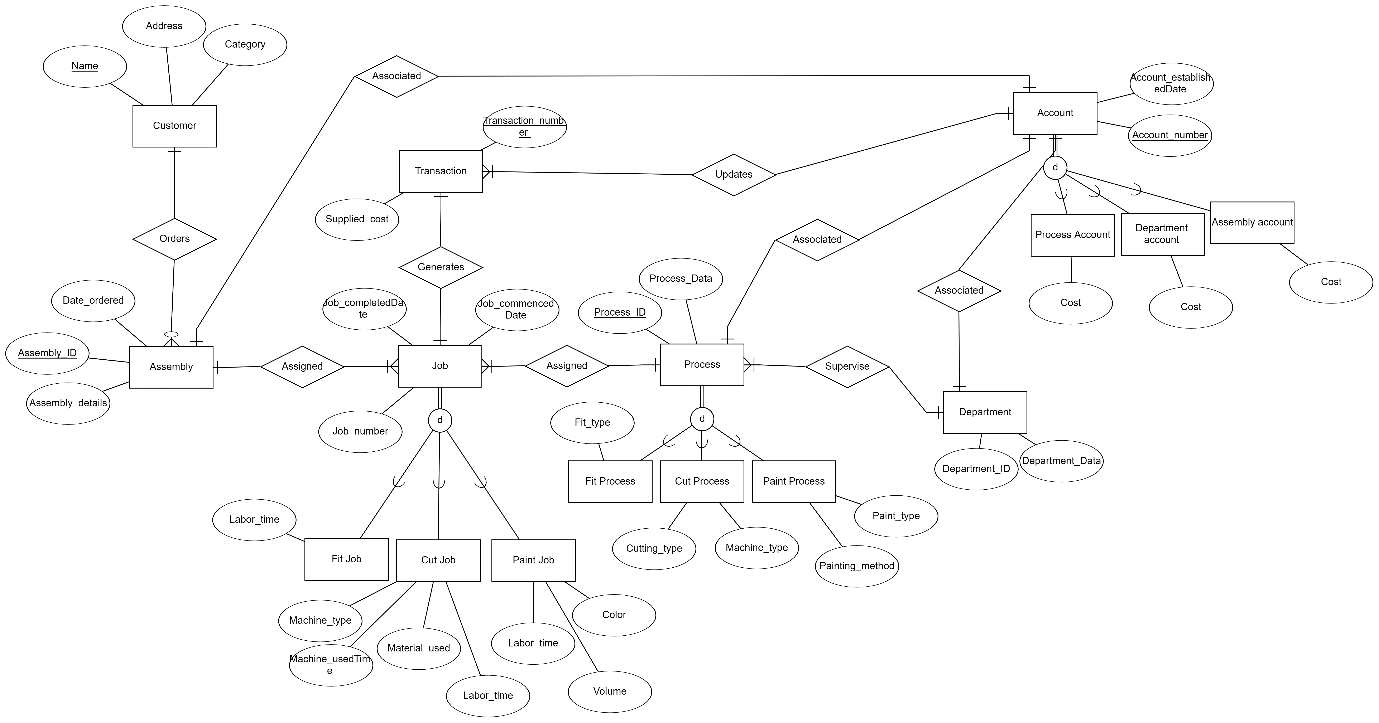
**Task 1:** ER diagram to represent the Job-Shop Accounting database.

