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

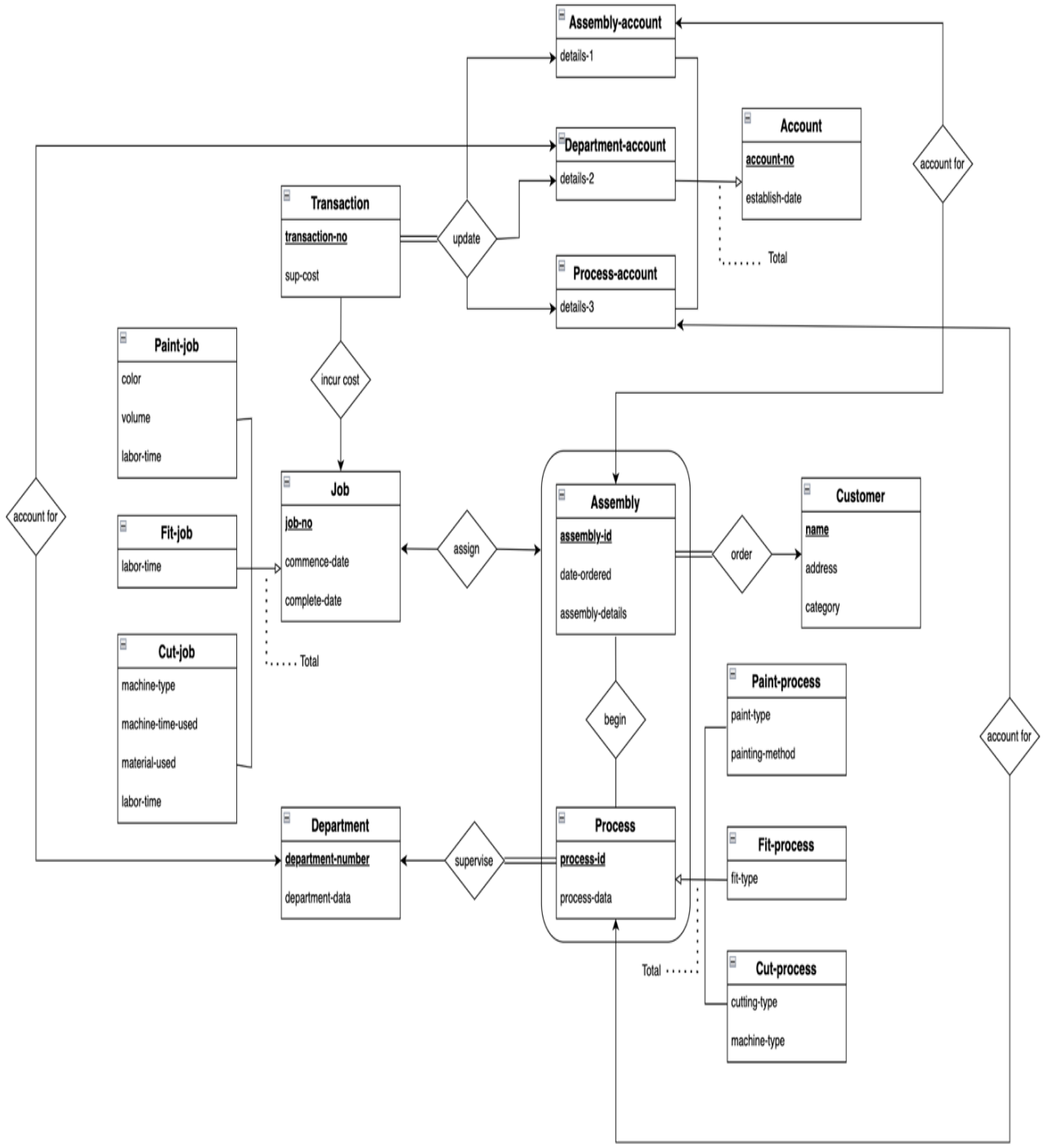
OU ID: 113601169

Email Address: Catherine.G.Donner-1@ou.edu

Table of Contents

TASK 1

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)

Indexes on Memory-Optimized Tables (docs.microsoft.com)

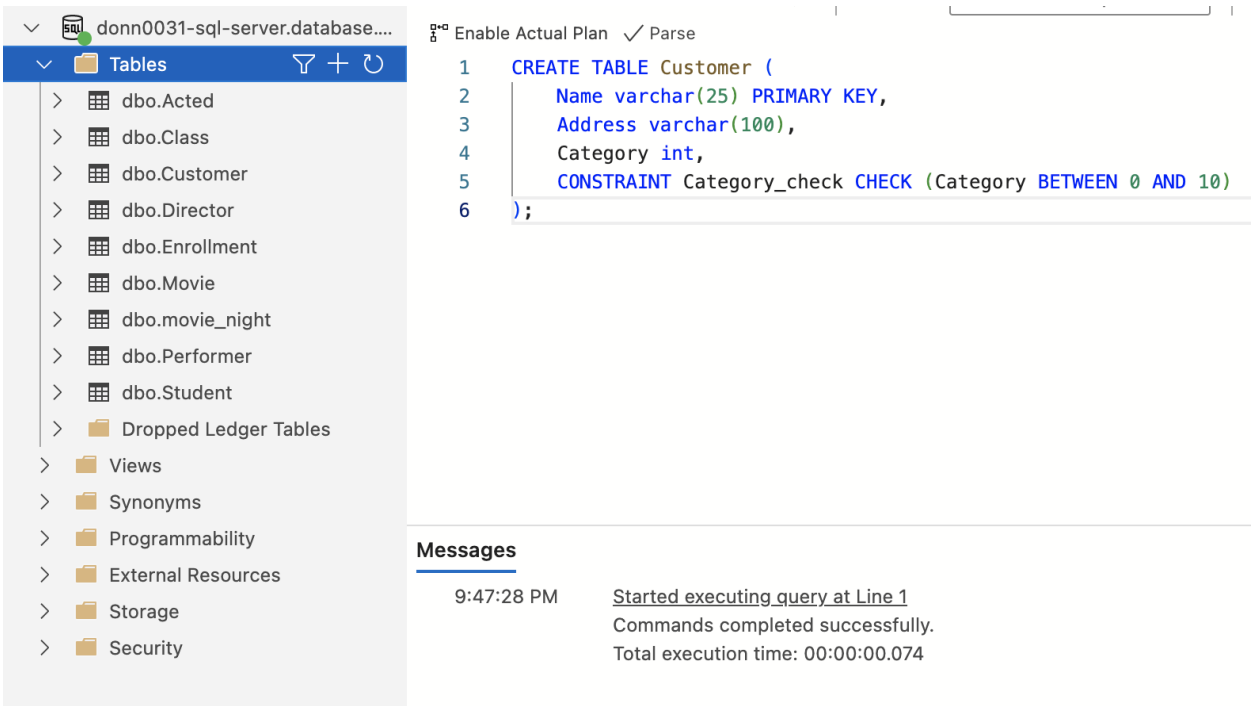
Heaps (Tables without Clustered Indexes) (docs.microsoft.com)

