

# Job-Shop Accounting System

**Course Name:** CS/DSA 4513 Database Management Systems

**Section Number:** 001

**Semester:** Fall 2023

**Instructor:** Dr. Le Gruenwald

**Author Name:** Catherine Donner

**OU ID:** 113601169

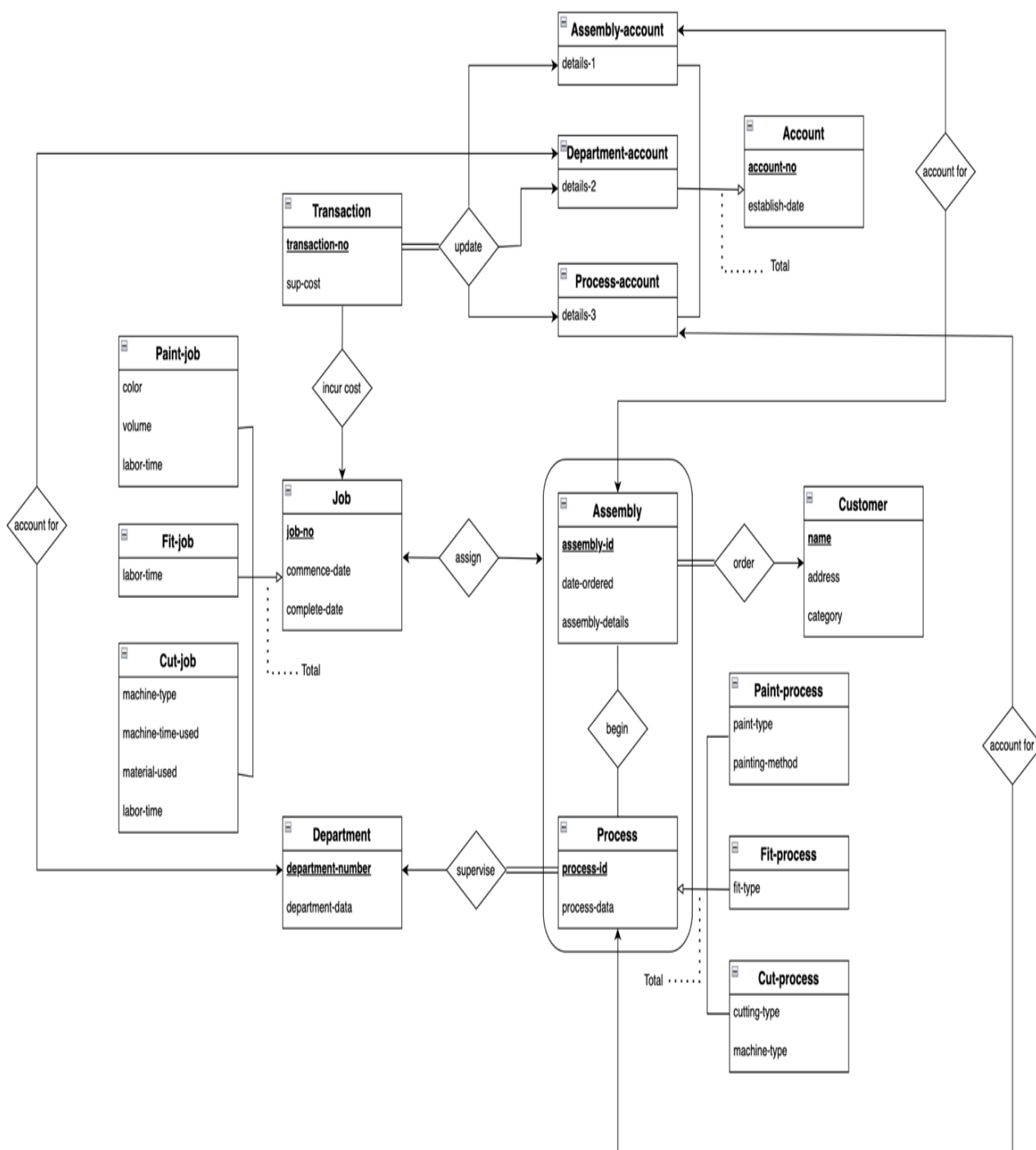
**Email Address:** [Catherine.G.Donner-1@ou.edu](mailto:Catherine.G.Donner-1@ou.edu)

# Table of Contents

<b>Tasks Performed</b>	<b>Page Number</b>
Task 1. E-R Diagram	4
Task 2. Relational Database Schemas	5
Task 3.	
3.1 Discussion of storage structures for tables	6-9
3.2 Discussion of storage structures for tables (Azure SQL Database)	10
Task 4. SQL statements and screenshots showing the creation of tables in	
Azure SQL Database	11-27
Task 5.	
5.1 SQL Statements for implementing all queries (1-14 and error checking)	28-33
5.2 Java source program and screenshots showing successful compilation	33-72
Task 6. Java Program Execution	
6.1 Screenshots showing the testing of Query 1	72-75
6.2 Screenshots showing the testing of Query 2	75-78
6.3 Screenshots showing the testing of Query 3	78-86
6.4 Screenshots showing the testing of Query 4	86-92
6.5 Screenshots showing the testing of Query 5	92-99
6.6 Screenshots showing the testing of Query 6	100-105
6.7 Screenshots showing the testing of Query 7	105-112
6.8 Screenshots showing the testing of Query 8	112-119
6.9 Screenshots showing the testing of Query 9	120-121
6.10 Screenshots showing the testing of Query 10	121-122
6.11 Screenshots showing the testing of Query 11	122-124

6.12 Screenshots showing the testing of Query 12	124-126
6.13 Screenshots showing the testing of Query 13	126-128
6.14 Screenshots showing the testing of Query 14	128-131
6.15 Screenshots showing the testing of import and export options	131-135
6.16 Screenshots showing the testing of three types of errors	135-138
6.17 Screenshots showing the testing of the quit option	138
Task 7. Web Database Application and its Execution	
7.1 Web database application source program and screenshots showing successful compilation	139-150
7.2 Screenshots showing the testing of the Web database application	150-152

### E-R Diagram



## TASK 2

### Relational Database Schemas

Customer(Name, Address, Category)

Assembly(Assembly\_Id, Customer\_Name, Date\_Ordered, Assembly\_Details)

