM(j.LaborTime)
  FROM Department d
  INNER JOIN Process p
  ON d.DepartmentNumber = p.DepartmentNumber
  INNER JOIN Job j
  ON j.ProcessID = p.ProcessID
  WHERE (j.JobDateEnd >= @StartDate OR @StartDate IS NULL)
  AND (j.JobDateEnd  <= @EndDate OR @EndDate IS NULL)
  AND d.DepartmentNumber = @DepartmentNumber

END;
GO
```

Query11

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- [dbo].[usp_GetCompletedProcesses] 1

CREATE PROCEDURE [dbo].[usp_GetCompletedProcesses]
(
   @AssemblyID        INT
)
AS
BEGIN
   SELECT DISTINCT p.ProcessID,
      p.DepartmentNumber,
      j.JobID,
      j.JobDateEnd
   FROM [Assembly] a
   INNER JOIN [AssemblyXREF] x
      ON a.AssemblyID = x.AssemblyID
   INNER JOIN Process p
```

```
    ON p.ProcessID = x.ProcessID
  INNER JOIN Job j
    ON j.ProcessID = p.ProcessID
  WHERE j.JobDateEnd IS NOT NULL
    AND a.AssemblyID = @AssemblyID
  ORDER BY j.JobDateEnd ASC

END;
GO
```

Query12

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE [dbo].[usp_GetCustomer]
(
  @CategoryMin       INT = NULL,
  @CategoryMax       INT = NULL
)
AS
BEGIN
  SELECT CustomerName,
    [Address],
    Category
  FROM Customer
  Where (Category >= @CategoryMin OR @CategoryMin IS NULL) AND
    (Category <= @CategoryMax OR @CategoryMax IS NULL)

END;
GO
```

Query13

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE [dbo].[usp_DeleteJobs]
(
   @JobIDMin       INT,
   @JobIDMax       INT
)
AS
BEGIN
   DELETE FROM JobCut
   WHERE (JobID >= @JobIDMin)
      AND (JobID <= @JobIDMax)
   DELETE FROM CostTransaction
   WHERE (JobID >= @JobIDMin)
      AND (JobID <= @JobIDMax)
   DELETE FROM Job
   WHERE (JobID >= @JobIDMin)
      AND (JobID <= @JobIDMax)
      AND JobTypeID = 3

END;
GO
```

Query14

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE [dbo].[usp_ChangeColor]
(
   @JobID         INT,
   @Color         VARCHAR(255)
)
AS
BEGIN
   IF EXISTS(SELECT 1 FROM JobPaint WHERE JobID = @JobID)
   BEGIN
      UPDATE JobPaint
      SET Color = @Color
      WHERE JobID = @JobID
   END
   ELSE
   BEGIN
     PRINT('Selected Job is not of Job:Paint type')
   END

END;
GO
```

Queries 15 and 16 are equivaluent to Queries 1 and 12 respectivly.

5.2: execution

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

Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
```

Task 6: JavaExecution

Task 6.1: create customers

```
1
Please enter Customer Name:
Gabe
Please enter Address:
Norman
Please enter Category 1-10:
1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1   SELECT TOP (1000) [CustomerName]
2         ,[Address]
3         ,[Category]
4     FROM [dbo].[Customer]
```

Results Messages

Results grid

	CustomerName	Addr	Category
1	Gabe	Norman	1
2	Jordan	Norman	8
3	Natalie	Broken Arrow	10
4	Richard	Catoosa	1
5	Tim	Tulsa	7

Task 6.2: create departments

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
2
Please enter Department Number:
1
Please enter Department Data:
This is the first department.
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1  SELECT TOP (1000) [DepartmentNumber]
2        ,[DepartmentData]
3        ,[AccountNumber]
4    FROM [dbo].[Department]
```

Results | Messages

	DepartmentNumber	DepartmentData	AccountNumber
1	1	This is the first department.	NULL
2	2	This is the second for monitoring	NULL
3	3	The Tour guide department	NULL
4	4	The hard working department	NULL
5	5	The best for last	NULL

Task 6.3: create processes

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
3
Please enter Process ID:
1
Please enter Process Type (1:Fit, 2:Paint, 3:Cut):
1
Please enter Department Number:
1
Please enter Fit Type:
Loose
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
SELECT TOP (1000) [ProcessID]
      ,[AssemblyID]
      ,[ProcessData]
      ,[ProcessInfoID]
      ,[ProcessTypeID]
      ,[DepartmentNumber]
      ,[AccountNumber]
  FROM [dbo].[Process]
```