TASK 4

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

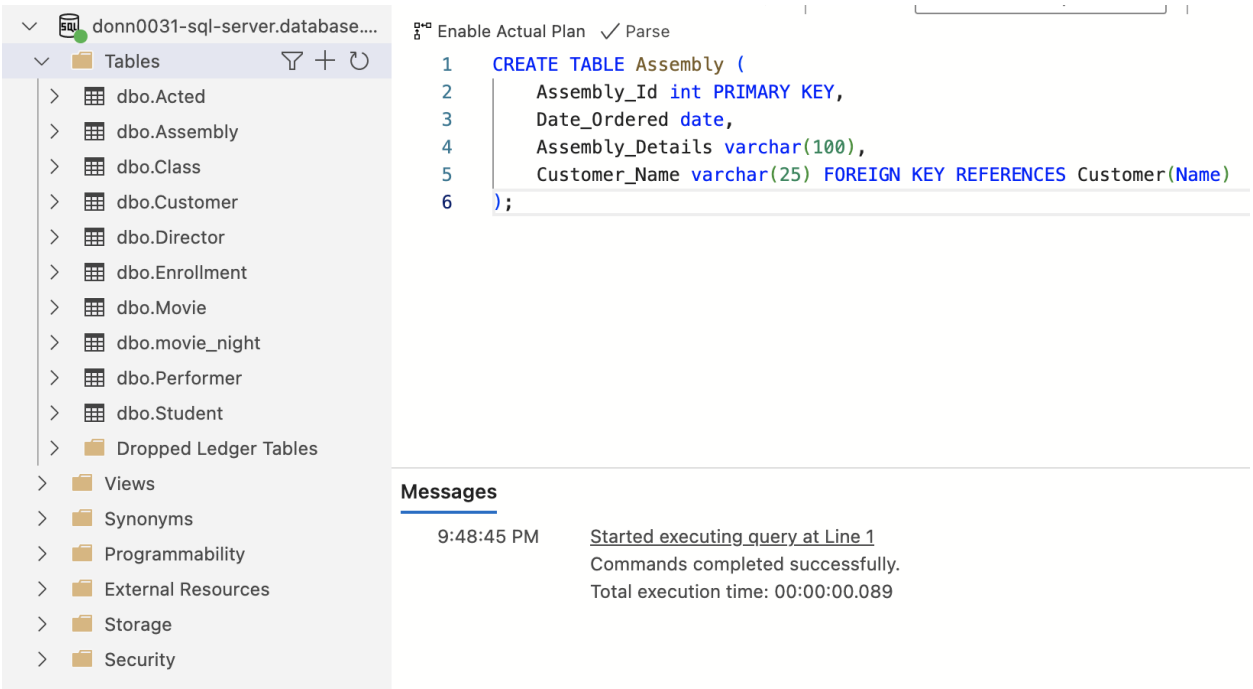
Create Table Statements

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




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

);

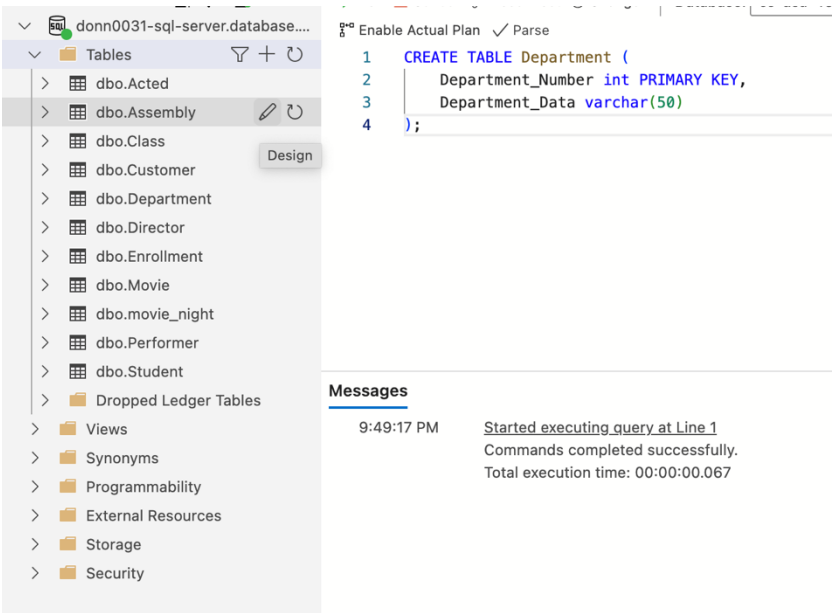


CREATE TABLE Department (

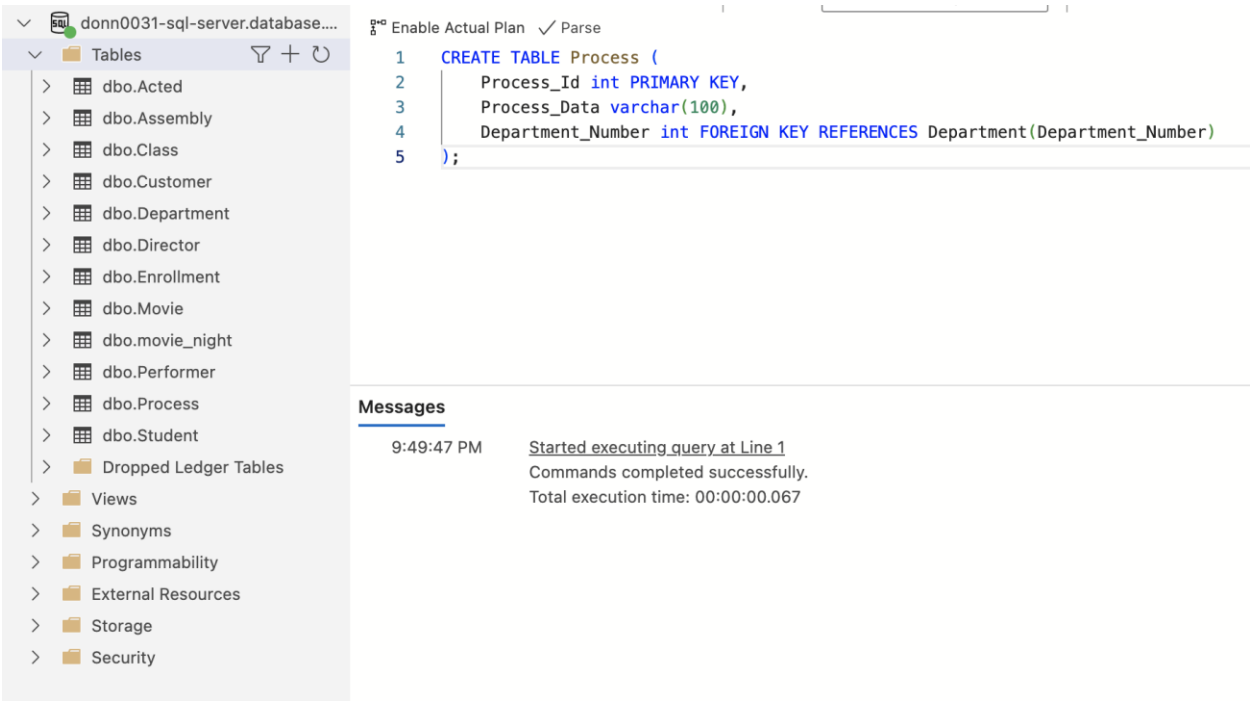
Department_Number int PRIMARY KEY,

Department_Data varchar(50)