Begin(Assembly\_Id, Process\_Id)

Process(Process\_Id, Department\_Number, Process\_Data)

Paint\_Process(Process\_Id, Paint\_Type, Painting\_Method)

Fit\_Process(Process\_Id, Fit\_Type)

Cut\_Process(Process\_Id, Cutting\_Type, Machine\_Type)

Department(Department Number, Department\_Data)

Job(Job\_No, Commence\_Date, Complete\_Date)

Paint\_Job(Job\_No, Color, Volume, Labor\_Time)

Fit\_Job(Job\_No, Labor\_Time)

Cut\_Job(Job\_No, Machine\_Type, Machine\_Time\_Used, Material\_Used, Labor\_Time)

Assign(Job\_No, Assembly\_Id, Process\_Id)

Transaction(Transaction\_No, Job\_No, Account\_No, Sup\_Cost)

Account(Account\_No, Establish\_Date)

Assembly\_Account(Account\_No, Details\_1)

Department\_Account(Account\_No, Details\_2)

Process\_Account(Account\_No, Details\_3)

Account\_For\_Assembly(Account\_No, Assembly\_Id)

Account\_For\_Process(Account\_No, Process\_Id)

Account\_For\_Department(Account\_No, Department Number)

## TASK 3

### 3.1 Discussion of storage structures for tables

Table Name	Query # and Search Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	1 Insertion	N/A	30/day	B+ Tree Index on Category	Good for efficient range search
	12 Range Search	Category	100/day		
Assembly	4 Insertion	N/A	40/day	Heap	Good for insertion
Begin	4 Insertion	N/A	40/day	Heap	Good for insertion
Process	3 Insertion	N/A	Infrequent	Dynamic Hashing with (secondary index) hash key Department_ Number	Good for frequent random search and insertion
	10 Random Search	Department_ Number	20/day		
	11 Random Search	Process_Id	100/day		
Paint_Process	3 Insertion	N/A	Infrequent	Heap	Good for infrequent insertions

Fit_Process	3 Insertion	N/A	Infrequent	Heap	Good for infrequent insertions
Cut_Process	3 Insertion	N/A	Infrequent	Heap	Good for infrequent insertions
Department	2 Insertion	N/A	Infrequent	Heap	Good for infrequent insertions
Job	6 Insertion	N/A	50/day	Dynamic Hashing with (secondary index) hash key Complete_ Date	Good for frequent random search and insertion
	7 Update	Job_No	50/day		
	10 Random Search	Complete_ Date	20/day		
	11 Random Search	Job_No	100/day		
Paint_Job	7 Insertion	N/A	50/day	Dynamic Hashing with (primary index) hash key Job_No	Good for frequent insertion and random search
	10 Random Search	Job_No	20/day		
	14 Update	Job_No	1/week		

Fit_Job	7 Insertion	N/A	50/day	Dynamic Hashing with (primary index) hash key Job_No	Good for insertion and random search
	10 Random Search	Job_No	20/day		
Cut_Job	7 Insertion	N/A	50/day	B+ Tree Index on Job_No	Good for range search
	10 Random Search	Job_No	20/day		
	13 Range Search	Job_No	1/month		
Assign	6 Insertion	N/A	50/day	Dynamic Hashing with (secondary index) hash key on Assembly_Id	Good for frequent random search and insertion
	11 Random Search	Assembly_Id	100/day		
Transaction	8 Insertion	N/A	50/day	Heap	Good for insertion
Account	5 Insertion	N/A	10/day	Heap	Good for insertions



Assembly_Account	5 Insertion	N/A	10/day	Dynamic Hashing with (primary index) hash key on Account_No	Good for frequent updates
	8 Update	Account_No	50/day		
Department_Account	5 Insertion	N/A	10/day	Dynamic Hashing with (primary index) hash key on Account_No	Good for frequent updates
	8 Update	Account_No	50/day		
Process_Account	5 Insertion	N/A	10/day	Dynamic Hashing with (primary index) hash key on Account_No	Good for frequent updates
	8 Update	Account_No	50/day		
Account_For_Assembly	5 Insertion	N/A	10/day	Dynamic Hashing with (primary index) hash key on Assembly_Id	Good for frequent random search
	9 Random Search	Assembly_Id	200/day		
Account_For_Process	5 Insertion	N/A	10/day	Heap	Good for insertion
Account_For_Department	5 Insertion	N/A	10/day	Heap	Good for insertion

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

For my storage structures, I chose heap structures for tables that involved insertions only, B+ trees with specified indexes for tables that involved range search, and dynamic hashing with specified hash keys for tables that involved random search or updates. Azure SQL uses B-trees for general index implementation, and B+ trees can be used for rowstore indexes. However, B+ tree indexes cannot apply to columnstore indexes. Since my B+ tree indexes would be on columns in my tables, Azure SQL does not support B+ tree structures. For dynamic hashing, there are ways to create hash indexes in Azure SQL, however these require memory-optimized tables, and memory-optimized tables are not supported by the class version of Azure SQL. Heaps are supported by Azure SQL, however they are strictly supported for insertions into tables only. Any querying involving secondary indexes, range search, etc. are not supported for heaps in Azure SQL. Primary indexes are automatically created in Azure SQL on primary keys in tables, and to optimize querying with secondary indexes, I created these indexes to resolve these issues with implementation of the file organization structures.