Results Messages

	ProcessID	AssemblyID	ProcessData	ProcessInfoID	ProcessTypeID	DepartmentNumber	AccountNumber
1	1	NULL	NULL	1	1	1	NULL
2	2	NULL	NULL	2	2	1	NULL
3	3	NULL	NULL	3	3	3	NULL
4	4	NULL	NULL	4	3	5	NULL
5	5	NULL	NULL	5	3	5	NULL
6	6	NULL	NULL	6	1	5	NULL
7	7	NULL	NULL	7	2	4	NULL
8	8	NULL	NULL	8	3	2	NULL
9	9	NULL	NULL	9	1	1	NULL
10	10	NULL	NULL	10	2	5	NULL

Task 6.4: create assemblies

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
4
Please enter Assembly ID:
1
Please enter Date Ordered (YYYY-MM-DD):
2022
Please enter Assembly Details:
Need more of these
Please enter Customer Name:
Gabe
Please enter Associated ProcessID:
1
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1   SELECT TOP (1000) [AssemblyID]
2         ,[DateOrdered]
3         ,[AssemblyDetails]
4         ,[CustomerName]
5         ,[AccountNumber]
6     FROM [dbo].[Assembly]
```

Results Messages

	AssemblyID	DateOrdered	AssemblyDetails	CustomerName	AccountNumber
1	1	2022-01-01	Need more of these	Gabe	NULL
2	2	2023-01-01	New Year Resupply needed	Tim	NULL
3	3	2025-01-01	I come from the future	Richard	NULL
4	4	2022-04-22	a birthday present	Jordan	NULL
5	5	2023-10-30	Halloween Costumes	Natalie	NULL
6	6	2022-12-20	Lase years christmas order	Gabe	NULL
7	7	2002-04-22	Childhood order	Richard	NULL
8	8	2023-01-01	New Vehicle	Tim	NULL
9	9	2025-01-01	Rocket ships	Jordan	NULL
10	10	2022-01-01	McDonalds Cups	Tim	NULL

```
1   SELECT TOP (1000) [ProcessID]
2         ,[AssemblyID]
3     FROM [dbo].[AssemblyXREF]
```

Results Messages

	ProcessID	AssemblyID
1	1	1
2	2	3
3	2	5
4	4	4
5	4	7
6	5	10
7	6	6
8	7	2
9	8	8
10	10	9

Task 6.5: Create accounts

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
5
Please enter Account Number:
1
Please enter Account Type (1:Assembly, 2:Department, 3:Process):
1
Please enter AssemblyID:
1
Please enter Todays Date (YYYY-MM-DD):
2023
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1   SELECT TOP (1000) [AccountNumber]
2         ,[DateCreated]
3         ,[Details]
4         ,[AccountTypeID]
5     FROM [dbo].[Account]
```

Results Messages

	AccountNumber	DateCreated	Details	AccountTypeID
1	1	2023-01-01	0	1
2	2	2022-01-01	0	2
3	3	2021-01-01	0	3
4	4	2021-01-01	0	1
5	5	2021-01-01	0	3
6	6	2023-01-01	0	1
7	7	2023-12-12	0	2
8	8	2020-01-01	0	2
9	9	2019-01-01	0	3
10	10	2024-01-01	0	1

Results grid

```
1  SELECT TOP (1000) [AssemblyID]
2        ,[DateOrdered]
3        ,[AssemblyDetails]
4        ,[CustomerName]
5        ,[AccountNumber]
6    FROM [dbo].[Assembly]
```