**Chen’s Notation ERD**



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

Customer (**Customer\_ID**, Name, Address, Category)

Department (**Department\_ID**, Department\_data)

Process (**Process\_ID**, Process\_data, *Department\_ID\**)

Fit\_Process (*Process\_ID\**, Fit\_type)

Cut\_Process (*Process\_ID\**, Cutting\_type, Machine\_type)

Paint\_Process (*Process\_ID\**, Paint\_type, Painting\_method)

Assembly (**Assembly\_ID**, Date\_Ordered, Assembly\_Details, *Customer\_ID\**)

Job (**Job\_number**, Job\_commencedDate, Job\_completedDate, Labor\_time, *Assembly\_ID\*, Process\_ID\**)

Fit\_Job (*Job\_number\**)

Cut\_Job (*Job\_number\*,* Machine\_type, Machine\_usedTime, Material\_used)

Paint\_Job (*Job\_number\*,* Color, Volume)

Transaction (**Transaction\_number**, Supplied\_cost, *Account\_Number\**)

Account (**Account\_Number***,* Account\_establishedDate, Cost)

Process\_Account (*Account\_number\**, *Process\_ID\*)*

Department\_Account (*Account\_number\*, Department\_ID\*)*

Assembly\_Account (*Account\_number\**, *Assembly\_ID\*)*

**Online queries and their frequencies for the job-shop accounting system:  
1. Enter a new customer (30/day).**

INSERT INTO Customer VALUES (‘1’, ‘name’, ‘address’, ‘category’);

**2. Enter a new department (infrequent).**

INSERT INTO Department VALUES (‘1’, ‘departmentData’)

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

INSERT INTO Process VALUES (‘11’, ‘processData’, ‘1’);

INSERT INTO Fit\_Process VALUES (‘11’, ‘FitType’);

**4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and date ordered and associate it with one or more processes (40/day).**

INSERT INTO Assembly VALUES (‘21’, ‘dateordered’, ‘assemblyDetails’, ‘1’);

INSERT INTO Job VALUES (‘1’, ‘CompletedDate’, ‘CommencedDate’, ‘21’, ‘11’, ‘laborTime’);

INSERT INTO Paint\_Job VALUES (‘1’, ‘Color’, ‘Volume’);

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

INSERT INTO Account VALUES (‘34567’, ‘accountEstablishedDate’, ‘Cost’);

INSERT INTO Assembly\_Account VALUES (‘34567’, ‘11’);

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

INSERT INTO Job VALUES (‘31’, ‘jobCompletedDate’, ‘jobCommencedDate’, ‘21’, ‘11’, ‘laborTime’);

INSERT INTO Cut\_Job VALUES (‘31’, ‘MachineType’, ‘MachineUsedTime’, ‘MaterialUsed’);

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

UPDATE Job Set completed\_date = `jobCompletedDate’ WHERE job\_number = ‘31’

**8. Enter a transaction-no and its sup-cost and update all the costs (details) of the affected accounts by adding sup-cost to their current values of details (50/day).**

DECLARE @Transaction\_number INT = 41;

DECLARE @Supplied\_cost DECIMAL(10, 2) = 250.00;

UPDATE Process\_Account SET Cost = Cost + @Supplied\_cost

WHERE Account\_number = (SELECT Account\_number FROM Transaction WHERE Transaction\_number = @Transaction\_number);

UPDATE Department\_Account SET Cost = Cost + @Supplied\_cost

WHERE Account\_number = (SELECT Account\_number FROM Transaction WHERE Transaction\_number = @Transaction\_number);

UPDATE Assembly\_Account SET Cost = Cost + @Supplied\_cost

WHERE Account\_number = (SELECT Account\_number FROM Transaction WHERE Transaction\_number = @Transaction\_number);

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

DECLARE @Assembly\_ID INT = ‘22’

SELECT a.Assembly\_Id, SUM(c.Cost) FROM Assembly a   
JOIN Assembly\_Account b ON a.Assembly\_Id = b.Assembly\_Id  
JOIN Account c ON b.Account\_Number = c.Account\_Number  
WHERE a.Assembly\_Id = @Assembly\_ID  
GROUP BY a.Assembly\_Id

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

DECLARE @Department\_ID INT = 1;

DECLARE @Job\_completedDate DATE = '20-12-2022’;