Supporting documentation:

Columnstore indexes: Overview ([docs.microsoft.com](https://docs.microsoft.com))

Indexes on Memory-Optimized Tables ([docs.microsoft.com](https://docs.microsoft.com))

Heaps (Tables without Clustered Indexes) ([docs.microsoft.com](https://docs.microsoft.com))

## TASK 4

### SQL Statements and Screenshots Showing the Creation of Tables in Azure SQL Database

#### Create Table Statements

```
CREATE TABLE Customer (
    Name varchar(25) PRIMARY KEY,
    Address varchar(100),
    Category int,
    CONSTRAINT Category_check CHECK (Category BETWEEN 0 AND 10)
);
```

The screenshot displays the Azure SQL Database interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Class, dbo.Customer, dbo.Director, dbo.Enrollment, dbo.Movie, dbo.movie\_night, dbo.Performer, dbo.Student, and Dropped Ledger Tables. The main pane shows the execution of a SQL query. The query is as follows:

```
1 CREATE TABLE Customer (
2     Name varchar(25) PRIMARY KEY,
3     Address varchar(100),
4     Category int,
5     CONSTRAINT Category_check CHECK (Category BETWEEN 0 AND 10)
6 );
```

Below the query, the 'Messages' pane shows the execution results:

```
9:47:28 PM    Started executing query at Line 1
              Commands completed successfully.
              Total execution time: 00:00:00.074
```

```
CREATE TABLE Assembly (
    Assembly_Id int PRIMARY KEY,
    Date_Ordered date,
    Assembly_Details varchar(100),
    Customer_Name varchar(25) FOREIGN KEY REFERENCES Customer(Name)
```

);

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Class, dbo.Customer, dbo.Director, dbo.Enrollment, dbo.Movie, dbo.movie\_night, dbo.Performer, dbo.Student, and Dropped Ledger Tables. The 'dbo.Assembly' table is selected. On the right, the SQL query editor shows the following code:

```
1 CREATE TABLE Assembly (
2     Assembly_Id int PRIMARY KEY,
3     Date_Ordered date,
4     Assembly_Details varchar(100),
5     Customer_Name varchar(25) FOREIGN KEY REFERENCES Customer(Name)
6 );
```

Below the query editor, the 'Messages' pane shows the following message:

9:48:45 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.089

```
CREATE TABLE Department (
    Department_Number int PRIMARY KEY,
    Department_Data varchar(50)
);
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Class, dbo.Customer, dbo.Department, dbo.Director, dbo.Enrollment, dbo.Movie, dbo.movie\_night, dbo.Performer, dbo.Student, and Dropped Ledger Tables. The 'dbo.Department' table is selected. On the right, the SQL query editor shows the following code:

```
1 CREATE TABLE Department (
2     Department_Number int PRIMARY KEY,
3     Department_Data varchar(50)
4 );
```

Below the query editor, the 'Messages' pane shows the following message:

9:49:17 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.067

```

CREATE TABLE Process (

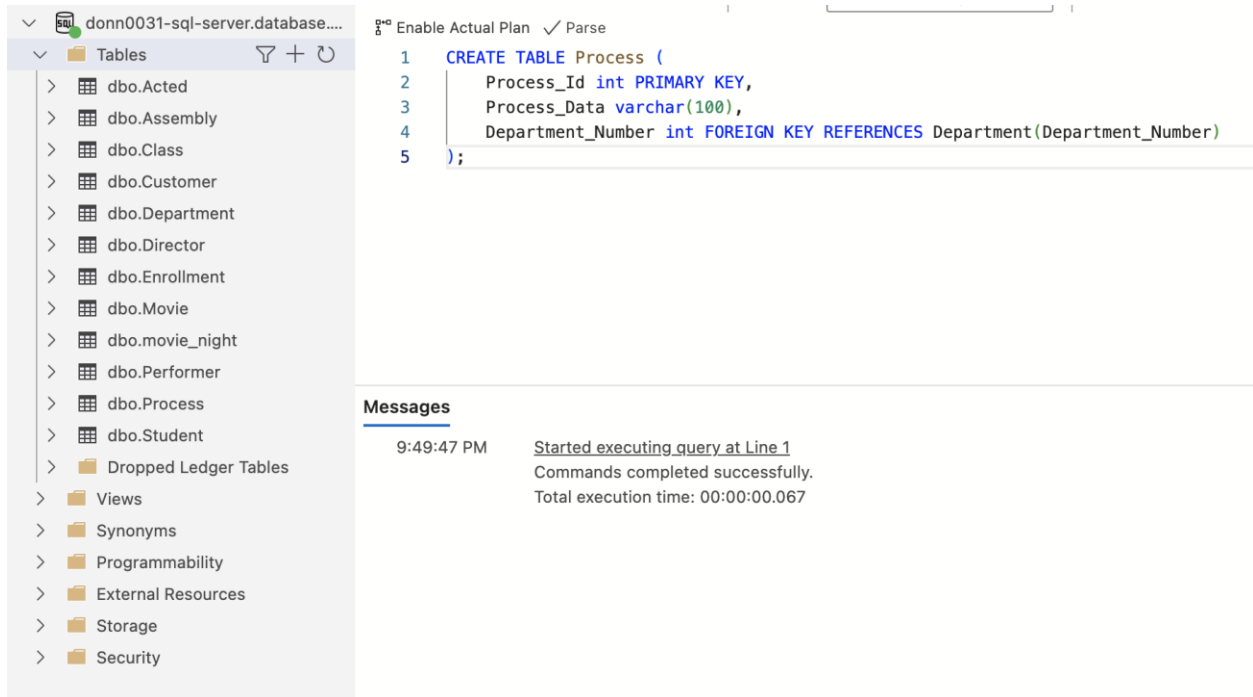
    Process_Id int PRIMARY KEY,

    Process_Data varchar(100),

    Department_Number int FOREIGN KEY REFERENCES Department(Department_Number)

);

```



```

CREATE TABLE Paint_Process (

    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    Paint_Type varchar(25),

    Painting_Method varchar(25),

    PRIMARY KEY(Process_Id)

);

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Class, dbo.Customer, dbo.Department, dbo.Director, dbo.Enrollment, dbo.Movie, dbo.movie\_night, dbo.Paint\_Process, dbo.Performer, dbo.Process, dbo.Student, and Dropped Ledger Tables. The main pane displays the SQL script for creating the Paint\_Process table:

```

1 CREATE TABLE Paint_Process (
2     Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
3     Paint_Type varchar(25),
4     Painting_Method varchar(25),
5     PRIMARY KEY(Process_Id)
6 );

```

Below the script, the 'Messages' pane shows the execution results:

```

9:50:15 PM    Started executing query at Line 1
              Commands completed successfully.
              Total execution time: 00:00:00.068

```

```

CREATE TABLE Fit_Process (

    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    Fit_Type varchar(25),

    PRIMARY KEY(Process_Id)

);

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Class, dbo.Customer, dbo.Department, dbo.Director, dbo.Enrollment, dbo.Fit\_Process, dbo.Movie, dbo.movie\_night, dbo.Paint\_Process, dbo.Performer, dbo.Process, dbo.Student, and Dropped Ledger Tables. The main pane displays the SQL script for creating the Fit\_Process table:

```

1 CREATE TABLE Fit_Process (
2     Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
3     Fit_Type varchar(25),
4     PRIMARY KEY(Process_Id)
5 );

```

Below the script, the 'Messages' pane shows the execution results:

```

9:50:49 PM    Started executing query at Line 1
              Commands completed successfully.
              Total execution time: 00:00:00.066

```

```
CREATE TABLE Cut_Process (

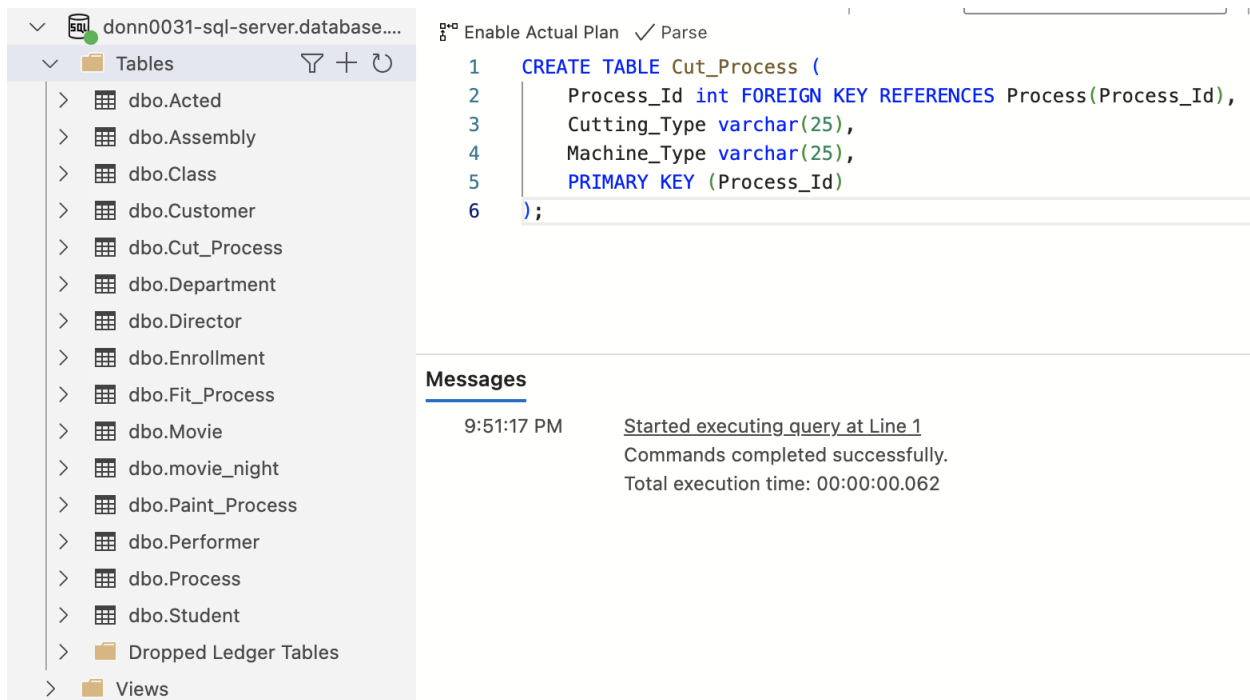
    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    Cutting_Type varchar(25),

    Machine_Type varchar(25),

    PRIMARY KEY(Process_Id)

);
```



```
CREATE TABLE Begin_Manufacture (

    Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),

    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    PRIMARY KEY (Assembly_Id, Process_Id)

); -- this was changed from Begin to Begin_Manufacture to not confuse SQL BEGIN syntax
```

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including `dbo.Acted`, `dbo.Assembly`, `dbo.Begin_Manufacture`, `dbo.Class`, `dbo.Customer`, `dbo.Cut_Process`, `dbo.Department`, `dbo.Director`, `dbo.Enrollment`, `dbo.Fit_Process`, `dbo.Movie`, `dbo.movie_night`, `dbo.Paint_Process`, `dbo.Performer`, `dbo.Process`, `dbo.Student`, `Dropped Ledger Tables`, and `Views`. The main pane shows a SQL query to create a table named `Begin_Manufacture`. The query is as follows:

```

1 CREATE TABLE Begin_Manufacture (
2     Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),
3     Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
4     PRIMARY KEY(Assembly_Id, Process_Id)
5 );

```

Below the query, the 'Messages' pane shows the execution results:

```

9:52:25 PM    Started executing query at Line 1
              Commands completed successfully.
              Total execution time: 00:00:00.082

```

```

CREATE TABLE Job (

    Job_No int PRIMARY KEY,

    Commence_Date date,

    Complete_Date date

);

```



The screenshot shows the SQL Server Enterprise Manager interface. On the left, a tree view displays the database structure for 'donn0031-sql-server.database....'. Under the 'Tables' folder, a list of tables is shown, including 'dbo.Acted', 'dbo.Assembly', 'dbo.Begin\_Manufacture', 'dbo.Class', 'dbo.Customer', 'dbo.Cut\_Process', 'dbo.Department', 'dbo.Director', 'dbo.Enrollment', 'dbo.Fit\_Process', 'dbo.Job', 'dbo.Movie', 'dbo.movie\_night', 'dbo.Paint\_Process', 'dbo.Performer', 'dbo.Process', 'dbo.Student', 'Dropped Ledger Tables', and 'Views'. On the right, a query window is open, showing a SQL command to create a table named 'Job'. The command is as follows:

```

1 CREATE TABLE Job (
2     Job_No int PRIMARY KEY,
3     Commence_Date date,
4     Complete_Date date
5 );

```

Below the query window, the 'Messages' pane displays the execution results:

```

9:52:58 PM    Started executing query at Line 1
              Commands completed successfully.
              Total execution time: 00:00:00.067

```

```

CREATE TABLE Paint_Job (

    Job_No int FOREIGN KEY REFERENCES Job(Job_No),

    Color varchar(25),

    Volume int,

    Labor_Time int,

    PRIMARY KEY(Job_No)

);