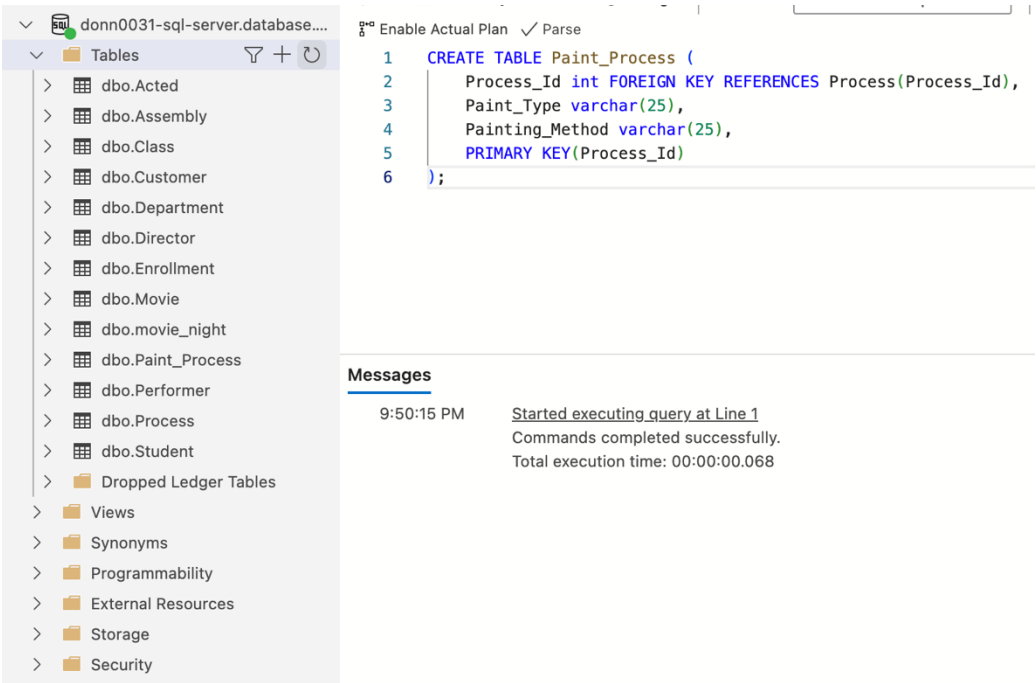
);



```
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)
);
```



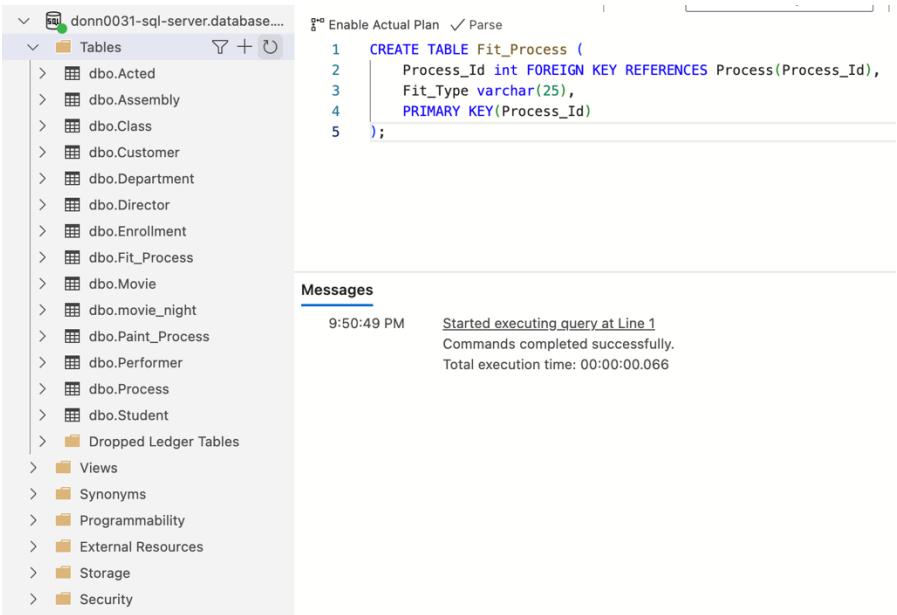
CREATE TABLE Fit_Process (

Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

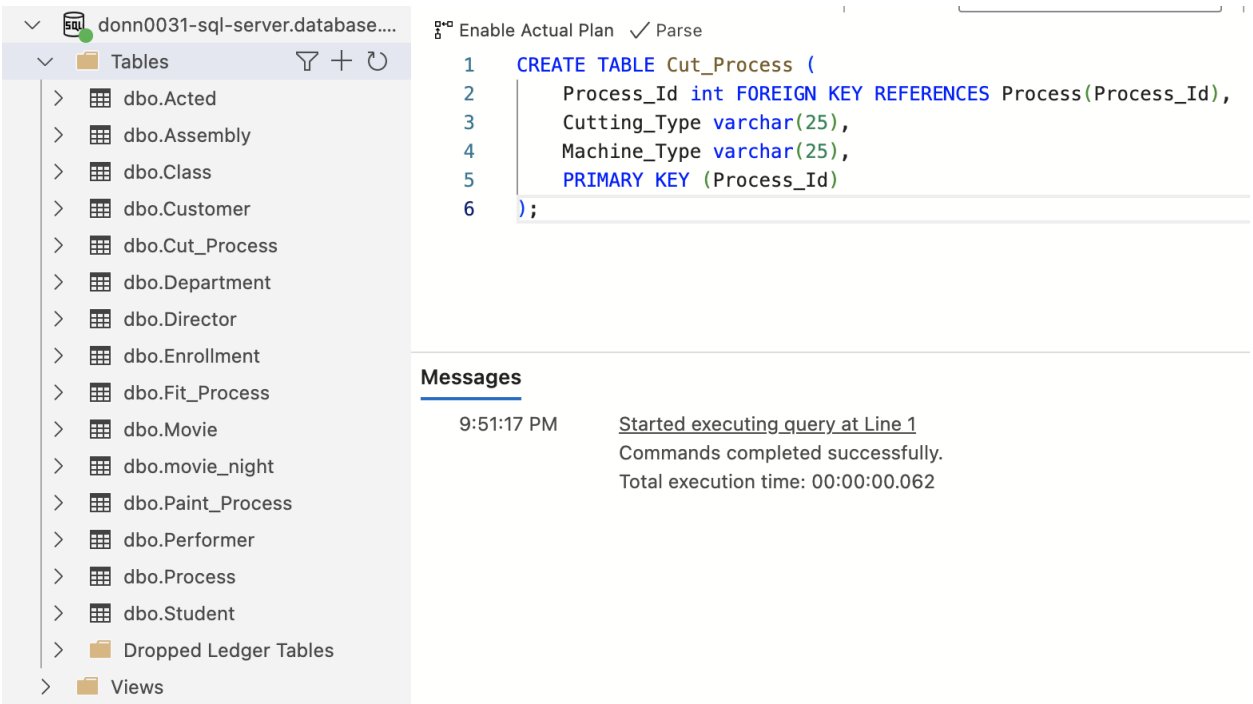
Fit_Type varchar(25),

PRIMARY KEY(Process_Id)