Results Messages

	AssemblyID	DateOrdered	AssemblyDetails	CustomerName	AccountNumber
1	1	2022-01-01	Need more of these	Gabe	1
2	2	2023-01-01	New Year Resupply needed	Tim	NULL
3	3	2025-01-01	I come from the future	Richard	NULL
4	4	2022-04-22	a birthday present	Jordan	4
5	5	2023-10-30	Halloween Costumes	Natalie	NULL
6	6	2022-12-20	Lase years christmas order	Gabe	NULL
7	7	2002-04-22	Childhood order	Richard	NULL
8	8	2023-01-01	New Vehicle	Tim	10
9	9	2025-01-01	Rocket ships	Jordan	NULL
10	10	2022-01-01	McDonalds Cups	Tim	6

Results

```
SELECT TOP (1000) [DepartmentNumber]
      ,[DepartmentData]
      ,[AccountNumber]
  FROM [dbo].[Department]
```

Results Messages

	DepartmentNumber	DepartmentData	AccountNumber
1	1	This is the first department.	NULL
2	2	This is the second for monitoring	8
3	3	The Tour guide department	NULL
4	4	The hard working department	2
5	5	The best for last	NULL

dbo.Job

```
SELECT TOP (1000) [ProcessID]
      ,[AssemblyID]
      ,[ProcessData]
      ,[ProcessInfoID]
      ,[ProcessTypeID]
      ,[DepartmentNumber]
      ,[AccountNumber]
  FROM [dbo].[Process]
```

Results Messages

	ProcessID	AssemblyID	ProcessData	ProcessInfoID	ProcessTypeID	DepartmentNumber	AccountNumber
1	1	NULL	NULL	1	1	1	5
2	2	NULL	NULL	2	2	1	NULL
3	3	NULL	NULL	3	3	3	9
4	4	NULL	NULL	4	3	5	NULL
5	5	NULL	NULL	5	3	5	NULL
6	6	NULL	NULL	6	1	5	NULL
7	7	NULL	NULL	7	2	4	NULL
8	8	NULL	NULL	8	3	2	NULL
9	9	NULL	NULL	9	1	1	3
10	10	NULL	NULL	10	2	5	NULL

Results grid

Task 6.6: Create Jobs

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
6
Please enter new Job ID:
10
Please enter AssemblyID:
3
Please enter Process ID:
7
Please enter Job start Date (YYYY-MM-DD):
2022
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
SELECT TOP (1000) [JobID]
      ,[JobDateStart]
      ,[JobDateEnd]
      ,[JobTypeID]
      ,[ProcessID]
      ,[AssemblyID]
      ,[LaborTime]
  FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	NULL	NULL	1	1	NULL
2	2	2021-01-01	NULL	NULL	2	1	NULL
3	3	2023-01-01	NULL	NULL	7	4	NULL
4	4	2023-01-01	NULL	NULL	1	9	NULL
5	5	2023-01-01	NULL	NULL	5	8	NULL
6	6	2023-01-01	NULL	NULL	2	6	NULL
7	7	2023-01-01	NULL	NULL	1	10	NULL
8	8	2023-01-01	NULL	NULL	3	3	NULL
9	9	2023-01-01	NULL	NULL	3	10	NULL
10	10	2022-01-01	NULL	NULL	7	3	NULL

Task 6.7: Complete Jobs

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
7
Please enter Job ID:
1
Please enter Job End Date (YYYY-MM-DD):
2024
Please enter Job Type ID:
1
Please enter Labor Time:
12
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1  SELECT TOP (1000) [JobID]
2        ,[JobDateStart]
3        ,[JobDateEnd]
4        ,[JobTypeID]
5        ,[ProcessID]
6        ,[AssemblyID]
7        ,[LaborTime]
8    FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	2024-01-01	1	1	1	12
2	2	2021-01-01	2024-01-01	2	2	1	20
3	3	2023-01-01	2024-01-01	3	7	4	100
4	4	2023-01-01	2024-01-01	3	1	9	12
5	5	2023-01-01	2024-01-01	2	5	8	10
6	6	2023-01-01	2024-06-01	3	2	6	40
7	7	2023-01-01	2025-01-01	1	1	10	1
8	8	2023-01-01	2025-01-01	2	3	3	15
9	9	2023-01-01	2026-01-01	1	3	10	10
10	10	2022-01-01	2025-01-01	3	7	3	40