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Begin\_Manufacture, dbo.Class, dbo.Customer, dbo.Cut\_Process, dbo.Department, dbo.Director, dbo.Enrollment, dbo.Fit\_Process, dbo.Job, dbo.Movie, dbo.movie\_night, dbo.Paint\_Job, dbo.Paint\_Process, dbo.Performer, dbo.Process, dbo.Student, and Dropped Ledger Tables. The main pane displays the SQL script for creating the Paint\_Job table:

```
1 CREATE TABLE Paint_Job (
2     Job_No int FOREIGN KEY REFERENCES Job(Job_No),
3     Color varchar(25),
4     Volume int,
5     Labor_Time int,
6     PRIMARY KEY (Job_No)
7 );
```

Below the script, the 'Messages' pane shows the execution results:

```
9:53:29 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.062
```

```
CREATE TABLE Fit_Job (
    Job_No int FOREIGN KEY REFERENCES Job(Job_No),
    Labor_Time int,
    PRIMARY KEY(Job_No)
);
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including dbo.Acted, dbo.Assembly, dbo.Begin\_Manufacture, dbo.Class, dbo.Customer, dbo.Cut\_Process, dbo.Department, dbo.Director, dbo.Enrollment, dbo.Fit\_Job, dbo.Fit\_Process, dbo.Job, dbo.Movie, dbo.movie\_night, dbo.Paint\_Job, dbo.Paint\_Process, dbo.Performer, dbo.Process, dbo.Student, and Dropped Ledger Tables. The main pane displays the SQL script for creating the Fit\_Job table:

```
1 CREATE TABLE Fit_Job (
2     Job_No int FOREIGN KEY REFERENCES Job(Job_No),
3     Labor_Time int,
4     PRIMARY KEY(Job_No)
5 );
```

Below the script, the 'Messages' pane shows the execution results:

```
9:54:05 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.061
```

```
CREATE TABLE Cut_Job (
```

```

Job_No int FOREIGN KEY REFERENCES Job(Job_No),

Machine_Type varchar(25),

Machine_Time_Used int,

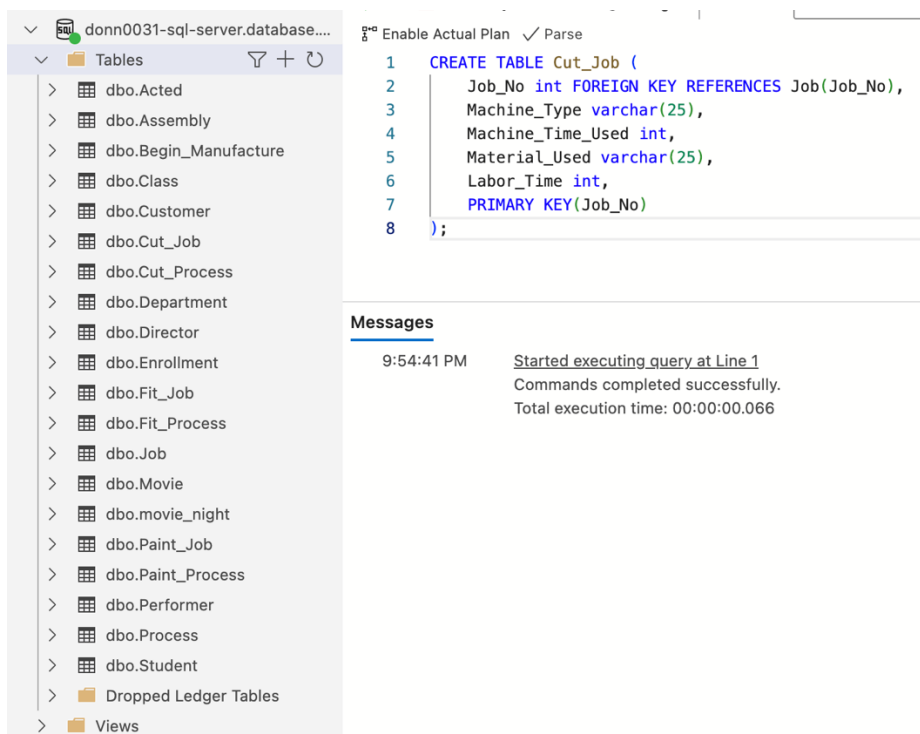
Material_Used varchar(25),

Labor_Time int,

PRIMARY KEY(Job_No)

);

```



```

CREATE TABLE Assign (

    Job_No int FOREIGN KEY REFERENCES Job(Job_No),

    Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),

    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    PRIMARY KEY(Job_No)

);

```

donn0031-sql-server.database...

Enable Actual Plan Parse

```

1 CREATE TABLE Assign (
2     Job_No int FOREIGN KEY REFERENCES Job(Job_No),
3     Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),
4     Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
5     PRIMARY KEY(Job_No)
6 );

```

**Messages**

9:55:09 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.062

```

CREATE TABLE Account (

    Account_No int PRIMARY KEY,

    Establish_Date date

);

```

donn0031-sql-server.database...

1 CREATE TABLE Account (
2 Account\_No int PRIMARY KEY,
3 Establish\_Date date
4 );

**Messages**

9:55:46 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.061