SELECT c.Department\_Id, a.Completed\_Date, SUM(a.Labor\_time) AS Total\_Labor\_time   
FROM Job a  
JOIN Process b ON a.Process\_id = b.Process\_id  
JOIN Department c ON b.Department\_Id = c.Department\_Id  
WHERE c.Department\_Id = @Department\_ID AND a.Completed\_Date = @Job\_completedDate  
GROUP BY c.Department\_Id, a.Completed\_Date

**11. Retrieve the processes through which a given assembly-id has passed so far (in date commenced order) and the department responsible for each process (100/day).**

DECLARE @Assembly\_ID INT = 23;

SELECT a.Process\_ID, b.Department\_ID   
FROM Process a  
JOIN Job b ON a.Process\_Id = b.Process\_Id  
JOIN Assembly c ON c.Assembly\_Id = b.Assembly\_Id  
JOIN Department d ON d.Department\_Id = c.Department\_Id  
WHERE c.Assembly\_Id = @Assembly\_ID   
AND b.Completed\_Date IS NOT NULL  
ORDER BY b.Commenced\_Date ASC  
GROUP BY a.Process\_ID, b.Department\_ID;

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

DECLARE @Category1 INT = 9;

DECLARE @Category2 INT = 15;

SELECT Customer\_Id, Name, Address, Category FROM Customer WHERE Category BETWEEN @Category1 AND @Category2 ORDER BY Name;

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

DECLARE @Job\_number1 INT = 33;

DECLARE @Job\_number2 INT = 40;

DELETE FROM Cut\_Job WHERE Job\_number BETWEEN @Job\_number1 AND @Job\_number2;

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

DECLARE @Job\_number INT = 37;

DECLARE @Color VARCHAR(30) = 'Color';

UPDATE Paint\_Job SET Color = @Color WHERE Job\_number = @Job\_number;

**Task 3:**

**3.1. Discuss choices of appropriate storage structures for each relational table assuming that all types of storage structures 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 | Query1 (Insertion) | None | 30 insertions per day | Heap file | Insertions of customer data are performed on a regular basis. |
| Department | Query2 (Insertion) | None | Infrequent | Heap file | Insertions into departments are rare. |
| Process | Query3 (Insertion) | None | Infrequent | Heap file | Process and type production on occasion. |
| Assembly | Query4 (Insertion) | None | 40 insertions per day | Heap file | The set up and association of assemblies occurs frequently. |
| Account & Assembly\_Account | Query5 (Insertion) | None | 10 insertions per day | Heap file | Account creation for multiple entities. |
| Job | Query6 (Insertion) | None | 50 insertions per day | Heap file | Process-based employment generation is common. |
| Job | Query7 (Insertion) | None | 50 insertions per day | Heap file | Update on job completion information. |
| Transaction & Process\_Account, Department\_Account, Assembly\_Account | Query 8 (Update) | None | 50 updates per day | B-tree indexing | Account cost transaction updates. |
| Assembly, Assembly\_Account, Account | Query9 (Random search) | Assembly\_ID | 200 random search per day | B-tree indexing | Retrieve assembly costs efficiently. |
| Job, Process & Department | Query10 (Random search) | Department\_ID, Job\_completedDate | 20 random search per day | B-tree indexing | On a given day, retrieve labour time within a department. |
| Process, Job, Assembly & Department | Query11 (Random search) | CompletedDate | 100 random search per day | B-tree indexing | Retrieve processes and their departments in preparation for an assembly. |
| Customer | Query12 (Range Search) | Category | 100 range searches per day | B-tree indexing | Customers from a specific category range are returned. |
| Cut\_Job | Query13 (Deletion) | Job\_number | 1 deletion per month | Heap file | Cut jobs are deleted on a regular basis. |
| Paint\_Job | Query14 (Update) | Job\_number | 1 update per week | Heap file | Colours for paint jobs are changed on a weekly basis. |

**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).**

The relational tables are stored in SQL database storage structure. The Frequency of these processes and the particular fields that require indexing in order to facilitate effective querying. Because storage and indexing are automatically managed by Azure SQL Database's underlying infrastructure, one can concentrate on database design and queries rather than the intricacies of low-level storage.