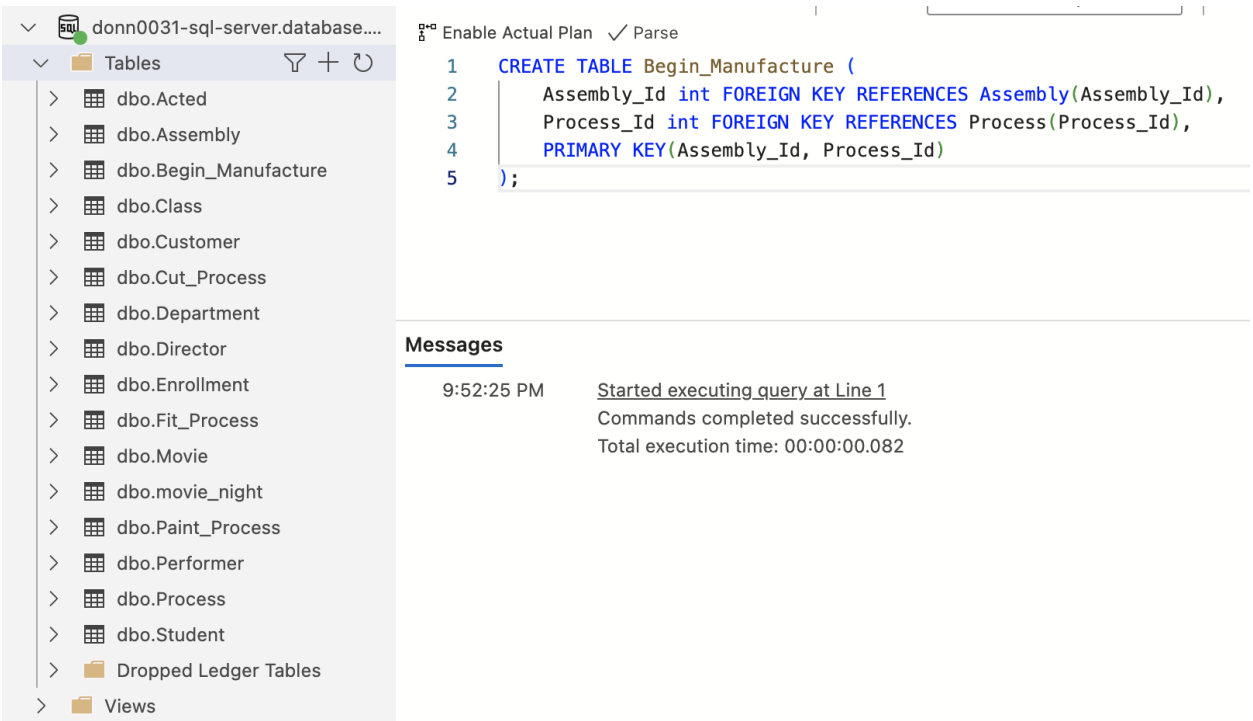
);



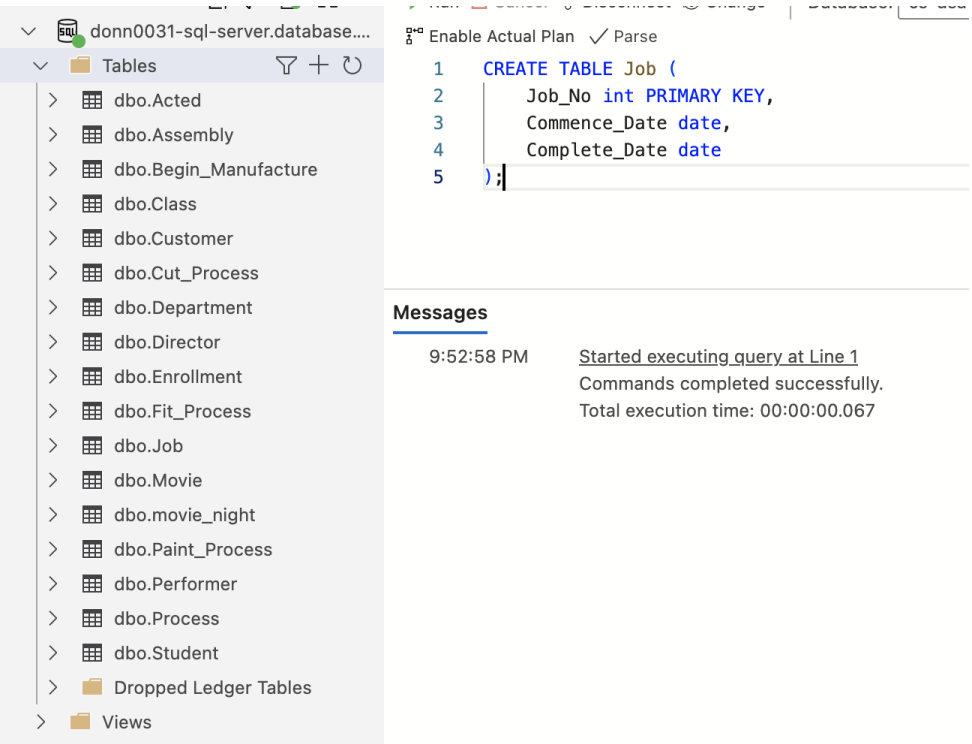
```
CREATE TABLE Cut_Process (
        Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
        Cutting_Type varchar(25),
        Machine_Type varchar(25),
        PRIMARY KEY(Process_Id)
);
```



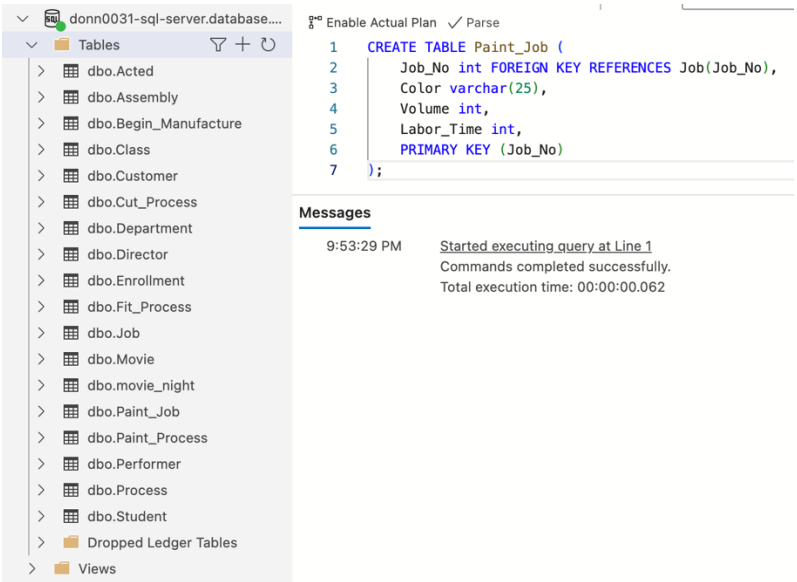

```
CREATE TABLE Begin_Manufacture (
        Assembly_Id int FOREIGN KEY REFERENCES Assembly(Assembly_Id),
        Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),
        PRIMARY KEY (Assembly_Id, Process_Id)
); -- this was changed from Begin to Begin_Manufacture to not confuse SQL BEGIN syntax
```




```
CREATE TABLE Job (
        Job_No int PRIMARY KEY,
        Commence_Date date,
        Complete_Date date
);
```




```
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)
);
```