```

CREATE TABLE Assembly_Account (

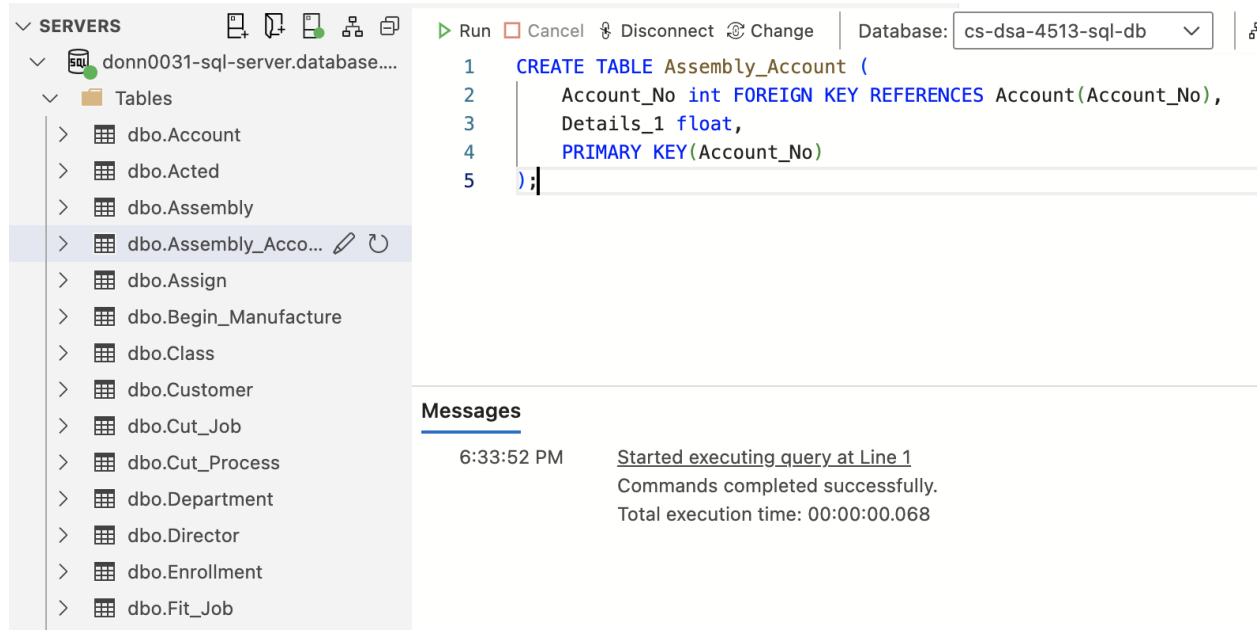
    Account_No int FOREIGN KEY REFERENCES Account(Account_No),

    Details_1 float,

    PRIMARY KEY(Account_No)

);

```



```

CREATE TABLE Department_Account (

    Account_No int FOREIGN KEY REFERENCES Account(Account_No),

    Details_2 float,

    PRIMARY KEY(Account_No)

);

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including 'dbo.Account', 'dbo.Acted', 'dbo.Assembly', 'dbo.Assembly\_Acco...', 'dbo.Assign', 'dbo.Begin\_Manufacture', 'dbo.Class', 'dbo.Customer', 'dbo.Cut\_Job', 'dbo.Cut\_Process', 'dbo.Department', 'dbo.Department\_Account', and 'dbo.Director'. The 'dbo.Assembly\_Acco...' table is selected. On the right, the SQL query editor shows the following code:

```

1 CREATE TABLE Department_Account (
2     Account_No int FOREIGN KEY REFERENCES Account(Account_No),
3     Details_2 float,
4     PRIMARY KEY(Account_No)
5 );

```

Below the query editor, the 'Messages' pane shows the following message:

6:34:30 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.075

```

CREATE TABLE Process_Account (

    Account_No int FOREIGN KEY REFERENCES Account(Account_No),

    Details_3 float,

    PRIMARY KEY(Account_No)

);

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables including 'dbo.Account', 'dbo.Acted', 'dbo.Assembly', 'dbo.Assembly\_Account', 'dbo.Assign', 'dbo.Begin\_Manufacture', 'dbo.Class', 'dbo.Customer', 'dbo.Cut\_Job', 'dbo.Cut\_Process', 'dbo.Department', 'dbo.Department\_Account', 'dbo.Director', 'dbo.Enrollment', 'dbo.Fit\_Job', 'dbo.Fit\_Process', 'dbo.Job', 'dbo.Movie', 'dbo.movie\_night', 'dbo.Paint\_Job', 'dbo.Paint\_Process', 'dbo.Performer', 'dbo.Process', and 'dbo.Process\_Account'. The 'dbo.Process\_Account' table is selected. On the right, the SQL query editor shows the following code:

```

1 CREATE TABLE Process_Account (
2     Account_No int FOREIGN KEY REFERENCES Account(Account_No),
3     Details_3 float,
4     PRIMARY KEY(Account_No)
5 );

```

Below the query editor, the 'Messages' pane shows the following message:

6:34:57 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.069

```

CREATE TABLE Cost_Transaction (

    Transaction_No int PRIMARY KEY,

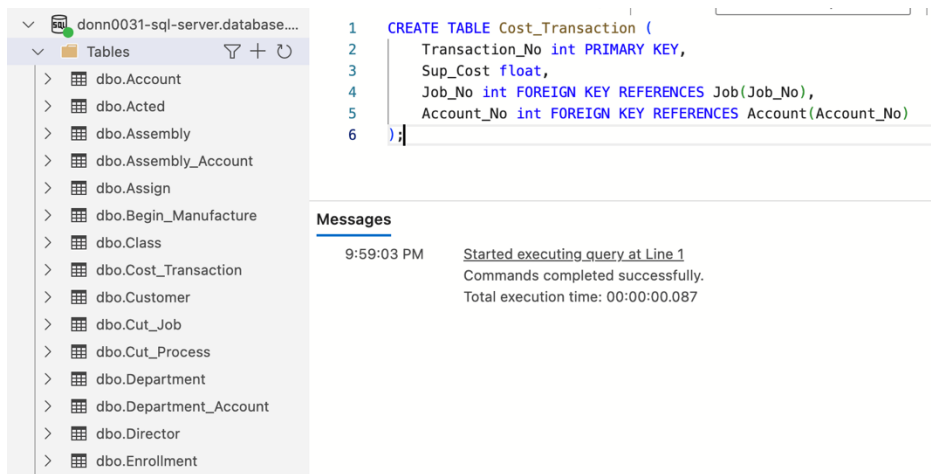
    Sup_Cost float,

    Job_No int FOREIGN KEY REFERENCES Job(Job_No),

    Account_No int FOREIGN KEY REFERENCES Account(Account_No)

); -- changed from Transaction to Cost_Transaction to not confuse SQL TRANSACTION syntax

```