**Task 4. 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).**

* 1. **SQL statements:**

CREATE TABLE [account](

[account\_number] [varchar](30) PRIMARY KEY NOT NULL,

[established\_date] [date] NULL,

[cost] [decimal](18, 0) NULL

)

CREATE TABLE [assembly](

[assembly\_id] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[date\_ordered] [date] NULL,

[assembly\_detail] [varchar](255) NULL,

[customer\_id] [int] NULL,

)

CREATE TABLE assembly\_account](

[account\_number] [varchar](30) PRIMARY KEY NOT NULL,

[assembly\_id] [int] NULL

)

CREATE TABLE [customer](

[customer\_id] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[name] [varchar](255) NULL,

[address] [varchar](255) NULL,

[category] [varchar](255) NULL

)

CREATE TABLE [cut\_job](

[job\_number] [int] PRIMARY KEY NOT NULL,

[machine\_type] [varchar](255) NULL,

[machine\_used\_time] [int] NULL,

[material\_used] [varchar](255) NULL

)

CREATE TABLE [cut\_process](

[process\_id] [int] PRIMARY KEY NOT NULL,

[cutting\_type] [varchar](255) NULL,

[machine\_type] [varchar](255) NULL

)

CREATE TABLE [department](

[department\_id] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[department\_data] [varchar](255) NULL

)

CREATE TABLE [department\_account](

[account\_number] [varchar](30) PRIMARY KEY NOT NULL,

[department\_id] [int] NULL

)

CREATE TABLE fit\_job](

[job\_number] [int] PRIMARY KEY NOT NULL

)

CREATE TABLE [fit\_process](

[process\_id] [int] PRIMARY KEY NOT NULL,

[fit\_type] [varchar](255) NULL

)

CREATE TABLE [job](

[job\_number] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[completed\_date] [date] NULL,

[commenced\_date] [date] NULL,

[assembly\_id] [int] NULL,

[process\_id] [int] NULL,

[labor\_time] [int] NULL

)

CREATE TABLE [paint\_job](

[job\_number] [int] PRIMARY KEY NOT NULL,

[color] [varchar](255) NULL,

[volume] [int] NULL

)

CREATE TABLE [paint\_process](

[process\_id] [int] PRIMARY KEY NOT NULL,

[paint\_type] [varchar](255) NULL,

[paint\_method] [varchar](255) NULL

)

CREATE TABLE [process](

[process\_id] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[process\_data] [varchar](255) NULL,

[department\_id] [int] NULL

)

CREATE TABLE [process\_account](

[account\_number] [varchar](30) PRIMARY KEY NOT NULL,

[process\_id] [int] NULL

)

CREATE TABLE [transaction](

[transaction\_number] [int] IDENTITY(1,1) PRIMARY KEY NOT NULL,

[supplied\_cost] [decimal](18, 0) NULL,

[account\_number] [varchar](30) NULL

)

ALTER TABLE [assembly] WITH CHECK ADD CONSTRAINT [fk\_assembly\_1] FOREIGN KEY([customer\_id])

REFERENCES [customer] ([customer\_id])

ALTER TABLE [assembly\_account] WITH CHECK ADD CONSTRAINT [assembly\_account\_1] FOREIGN KEY([account\_number])

REFERENCES [account] ([account\_number])

ALTER TABLE [cut\_job] WITH CHECK ADD CONSTRAINT [fk\_cut\_job\_1] FOREIGN KEY([job\_number])

REFERENCES [job] ([job\_number])

ALTER TABLE [cut\_process] WITH CHECK ADD CONSTRAINT [fk\_cut\_process\_1] FOREIGN KEY([process\_id])