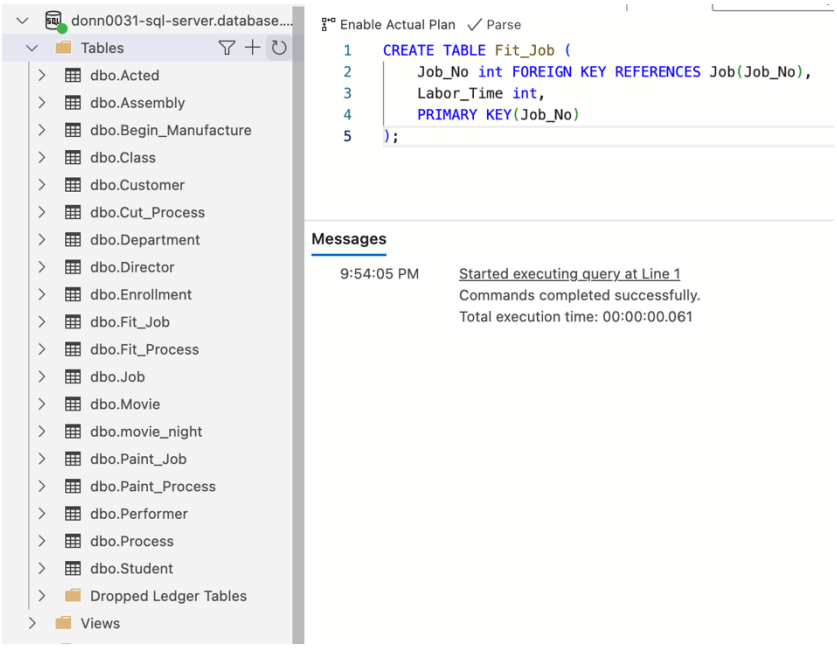
CREATE TABLE Fit_Job (

Job_No int FOREIGN KEY REFERENCES Job(Job_No),

Labor_Time int,

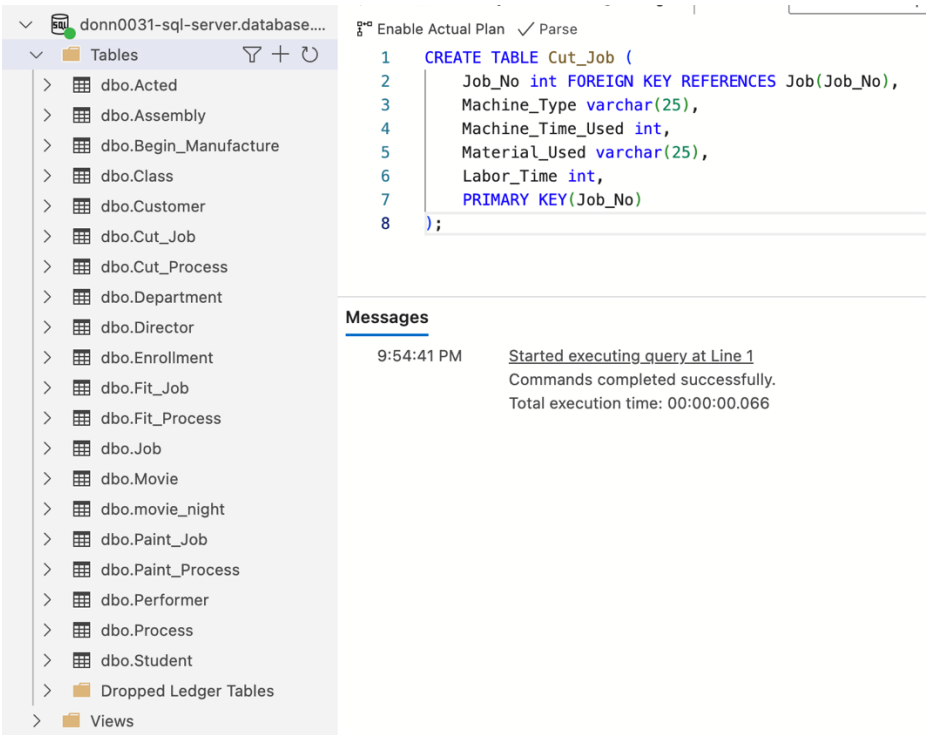
PRIMARY KEY(Job_No)

);

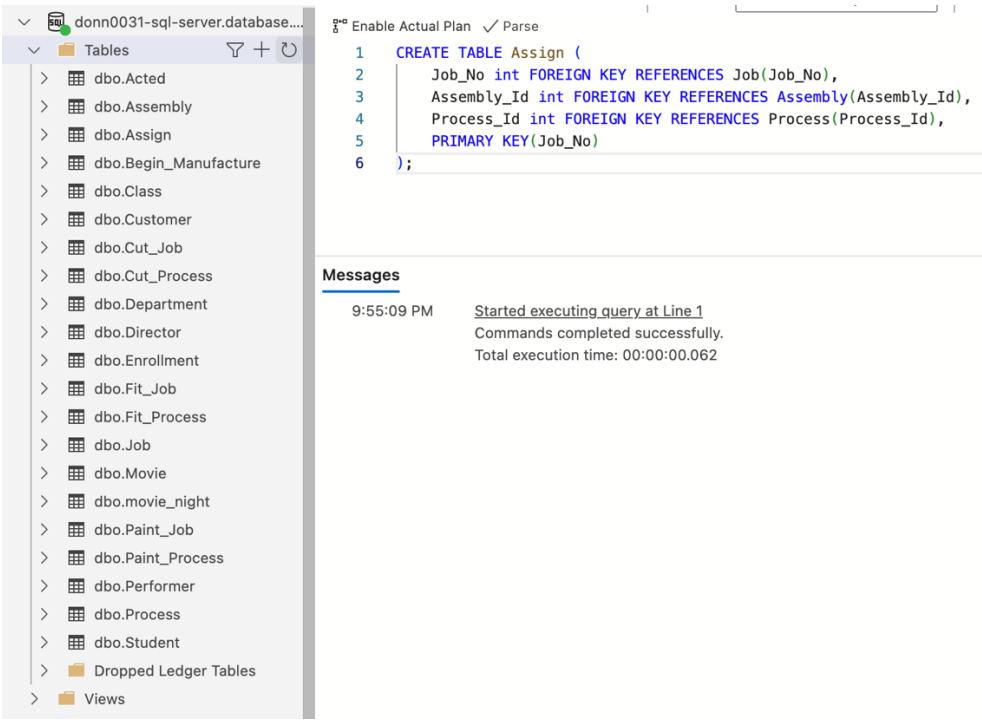


CREATE TABLE Cut_Job (

```
    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)
);
```