```

CREATE TABLE Account_For_Assembly (

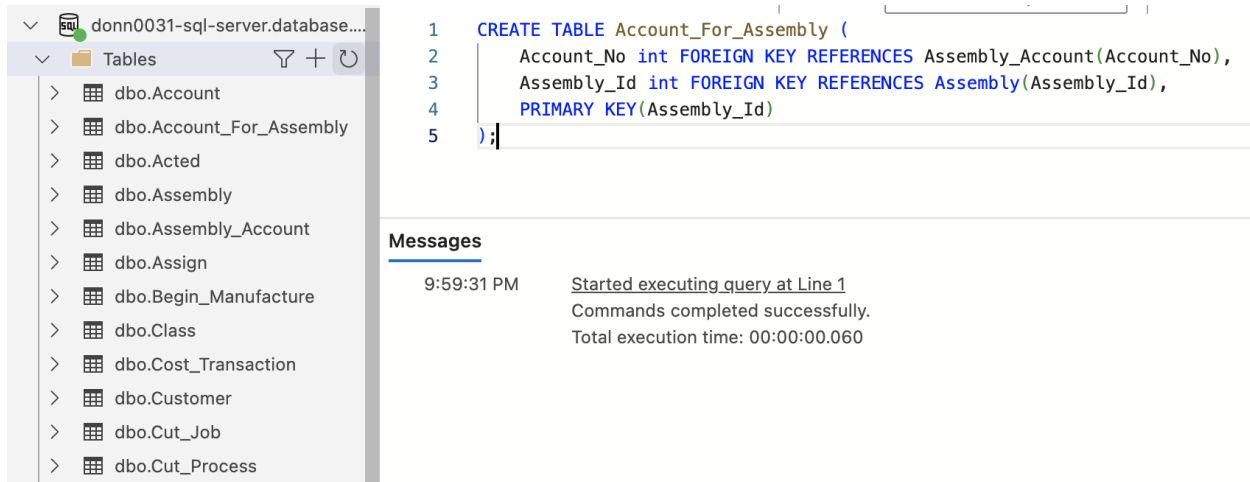
    Account_No int FOREIGN KEY REFERENCES Assembly_Account(Account_No),

    Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),

    PRIMARY KEY(Assembly_Id)

);

```



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables in the 'dbo' schema, including 'Account', 'Account\_For\_Assembly', 'Acted', 'Assembly', 'Assembly\_Account', 'Assign', 'Begin\_Manufacture', 'Class', 'Cost\_Transaction', 'Customer', 'Cut\_Job', and 'Cut\_Process'. The 'Account\_For\_Assembly' table is highlighted. On the right, the SQL query editor shows the following code:

```
1 CREATE TABLE Account_For_Assembly (
2     Account_No int FOREIGN KEY REFERENCES Assembly_Account(Account_No),
3     Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),
4     PRIMARY KEY(Assembly_Id)
5 );
```

Below the query editor, the 'Messages' pane shows the following information:

9:59:31 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.060

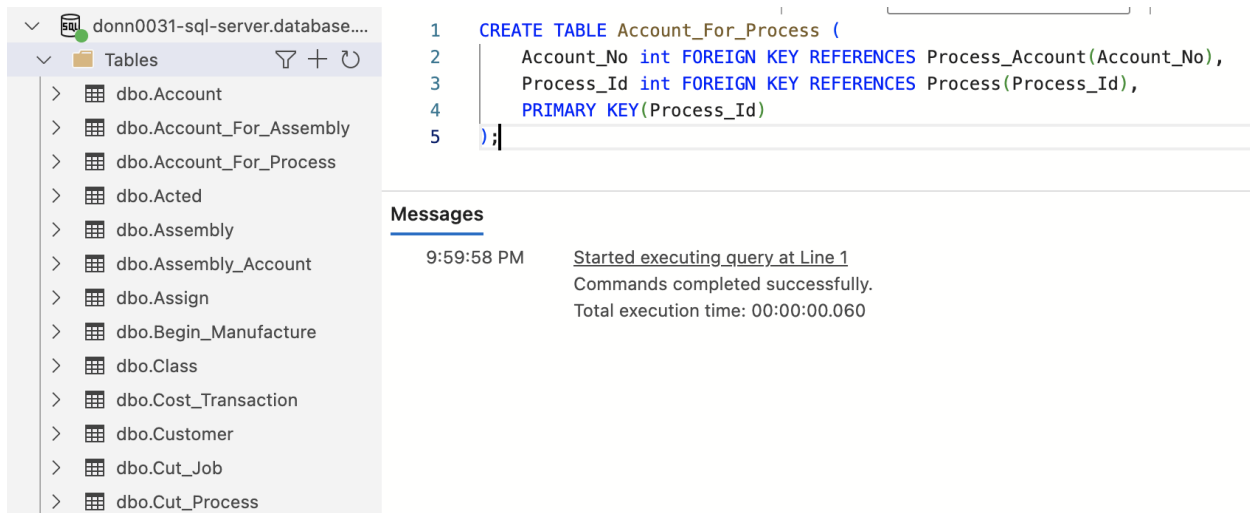
```
CREATE TABLE Account_For_Process (

    Account_No int FOREIGN KEY REFERENCES Process_Account(Account_No),

    Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

    PRIMARY KEY(Process_Id)

);
```



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded, showing a list of tables in the 'dbo' schema, including 'Account', 'Account\_For\_Assembly', 'Account\_For\_Process', 'Acted', 'Assembly', 'Assembly\_Account', 'Assign', 'Begin\_Manufacture', 'Class', 'Cost\_Transaction', 'Customer', 'Cut\_Job', and 'Cut\_Process'. The 'Account\_For\_Process' table is highlighted. On the right, the SQL query editor shows the following code:

```
1 CREATE TABLE Account_For_Process (
2     Account_No int FOREIGN KEY REFERENCES Process_Account(Account_No),
3     Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
4     PRIMARY KEY(Process_Id)
5 );
```

Below the query editor, the 'Messages' pane shows the following information:

9:59:58 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.060