REFERENCES [process] ([process\_id])

ALTER TABLE [department\_account] WITH CHECK ADD CONSTRAINT [fk\_department\_account\_1] FOREIGN KEY([account\_number])

REFERENCES [account] ([account\_number])

ALTER TABLE [department\_account] WITH CHECK ADD CONSTRAINT [fk\_department\_account\_2] FOREIGN KEY([department\_id])

REFERENCES [department] ([department\_id])

ALTER TABLE [fit\_job] WITH CHECK ADD CONSTRAINT [fk\_fit\_job\_1] FOREIGN KEY([job\_number])

REFERENCES [job] ([job\_number])

ALTER TABLE [fit\_process] WITH CHECK ADD CONSTRAINT [fk\_fit\_process\_1] FOREIGN KEY([process\_id])

REFERENCES [process] ([process\_id])

ALTER TABLE [job] WITH CHECK ADD CONSTRAINT [fk\_job\_2] FOREIGN KEY([process\_id])

REFERENCES [process] ([process\_id])

ALTER TABLE [paint\_job] WITH CHECK ADD CONSTRAINT [fk\_paint\_job\_1] FOREIGN KEY([job\_number])

REFERENCES [job] ([job\_number])

ALTER TABLE [paint\_process] WITH CHECK ADD CONSTRAINT [fk\_paint\_process\_1] FOREIGN KEY([process\_id])

REFERENCES [process] ([process\_id])

ALTER TABLE [process] WITH CHECK ADD CONSTRAINT [fk\_department\_1] FOREIGN KEY([department\_id])

REFERENCES [department] ([department\_id])

ALTER TABLE [process\_account] WITH CHECK ADD CONSTRAINT [fk\_process\_account\_1] FOREIGN KEY([account\_number])

REFERENCES [account] ([account\_number])

ALTER TABLE [process\_account] WITH CHECK ADD CONSTRAINT [fk\_process\_account\_2] FOREIGN KEY([process\_id])

REFERENCES [process] ([process\_id])

ALTER TABLE [transaction] WITH CHECK ADD CONSTRAINT [fk\_transaction\_1] FOREIGN KEY([account\_number])

REFERENCES [account] ([account\_number])

CREATE INDEX [idx\_category\_customer\_1] ON [customer]([category])

CREATE INDEX [idx\_completed\_date\_job\_1] ON [job]([completed\_date])

* 1. **Screenshots showing the creation of tables in Azure SQL Database**🡪 **You should do it yourself, because I don’t have an Azure account.**

**Task 5.** 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.

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

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

**Task 6.** Run the program created for Tasks 5 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.

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

14).

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

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

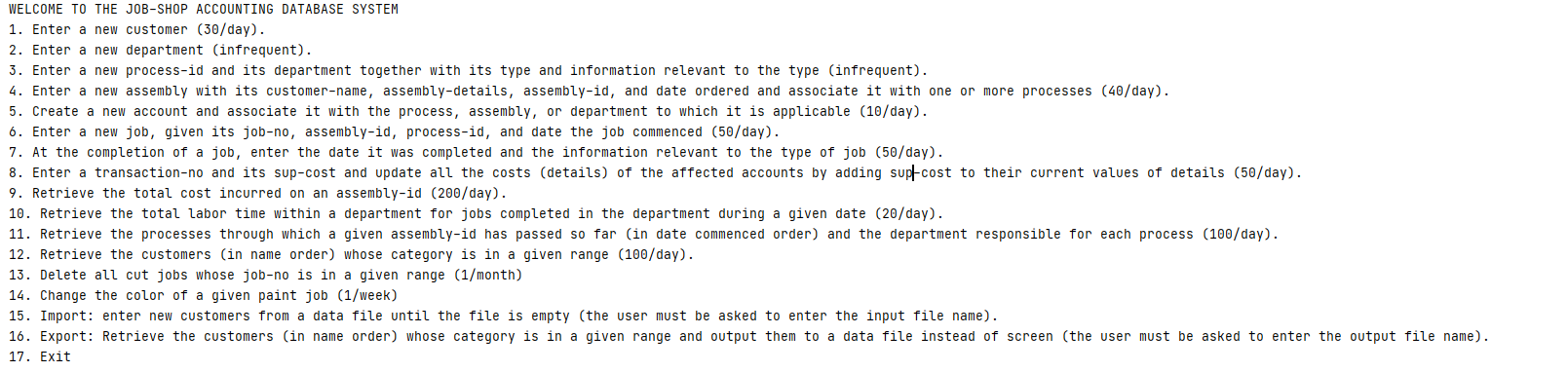
• To demonstrate that Azure SQL Database can detect errors, you also need to perform 3