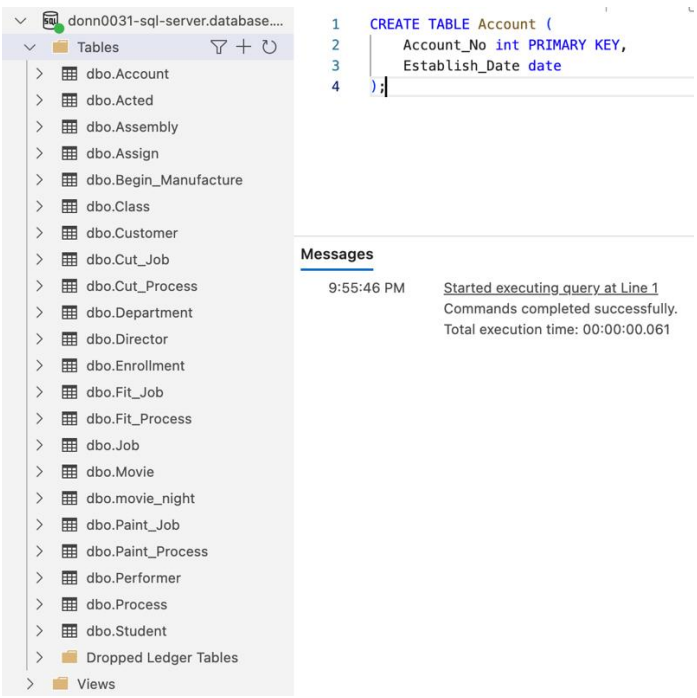


CREATE TABLE Account (

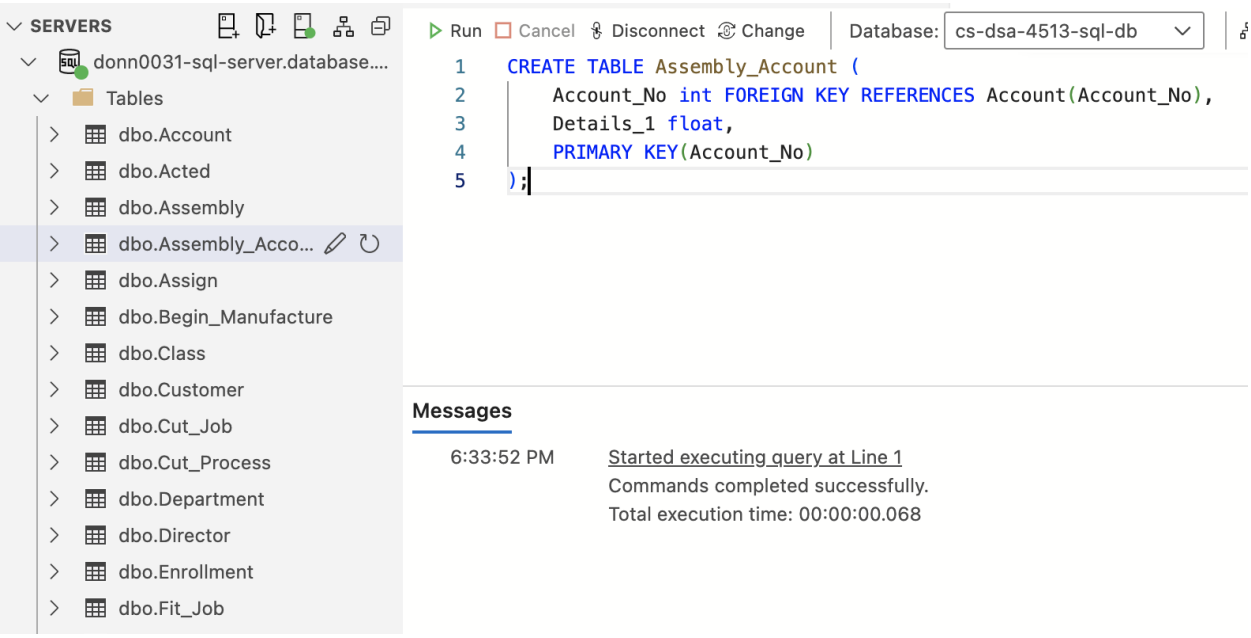
Account_No int PRIMARY KEY,

Establish_Date date

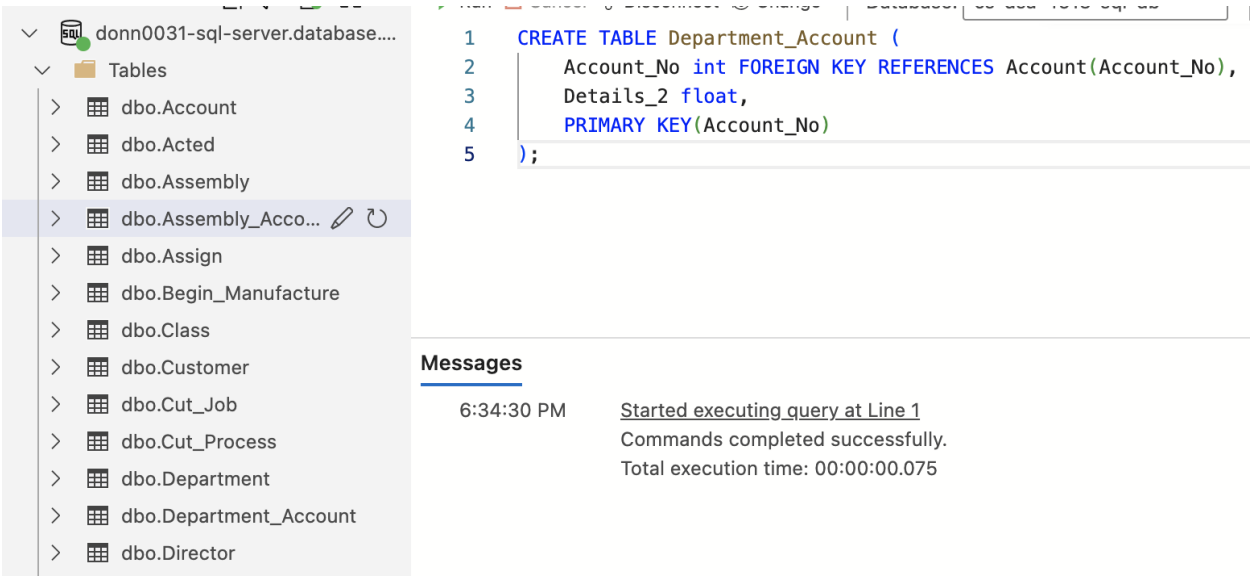
);



```
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)
);
```



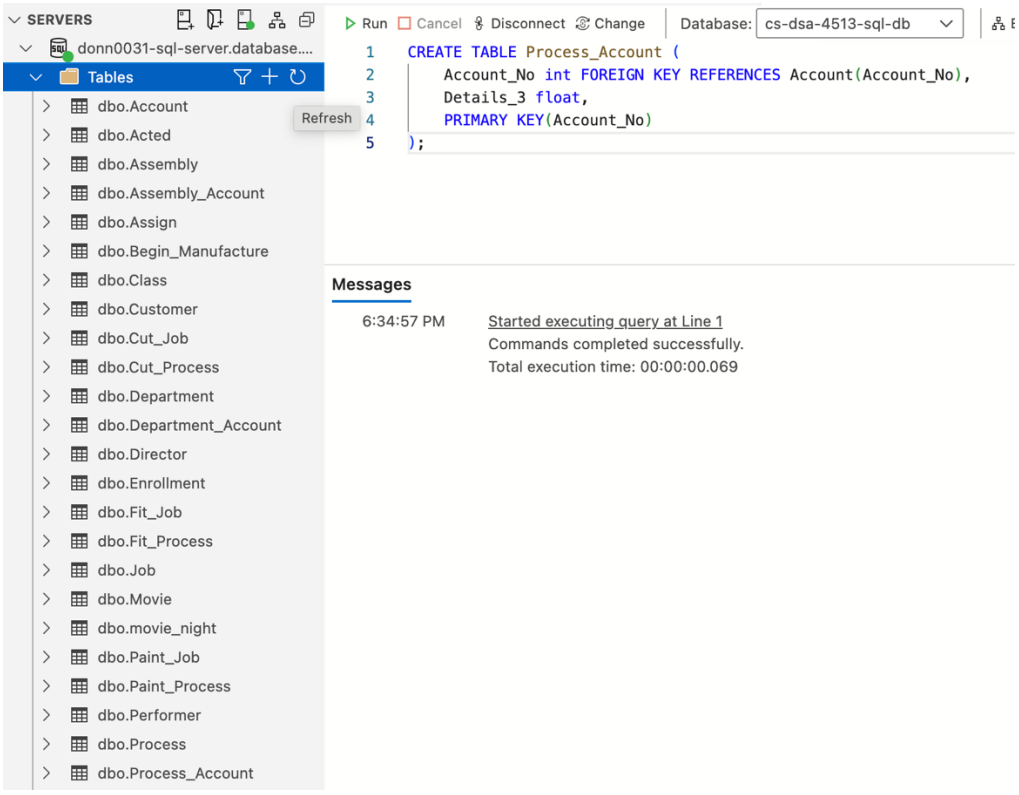
CREATE TABLE Process_Account (

Account_No int FOREIGN KEY REFERENCES Account(Account_No),

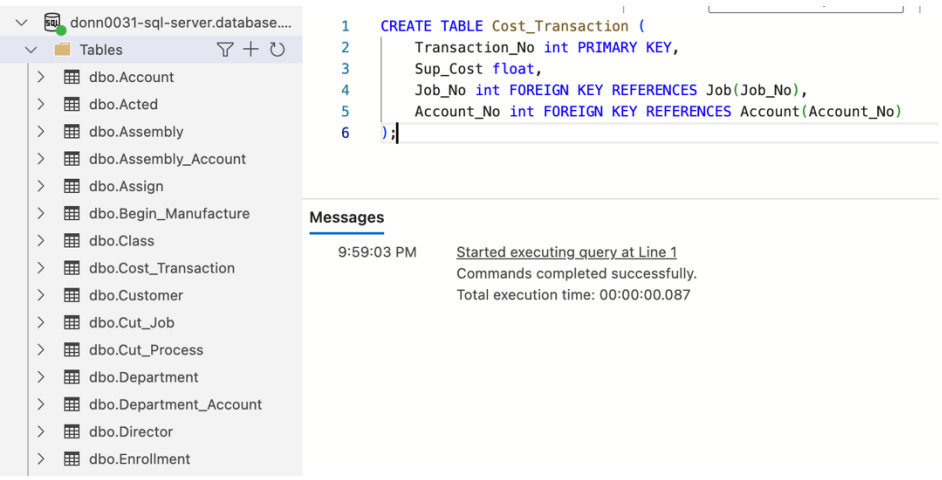
Details_3 float,

PRIMARY KEY(Account_No)