Task 6.8: insert transaction

```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
8
Please enter new Transaction Number:
1
Please enter Job Number:
1
Please enter SupCost:
100
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1   SELECT TOP (1000) [TransactionNumber]
2         ,[SupCost]
3         ,[JobID]
4         ,[ProcessAccountID]
5         ,[DepartmentAccountID]
6         ,[AssemblyAccountID]
7     FROM [dbo].[CostTransaction]
```

Results | Messages

	TransactionNumber	SupCost	JobID
1	1	100	1
2	2	0.5	2
3	3	59.36000061035156	3
4	4	900	4
5	5	500	1
6	6	105.25	4
7	7	1000000	10
8	8	202.75999450683594	6
9	9	10	7
10	10	20	6

Task 6.9: Get Assembly Costs

```
9
Please enter Assembly Number:
1
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
AccountNumber | AssemblyID | Details
1 | 1 | 0.00
Query executed successfully.
9
Please enter Assembly Number:
10
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
AccountNumber | AssemblyID | Details
6 | 10 | 1002097.88
Query executed successfully.
9
Please enter Assembly Number:
1
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
AccountNumber | AssemblyID | Details
1 | 1 | 3.00
Query executed successfully.
```

Task 6.10: get department hours

```
10
Please enter Department Number:
3
Please enter the start of the date range (YYYY-MM-DD):
2022
Please enter the end of the date range (YYYY-MM-DD):
2025
Connecting to the database...
Dispatching the query...
Department Labor hours:
| 15.00 |
Query executed successfully.
```

```
10
Please enter Department Number:
4
Please enter the start of the date range (YYYY-MM-DD):
2010
Please enter the end of the date range (YYYY-MM-DD):
3000
Connecting to the database...
Dispatching the query...
Department Labor hours:
| 140.00 |
Query executed successfully.
```

```
10
Please enter Department Number:
1
Please enter the start of the date range (YYYY-MM-DD):
2020
Please enter the end of the date range (YYYY-MM-DD):
2025
Connecting to the database...
Dispatching the query...
Department Labor hours:
| 85.00 |
Query executed successfully.
```

Task 6.11: get completed processes for assembly

```
11
Please enter Assembly Number:
1
Connecting to the database...
Dispatching the query...
Contents completed processes for that assembly:
ProcessID | Department No. | JobID | Date Ended
1 | 1 | 1 | 2024-01-01
1 | 1 | 4 | 2024-01-01
1 | 1 | 7 | 2025-01-01
Query executed successfully.
```

```
11
Please enter Assembly Number:
2
Connecting to the database...
Dispatching the query...
Contents completed processes for that assembly:
ProcessID | Department No. | JobID | Date Ended
7 | 4 | 3 | 2024-01-01
7 | 4 | 10 | 2025-01-01
Query executed successfully.
```

```
11
Please enter Assembly Number:
3
Connecting to the database...
Dispatching the query...
Contents completed processes for that assembly:
ProcessID | Department No. | JobID | Date Ended
2 | 1 | 2 | 2024-01-01
2 | 1 | 6 | 2024-06-01
Query executed successfully.
```

Task 6.12: Get customers by category

```
12
Please enter Catergory Lower bound 1-10:
1
Please enter Catergory Upper bound 1-10:
5
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
Name | Addresss | Category
Gabe | Norman | 1
Richard | Catoosa | 1
Query executed successfully.
```

```
12
Please enter Catergory Lower bound 1-10:
5
Please enter Catergory Upper bound 1-10:
10
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
Name | Addresss | Category
Jordan | Norman | 8
Natalie | Broken Arrow | 10
Tim | Tulsa | 7
Query executed successfully.
```

```
12
Please enter Catergory Lower bound 1-10:
1
Please enter Catergory Upper bound 1-10:
10
Connecting to the database...
Dispatching the query...
Contents of the Assembly Cost:
Name | Addresss | Category
Gabe | Norman | 1
Jordan | Norman | 8
Natalie | Broken Arrow | 10
Richard | Catoosa | 1
Tim | Tulsa | 7
Query executed successfully.
```