```
CREATE TABLE Account_For_Department (

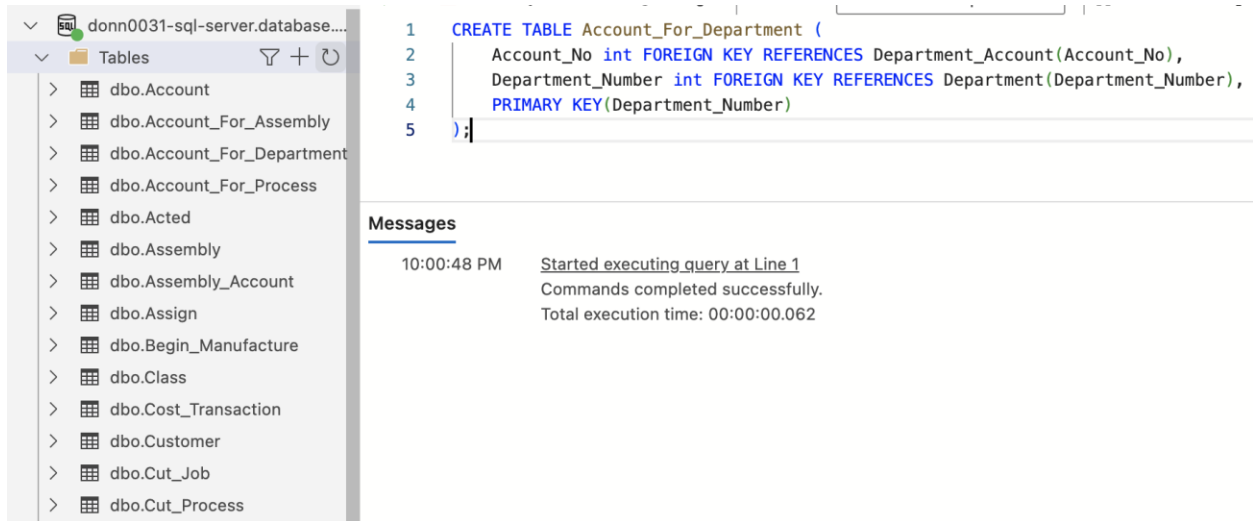
    Account_No int FOREIGN KEY REFERENCES Department_Account(Account_No),

    Department_Number FOREIGN KEY REFERENCES Department(Department_Number),

    PRIMARY KEY(Department_Number)

);
```





The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Tables' folder is expanded under the 'donn0031-sql-server.database...' server. The table list includes: dbo.Account, dbo.Account\_For\_Assembly, dbo.Account\_For\_Department, dbo.Account\_For\_Process, dbo.Acted, dbo.Assembly, dbo.Assembly\_Account, dbo.Assign, dbo.Begin\_Manufacture, dbo.Class, dbo.Cost\_Transaction, dbo.Customer, dbo.Cut\_Job, and dbo.Cut\_Process. The 'Messages' pane on the right displays the execution of a SQL command:

```

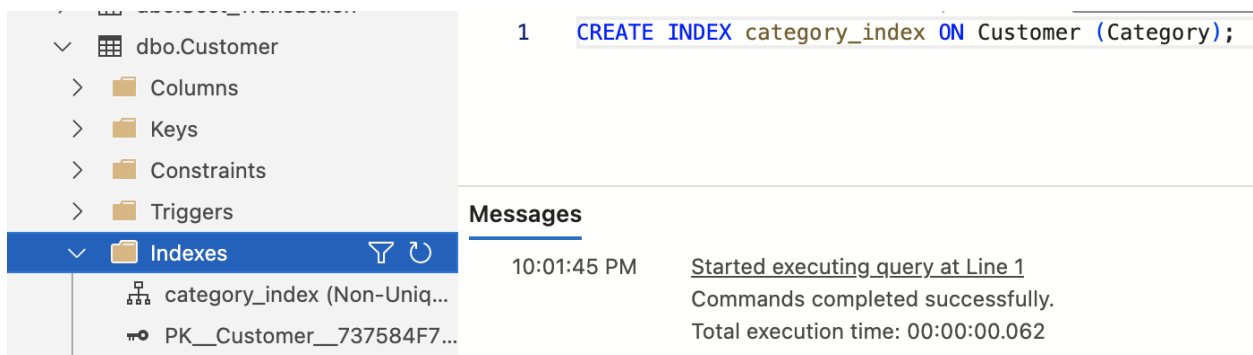
1 CREATE TABLE Account_For_Department (
2     Account_No int FOREIGN KEY REFERENCES Department_Account(Account_No),
3     Department_Number int FOREIGN KEY REFERENCES Department(Department_Number),
4     PRIMARY KEY(Department_Number)
5 );

```

The message log shows the command was executed successfully at 10:00:48 PM with a total execution time of 00:00:00.062.

## Create Index Statements

CREATE INDEX category\_index ON Customer (Category);



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Indexes' folder is expanded under the 'dbo.Customer' table. The index list includes: category\_index (Non-Uniq...) and PK\_\_Customer\_\_737584F7... The 'Messages' pane on the right displays the execution of a SQL command:

```

1 CREATE INDEX category_index ON Customer (Category);

```

The message log shows the command was executed successfully at 10:01:45 PM with a total execution time of 00:00:00.062.

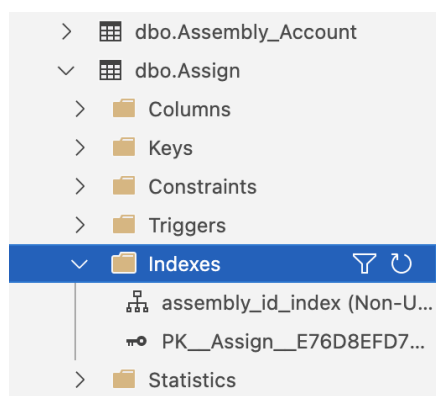
CREATE INDEX dept\_no\_index ON Process (Department\_Number);

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Indexes' folder for the 'Process' table is expanded, showing the newly created 'dept\_no\_index (Non-Unique)'. The main pane displays the SQL command: `1 CREATE INDEX dept_no_index ON Process (Department_Number);`. Below the command, the 'Messages' pane shows the execution results: 'Started executing query at Line 1', 'Commands completed successfully.', and 'Total execution time: 00:00:00.062'.

CREATE INDEX complete\_date\_index ON Job (Complete\_Date);

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Indexes' folder for the 'Job' table is expanded, showing the newly created 'complete\_date\_index (Non-Unique)'. The main pane displays the SQL command: `1 CREATE INDEX complete_date_index ON Job (Complete_Date);`. Below the command, the 'Messages' pane shows the execution results: 'Started executing query at Line 1', 'Commands completed successfully.', and 'Total execution time: 00:00:00.061'.

CREATE INDEX assembly\_id\_index ON Assign (Assembly\_Id);



```
1 CREATE INDEX assembly_id_index ON Assign (Assembly_Id);
```

#### Messages

10:04:34 PM Started executing query at Line 1  
Commands completed successfully.  
Total execution time: 00:00:00.067