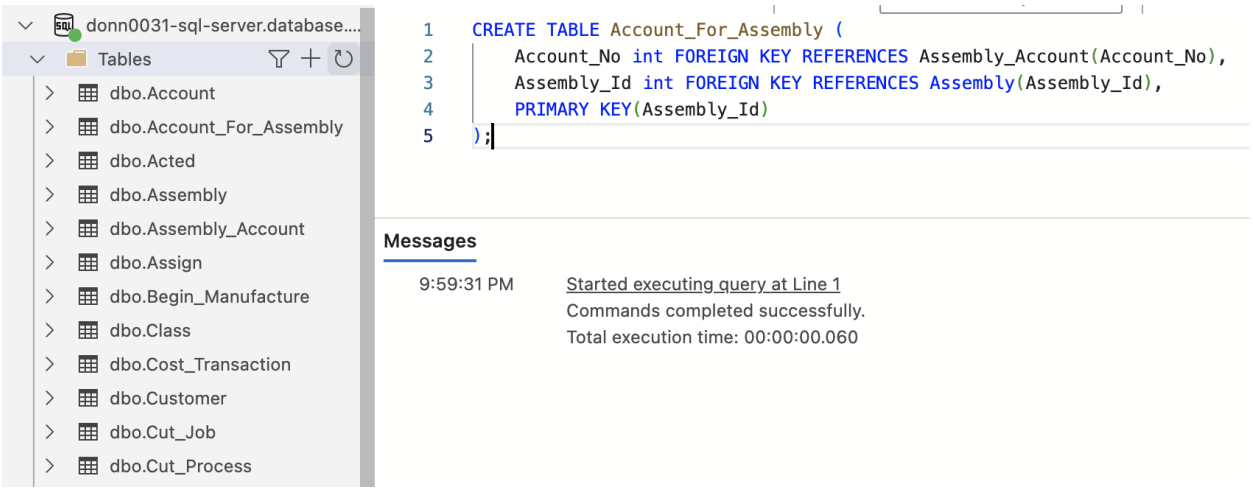
);



```
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)
);
```



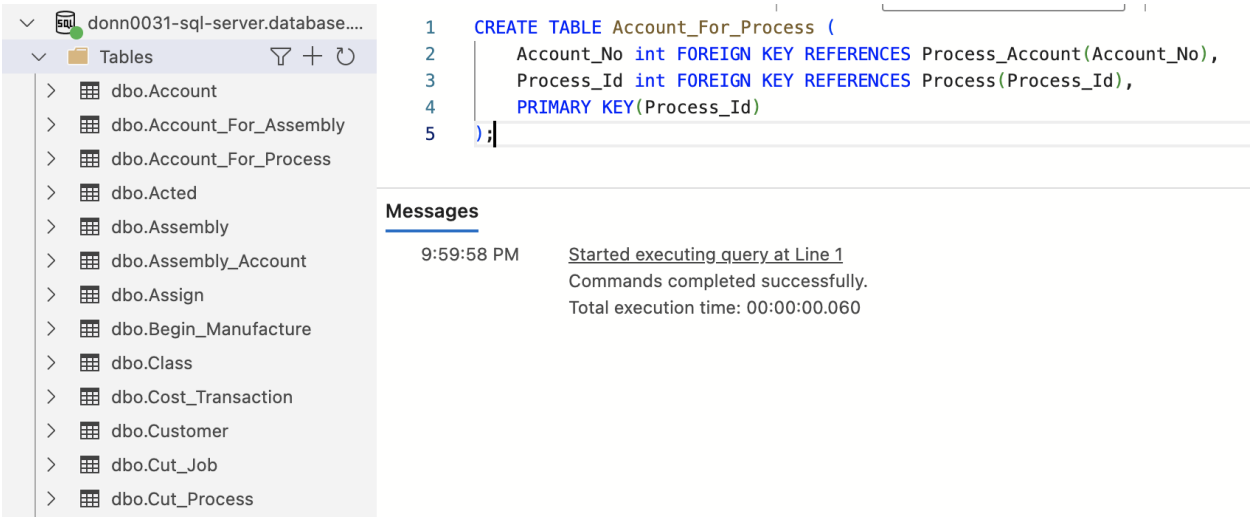
CREATE TABLE Account_For_Process (

Account_No int FOREIGN KEY REFERENCES Process_Account(Account_No),

Process_Id int FOREIGN KEY REFERENCES Process(Process_Id),

PRIMARY KEY(Process_Id)

);