queries of different types that contain some errors.

**Important Notes for the Java + JDBC + Azure SQL Database Part (Tasks 5 and 6):**

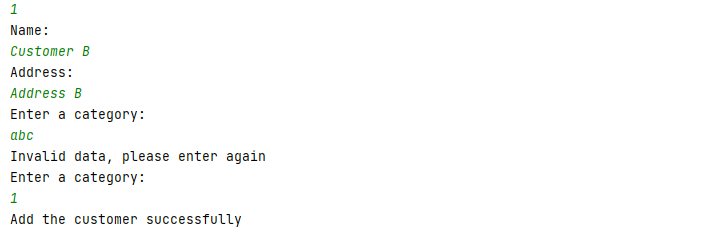
Data manipulation and error checking **must be done** by Azure SQL Database. Your program is only to create the menu, accept choices, form queries, submit them to Azure SQL Database for execution, and display results or error messages.

**Java program Execution**



**6.1. Screenshots showing the testing of query 1**



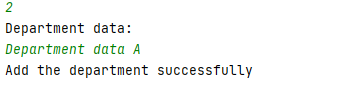


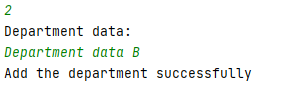


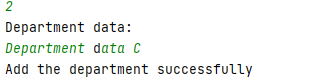


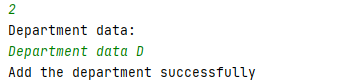


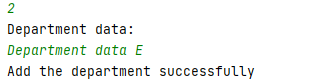
**6.2. Screenshots showing the testing of query 2**