Task 6.13: Delete Cut jobs

```
SELECT TOP (1000) [JobID]
      ,[JobDateStart]
      ,[JobDateEnd]
      ,[JobTypeID]
      ,[ProcessID]
      ,[AssemblyID]
      ,[LaborTime]
  FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	2024-01-01	1	1	1	12
2	2	2021-01-01	2024-01-01	2	2	1	20
3	3	2023-01-01	2024-01-01	3	7	4	100
4	4	2023-01-01	2024-01-01	3	1	9	12
5	5	2023-01-01	2024-01-01	2	5	8	10
6	6	2023-01-01	2024-06-01	3	2	6	40
7	7	2023-01-01	2025-01-01	1	1	10	1
8	8	2023-01-01	2025-01-01	2	3	3	15
9	9	2023-01-01	2026-01-01	1	3	10	10
10	10	2022-01-01	2025-01-01	3	7	3	40

```
13
Please enter JobID Lower bound:
1
Please enter JobID Upper bound:
5
Connecting to the database...
Dispatching the query...
Done. 2 rows inserted.
```

```
1  SELECT TOP (1000) [JobID]
2        ,[JobDateStart]
3        ,[JobDateEnd]
4        ,[JobTypeID]
5        ,[ProcessID]
6        ,[AssemblyID]
7        ,[LaborTime]
8    FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	2024-01-01	1	1	1	12
2	2	2021-01-01	2024-01-01	2	2	1	20
3	5	2023-01-01	2024-01-01	2	5	8	10
4	6	2023-01-01	2024-06-01	3	2	6	40
5	7	2023-01-01	2025-01-01	1	1	10	1
6	8	2023-01-01	2025-01-01	2	3	3	15
7	9	2023-01-01	2026-01-01	1	3	10	10
8	10	2022-01-01	2025-01-01	3	7	3	40

```
13
Please enter JobID Lower bound:
1
Please enter JobID Upper bound:
1
Connecting to the database...
Dispatching the query...
Done. 0 rows inserted.
```

```sql
SELECT TOP (1000) [JobID]
      ,[JobDateStart]
      ,[JobDateEnd]
      ,[JobTypeID]
      ,[ProcessID]
      ,[AssemblyID]
      ,[LaborTime]
  FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	2024-01-01	1	1	1	12
2	2	2021-01-01	2024-01-01	2	2	1	20
3	5	2023-01-01	2024-01-01	2	5	8	10
4	6	2023-01-01	2024-06-01	3	2	6	40
5	7	2023-01-01	2025-01-01	1	1	10	1
6	8	2023-01-01	2025-01-01	2	3	3	15
7	9	2023-01-01	2026-01-01	1	3	10	10
8	10	2022-01-01	2025-01-01	3	7	3	40

```
13
Please enter JobID Lower bound:
6
Please enter JobID Upper bound:
7
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```sql
SELECT TOP (1000) [JobID]
      ,[JobDateStart]
      ,[JobDateEnd]
      ,[JobTypeID]
      ,[ProcessID]
      ,[AssemblyID]
      ,[LaborTime]
  FROM [dbo].[Job]
```

Results Messages

	JobID	JobDateStart	JobDateEnd	JobTypeID	ProcessID	AssemblyID	LaborTime
1	1	2022-01-01	2024-01-01	1	1	1	12
2	2	2021-01-01	2024-01-01	2	2	1	20
3	5	2023-01-01	2024-01-01	2	5	8	10
4	7	2023-01-01	2025-01-01	1	1	10	1
5	8	2023-01-01	2025-01-01	2	3	3	15
6	9	2023-01-01	2026-01-01	1	3	10	10
7	10	2022-01-01	2025-01-01	3	7	3	40

Task 6.14: Change color

```
1   SELECT TOP (1000) [JobID]
2         ,[Color]
3         ,[Volume]
4     FROM [dbo].[JobPaint]
```

Results Messages

	JobID	Color	Volume
1	2	green	12
2	5	blue	200
3	8	yellow	150

```
14
Please enter JobID:
2
Please enter New Color:
red
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
14
Please enter JobID:
5
Please enter New Color:
red
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
14
Please enter JobID:
8
Please enter New Color:
red
Connecting to the database...
Dispatching the query...
Done. 1 rows inserted.
```