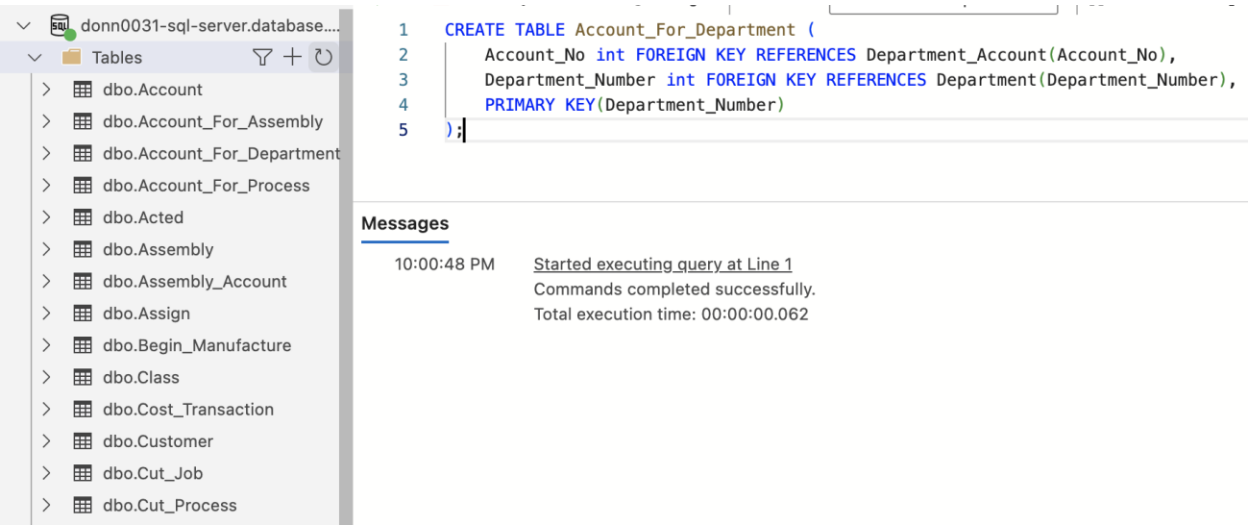
CREATE TABLE Account_For_Department (

Account_No int FOREIGN KEY REFERENCES Department_Account(Account_No),

Department_Number FOREIGN KEY REFERENCES Department(Department_Number),

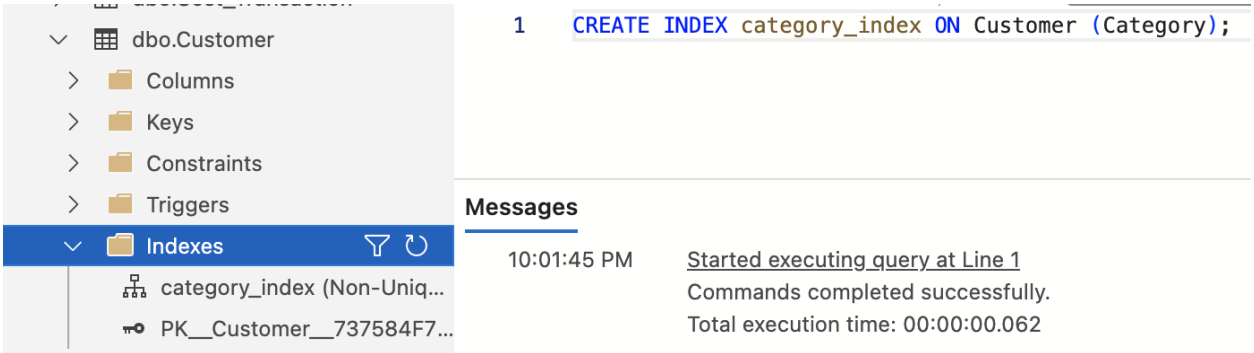
PRIMARY KEY(Department_Number)

);

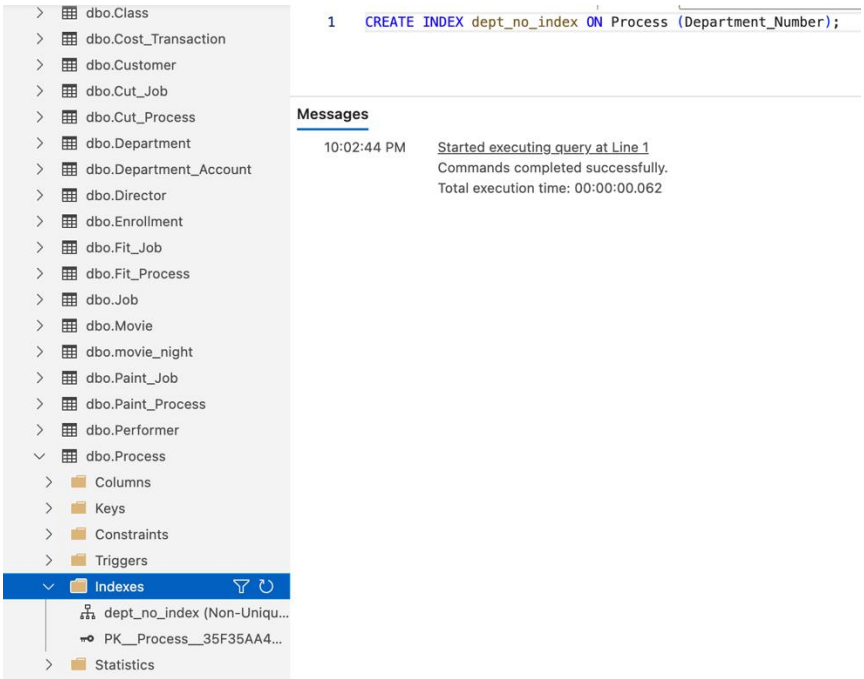


Create Index Statements

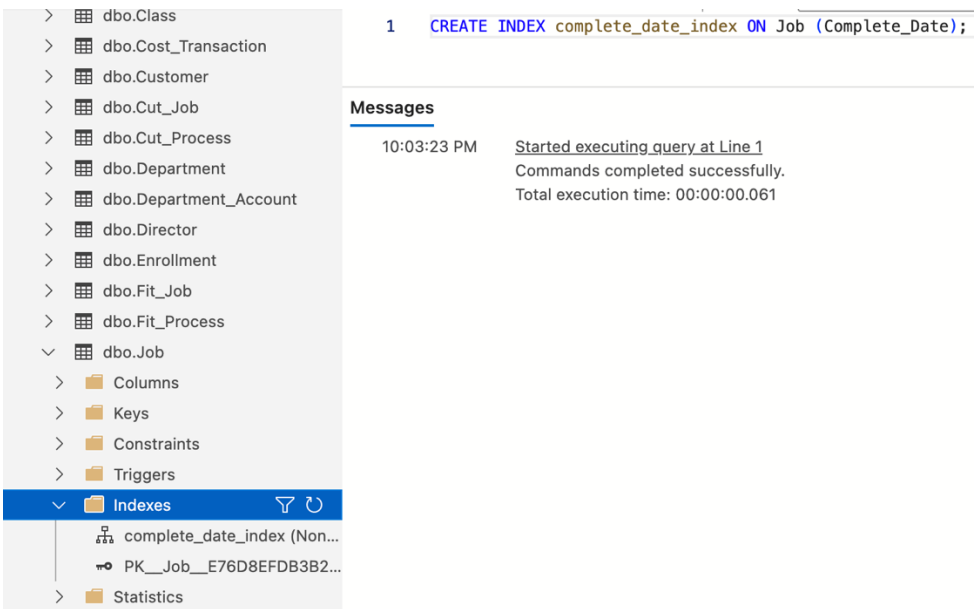
CREATE INDEX category_index ON Customer (Category);



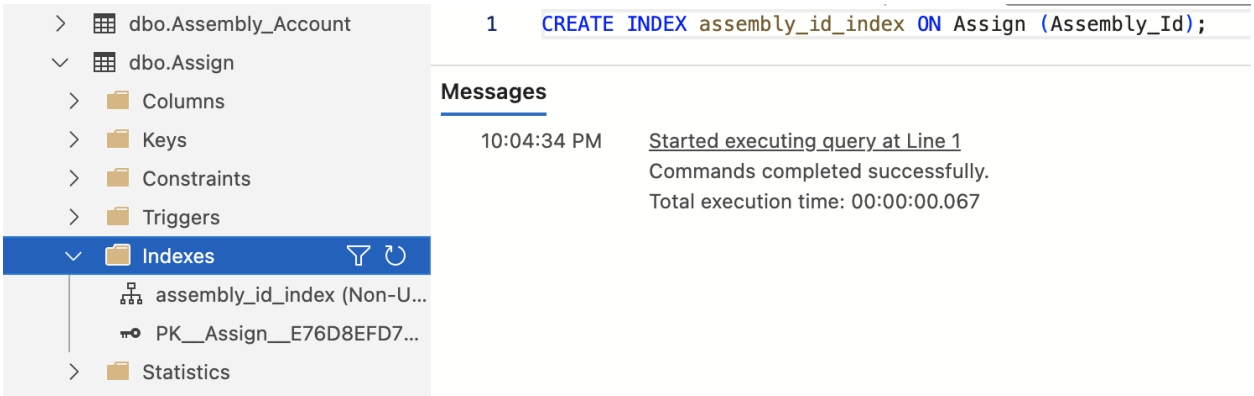
CREATE INDEX dept_no_index ON Process (Department_Number);



CREATE INDEX complete_date_index ON Job (Complete_Date);



CREATE INDEX assembly_id_index ON Assign (Assembly_Id);


dbo.Assembly_Account
dbo.Assign
Columns
Keys
Constraints
Triggers
Indexes
assembly_id_index (Non-U...
PK__Assign__E76D8EFD7...
Statistics
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