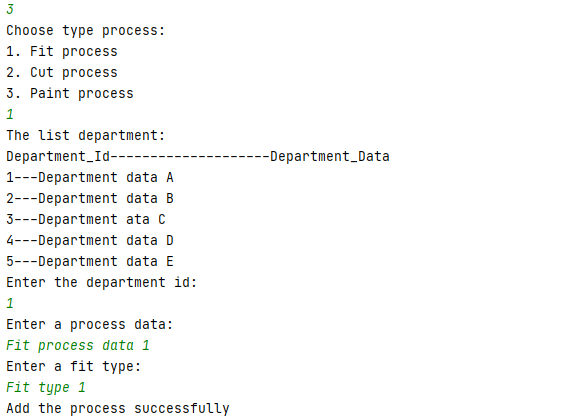


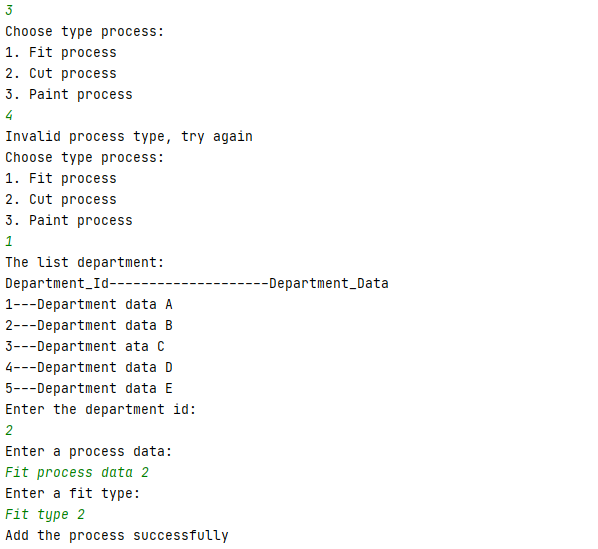


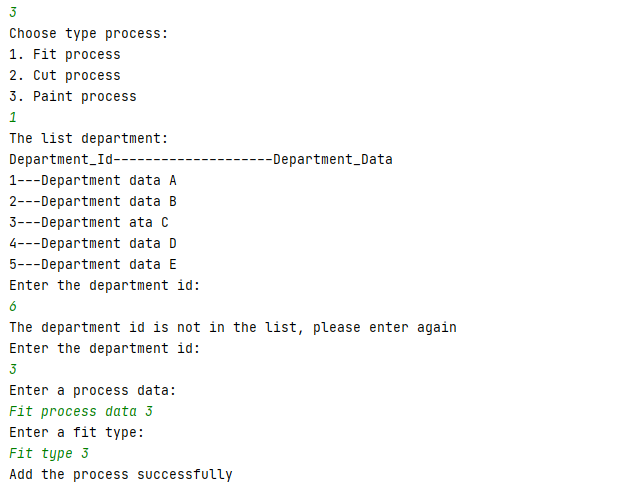


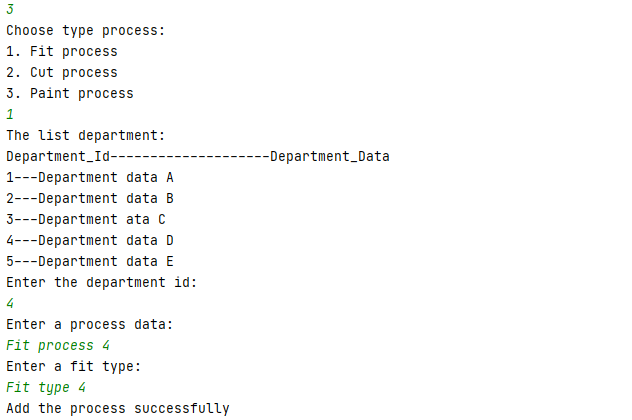


**6.3. Screenshots showing the testing of query 3**Fit process

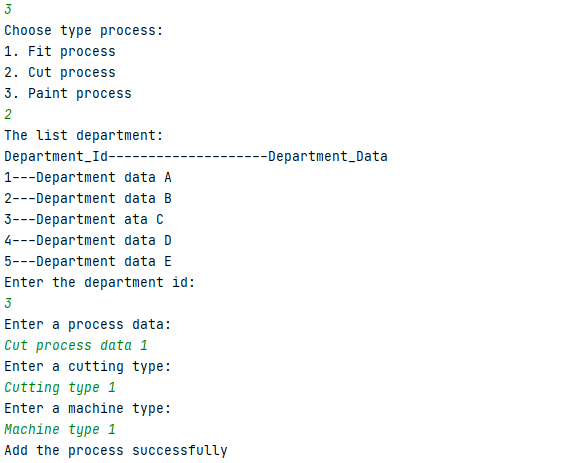