```
1    SELECT TOP (1000) [JobID]
2          ,[Color]
           ,[Volume]
4      FROM [dbo].[JobPaint]
```

Results grid

Results Messages

	JobID	Color	Volume
1	2	red	12
2	5	red	200
3	8	red	150

Task 6.15: Import CSV of customers

	A	B	C	D
1	Name	Address	Category	
2	Jack	StillWater	9	
3	JJ	Ttown	10	
4	Arnold	Moore	7	
5				
6				
7				

```
15
Please enter csv file location:
Customers.csv
Connecting to the database...
Query executed successfully.
```

```
SELECT TOP (1000) [CustomerName]
      ,[Address]
      ,[Category]
  FROM [dbo].[Customer]
```

Results | Messages

	CustomerName	Address	Category
1	Arnold	Moore	7
2	Gabe	Norman	1
3	Jack	StillWater	9
4	JJ	Ttown	10
5	Jordan	Norman	8
6	Natalie	Broken Arrow	10
7	Richard	Catoosa	1
8	Tim	Tulsa	7

Task 6.16: export csv

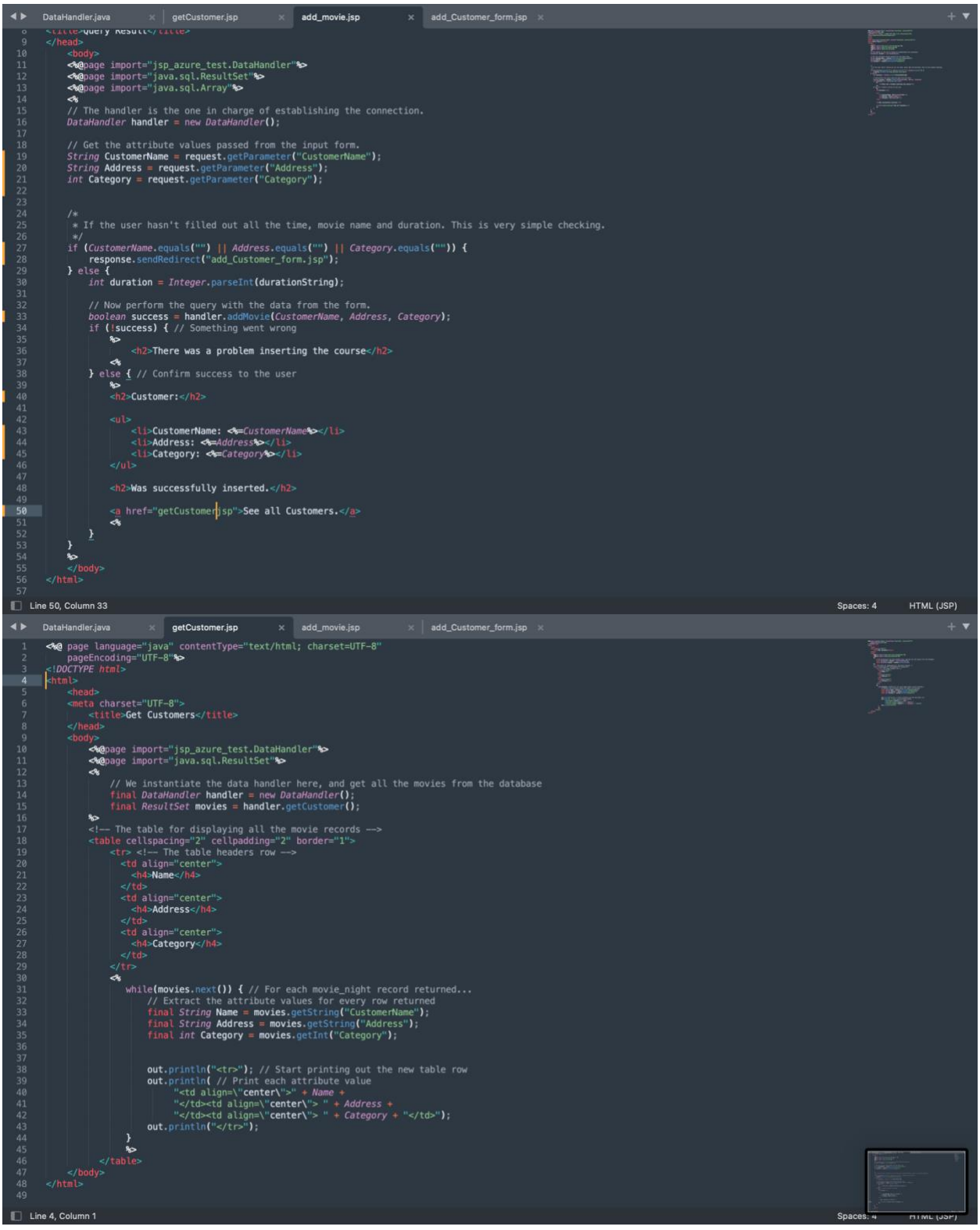
```
16
Please enter Catergory Lower bound 1-10:
1
Please enter Catergory Upper bound 1-10:
10
Please enter output file name:
OUTPUT
Connecting to the database...
Dispatching the query...
Query executed successfully.
```

	A	B	C
	CustomerNa	Address	Category
	Arnold	Moore	7
	Gabe	Norman	1
	Jack	StillWater	9
	JJ	Ttown	10
	Jordan	Norman	8
	Natalie	Broken Arrov	10
	Richard	Catoosa	1
	Tim	Tulsa	7

Task 6.17: Quit

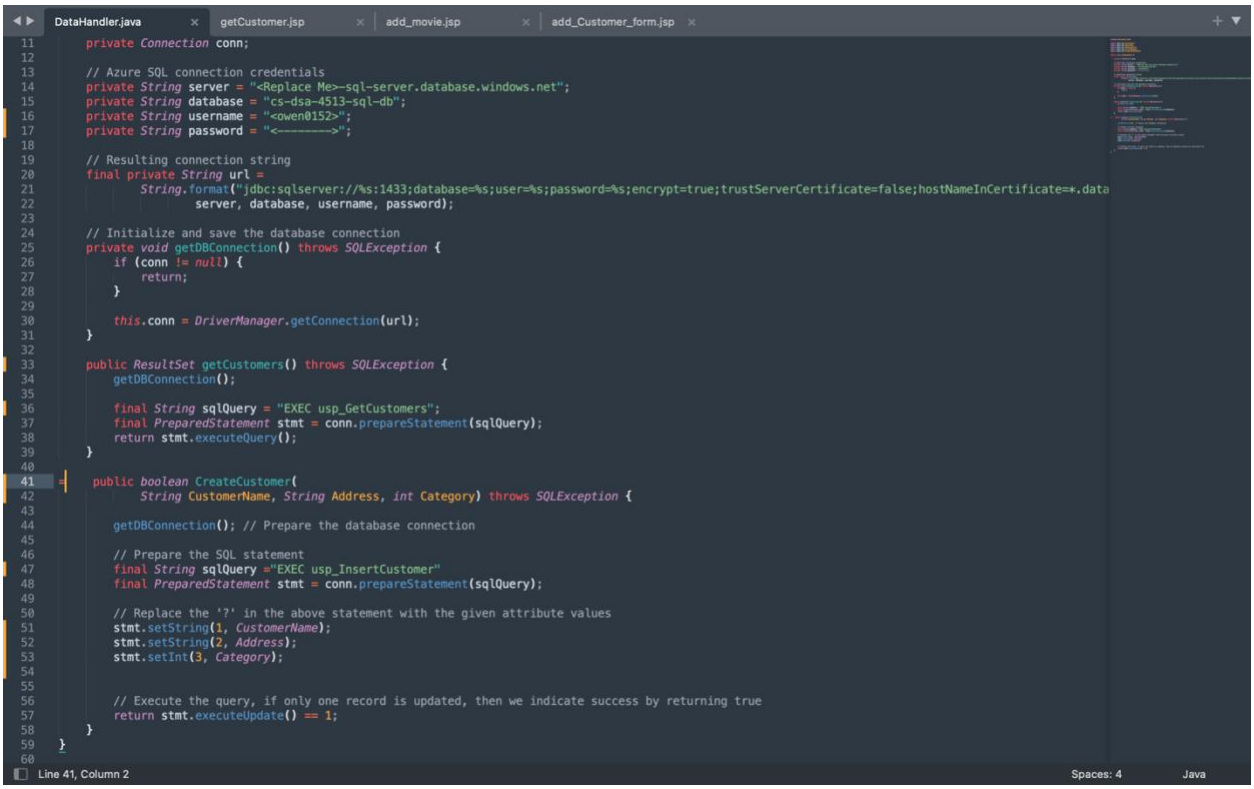
```
Please select one of the options below:
1) Insert new Customer;
2) Insert New Department;
3) Insert new Process;
4) Insert new Assembly;
5) Insert new Account;
6) Insert new Job;
7) Complete Job;
8) Insert Transaction;
9) Get Assembly Cost;
10) Get Department Work Hours;
11) Get completed jobs;
12) Get Customers;
13) Delete Cut Jobs;
14) Change Paint color;
15) Import;
16) Export;
17) Exit!
17
Exiting! Good-buy!
```

Task 7:



```
DataHandler.java | getCustomer.jsp | add_movie.jsp | add_Customer_form.jsp

</head>
    <body>
    <%@page import="jsp_azure_test.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.Array"%>
    <%
    // The handler is the one in charge of establishing the connection.
    DataHandler handler = new DataHandler();

    // Get the attribute values passed from the input form.
    String CustomerName = request.getParameter("CustomerName");
    String Address = request.getParameter("Address");
    int Category = request.getParameter("Category");


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

        // Now perform the query with the data from the form.
        boolean success = handler.addMovie(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:</h2>

            <ul>
                <li>CustomerName: <%=CustomerName%></li>
                <li>Address: <%=Address%></li>
                <li>Category: <%=Category%></li>
            </ul>

            <h2>Was successfully inserted.</h2>

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

Line 50, Column 33                                  Spaces: 4   HTML (JSP)
```

```
DataHandler.java | getCustomer.jsp | add_movie.jsp | add_Customer_form.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Get Customers</title>
    </head>
    <body>
        <%@page import="jsp_azure_test.DataHandler"%>
        <%@page import="java.sql.ResultSet"%>
        <%
            // We instantiate the data handler here, and get all the movies from the database
            final DataHandler handler = new DataHandler();
            final ResultSet movies = handler.getCustomer();
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>Name</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 Name = 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\">" + Name +
                        "</td><td align=\"center\"> " + Address +
                        "</td><td align=\"center\"> " + Category + "</td>");
                   out.println("</tr>");
               }
               %>
          </table>
    </body>
</html>

Line 4, Column 1                                    Spaces: 4   HTML (JSP)
```



```
DataHandler.java | getCustomer.jsp | add_movie.jsp | add_Customer_form.jsp

11      private Connection conn;
12
13      // Azure SQL connection credentials
14      private String server = "<Replace Me>-sql-server.database.windows.net";
15      private String database = "cs-dsa-4513-sql-db";
16      private String username = "<owen0152>";
17      private String password = "<-------->";
18
19      // Resulting connection string
20      final private String url =
21              String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.data
22                      server, database, username, password);
23
24      // Initialize and save the database connection
25      private void getDBConnection() throws SQLException {
26          if (conn != null) {
27              return;
28          }
29
30          this.conn = DriverManager.getConnection(url);
31      }
32
33      public ResultSet getCustomers() throws SQLException {
34          getDBConnection();
35
36          final String sqlQuery = "EXEC usp_GetCustomers";
37          final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
38          return stmt.executeQuery();
39      }
40
41      public boolean CreateCustomer(
42              String CustomerName, String Address, int Category) throws SQLException {
43
44          getDBConnection(); // Prepare the database connection
45
46          // Prepare the SQL statement
47          final String sqlQuery ="EXEC usp_InsertCustomer"
48          final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
49
50          // Replace the '?' in the above statement with the given attribute values
51          stmt.setString(1, CustomerName);
52          stmt.setString(2, Address);
53          stmt.setInt(3, Category);
54
55
56          // Execute the query, if only one record is updated, then we indicate success by returning true
57          return stmt.executeUpdate() == 1;
58      }
59  }
60

Line 41, Column 2                                                    Spaces: 4    Java
```

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

        <form action="addCustomer.jsp">
            <table border=1>
                <tr>
                    <th colspan="2">Enter 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>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>
```