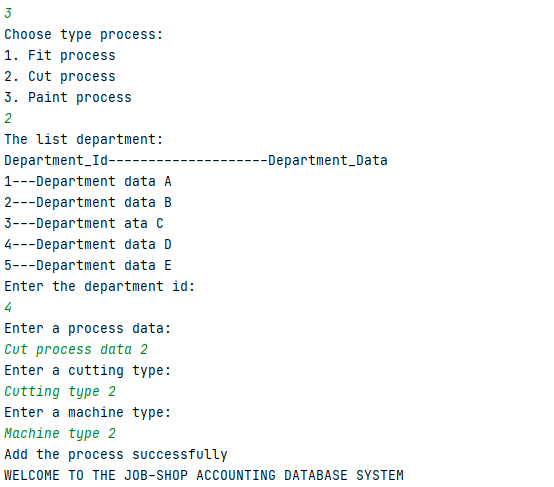


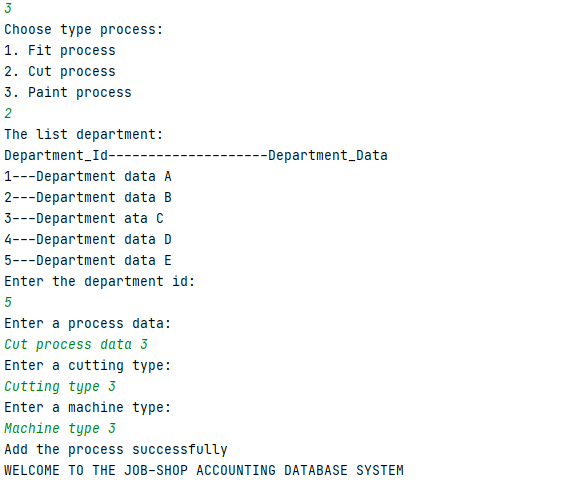




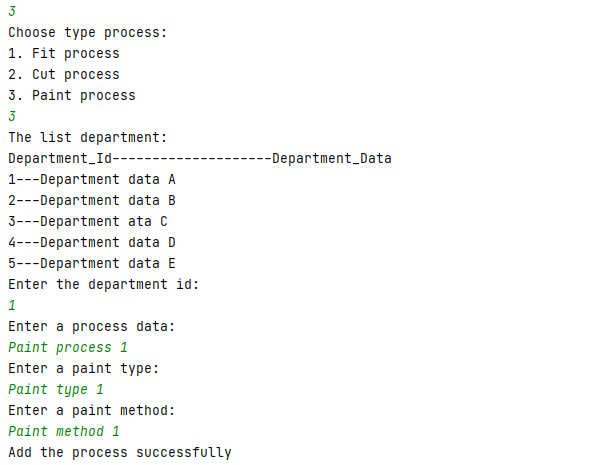
Cut Process

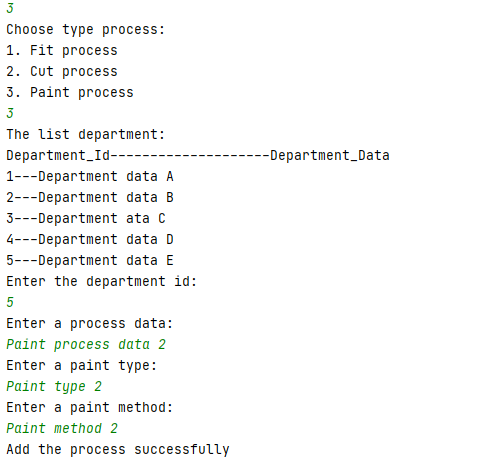


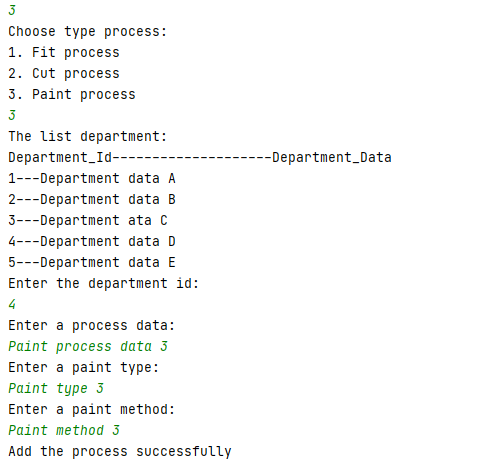




**Paint process**







**Task 7.** 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.